UAV-enabled Software Defined Wireless Sensor Network for Data Gathering

Pejman Karegar

A thesis submitted to

Auckland University of Technology

in fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD.)

School of Engineering, Computer & Mathematical Sciences

Primary Supervisor: Prof. Peter Han Joo Chong

2023

Abstract

Recent advances in Unmanned Aerial Vehicle (UAV) technologies make their use as data collection (sink) feasible for practical applications of Wireless Sensor Networks (WSN) and the Internet of Things (IoT). Using UAVs as mobile data ferries can amplify ground network performance by mitigating the impact of horizontal radio pollution on the environment and replacing that with individual nodes or local group of nodes vertically communicate with opportunistically accessible UAV. This can reduce packet loss performance and energy use on a per device basis, as reduced routing responsibilities require significantly less radio activity.

This research aims to develop a UAV-aided WSN model that provides energy efficient and scalable data gathering method and supports numerous applications such as forest fire monitoring, monitoring of microclimates, flood monitoring and monitoring of forest predator's traps, to name few possible applications. One of the challenges for the UAV-assisted WSN for data gathering efforts is the design of an energy-efficient UAV communication with randomly distributed ground sensors by improving the ground network structure dependent on the UAV path. Hence, in keeping with the context of UAV energy efficiency challenges and solutions, the proposed model incorporates three parts; the UAV trajectory planning in which the smooth UAV route is constructed, the UAV-Ground connectivity wherein the UAV data link communication is initiated based on the UAV's position in the selected path, and the wireless communication within the ground sensor nodes (SN) wherein sensors collaborate to organise data transmission to the UAV.

The first part is focused on identifying a flexible span for the UAV path design by conceptualizing an approach called 'UAV Fuzzy Travel Path' that supports UAV smooth trajectory planning and facilitates ground network re-orchestration for data gathering from spatially dispersed wireless sensors over a large space. This organisation enables the UAV path to be dynamically adjusted in accordance with the updated ground network structure and allows for defining an adaptable span for UAV path design instead of being fixed into a static route. Crisp circular/linear UAV path and Bezier Curve TSP (BTSP) path are two examples of smooth paths aligned on a variety of ground network formations here.

The second part proposes the dynamic orchestration of ground wireless sensors grouping for improving the network performance in data collection. The term 'Software Defined Wireless Sensor Network' (SDWSN) will be used broadly in this thesis to describe the flexibility of the ground network architecture. This offers a more dynamic and adaptive network that interacts with the UAV planned path. The SDWSN-enabled ground network communication is mainly divided into two major phases: the scanning topological pre-orchestration and sensing data collection post-orchestration phase. As the primary goal of data collection effort is on passing the sensing data information messages to the drone via data collection phase, passing the ground network topological orchestration data and assigning the optimal functionalities to each component of the network are operated via scanning pre-orchestration phase.

The third part presents the key prerequisites for designing a generic model for air-to-Ground network that takes the mobile network topology into account. The air-to-Ground communication structure via the proposed communication window of connectivity mechanism is expressed to provide a connection among the ground network and the UAV. The proposed communication design is identified based on two approaches: air-to-Ground communication with each individual sensor nodes or air-to-Ground communication with each group representative nodes serving as cluster heads. Both techniques attempt to offer a reliable and consistent communication among the sensor network and the drone.

The main evaluation and performance metrics for the first part focus on testing the UAV propulsion energy consumption and received packets on the UAV receiver while accounting for smooth paths. The theoretical and actual analysis of the proposed UAV path designs prove that the designed crisp circular/linear UAV path improves energy efficiency for propulsion energy consumption with higher packet delivery rate on the UAV receiver while compromising ground network energy usage compared to the BTSP. The key performance measures for the analysis of ground network are packet delivery and energy consumed for the overall ground network when dealing with different network densities and message rates. The analytical results of both scanning and sensing data collection phases mentioned above confirm that the ground network orchestration plays a crucial role in serving more SNs on the ground and enhancing UAV energy efficiency when compared to the situation prior to orchestration. Additionally, the impact of UAV velocity and network distribution on the percentage of received packets on the UAV receiver is evaluated in the air-to-Ground communication evaluation section. The

simulation outputs show that the proposed SDWSN-enabled data gathering modelling boosts packet delivery rate on the UAV-Ground connectivity when compared to the situation prior to network orchestration.

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Pejman ABD 18.01.2023

Signature Date

Acknowledgements

First and foremost, I am extremely grateful to Prof. Adnan Al-Anbuky for his invaluable input in my PhD. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. Special thanks to Prof. Peter Han Joo Chong for his supervision, worthwhile advice, continuous support, and patience during my PhD study. He believed in my work and guided me to completion the time that I felt like giving up. I owe a special thanks to my colleague Duaa Al-Hamid who was always a joy to collaborate with in every aspect of the project from scientific discussions to the development of testbeds. I would like to thank all the members in the SenSe Research Lab Centre. It is their kind help and support that have made my study and life in the NZ a wonderful time. I also express my sheer gratitude to the 'Callaghan Innovation' for their offering the funding for this project.

Most importantly, I would like to dedicate my thesis to the beautiful soul of my mother, Toran Rahatlouei Nejad, who always believed in my ability to be successful in the academia arena. For me, becoming a doctor was part of her dream and I'm glad I'm fulfilling that dream. You are gone, but your faith in me made this journey possible. Also, I would like to express my gratitude to my dad, Hassan A. Karegar, for his countless blessings, incredible faith and encouragement that encouraged me during this demanding phase of my student life. I am extremely gratitude to my brothers, Salar A. Karegar and Dr. Makan A. Karegar for their kind supports and guiding me in every aspect of my research work and my sister Dr. Baharak A. Karegar for her kind support and caring. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Contents

List of Figures

List of Tables

List of Terms and Abbreviations

Ack	Acknowledgements
ACO	Ant Colony Optimization
App	Application
ARM	Advanced RISC (Reduced Instruction Set Computer) Machine
AUT	Auckland University of Technology
AVR	Advanced Virtual RISC (Reduced Instruction Set Computer)
BLE	Bluetooth Low Energy
BS	Base Station
BTSP	Bezier curve TSP
CCS	Code Composer Studio
CDG	Compressive Data Gathering
CH	Cluster Head
CPU	Central Processing Unit
CSMA-CA	Carrier Sense Multiple Access-Collision Avoidance
CTEA	Centroid-based Trajectory Enhancement Approach
GA	Genetic Algorithm
GUI	Graphical User Interface
IBM	International Business Machines
ID	Identification
IDE	Integrated Development Environment
iOS	iPhone Operating System
IoT	Internet-of-Things
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LoS	Line of Sight
LPM	Low Power Mode
LPSTK	LaunchPad SensorTag Kit
LQ	Link Quality
MAC	‘Media Access Control’ Layer
MAVLink	Micro Air Vehicle Link
MCU	Microcontroller Unit
MQTT	Message Queuing Telemetry Transport

MSP	Mixed Signal Processing
NS-2	Network Simulator-2
NS-3	Network Simulator-3
NLoS	Non-Line of Sight
OAD	Over-the-Air Debugging
OMNET++	Optical Micro-Networks Plus Plus
OPNET	Optimized Network Engineering Tool
OS	Operating System
OSM	OpenStreetMap
PC	Personal Computer
PDP	Pick up-and Delivery Problem
PDR	Packet Delivery Ratio
PSO	Particle Swarm Optimization
QoS	Quality-of-Service
RF	Radio Frequency
RL	Reinforcement Learning
Router-L1	Router Level 1
Router-L2	Router Level 2
RPi	Raspberry Pi
RSSI	Received Signal-Strength Intensity
RTL	Return to Launch
RX	Reception
SCA	Successive Convex Approximation
SDN	Software Defined Networking

SDWSN	Software Defined Wireless Sensor Network
SeNSe	Sensor Network and Smart Environment Research Centre
SITL	Software in the Loop
SN	Sensor Node
SSID	service set identifier
SysConfig	System Configuration
TDMA	Time-Division Multiple Access
TI	Texas Instruments
TI-RTOS	Texas Instrument - Real Time Operating System
TSP	Travel Salesman Problem
TX	Transmission
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
Vtol	Vertical take-off and landing
V-SITL	Virtual IP based Software In The Loop
WSN	Wireless Sensor Networks
2D	two-dimensional
3D	three-dimensional

Chapter 1 Research Motivation, Direction and Thesis Organization

1.1 Introduction

This chapter begins with a general overview of UAV-based WSN data gathering and the typical path planning approaches adopted while ensuring the reliable communication among the ground networks and the UAV. It then delves into the aspect of ground network structure and the challenges encountered by the network due to the UAV dynamics. As the possible ground structure based on trap distribution use case is discussed, the UAV path design approach including the involved components to formulate and maintain a relaxed path is highlighted. Based on this, the challenges within the dynamic ground structure and the interaction with the cloud to provide the full data gathering architecture are identified and put forth. The motivation behind driving towards formulation of flexible grouping approach that has the ability to adapt with UAV path and re-orchestration demands is highlighted. Also, the role of virtualization, Wireless sensor network (WSN), and softwarisation is stated before laying the research question. The latter stages of the chapter are devoted towards brief discussions on the objectives so specified towards realization of a Software-Defined Wireless Sensor Network (SDWSN) organization applied on UAV path design. The research output in terms of publications is then listed followed by the chapter conclusion.

1.2 UAV-based WSN Data Gathering: Path Planning and Communication Strategies

Recently, WSNs have witnessed explosive growth among the research hot topics. In this thesis, typically the term ‘WSN’ tends to be used to refer to a number of inexpensive, battery-powered and energy-constrained dispersed sensor nodes (SNs), each of which is restrained to limited energy resources which makes it challenging to replenish after use and has the capabilities to perform a certain task cooperatively [1]. In order to extend the lifespan of the entire network, energy-efficient communication strategies amongst SNs are essential. There are various approaches to mitigate the energy usage amongst ground networks. One solution is through the usage of UAVs which offers more flexibility and manoeuvrability in data capturing use cases. In this thesis, the terms ‘Drone’ and ‘UAV’ are used interchangeably to refer to any aerial vehicle that enables vertical communication

with the ground network. From the communication point of view, data collection based on the interaction among the UAV and ground network mostly differs from that in a typical WSN. Since modern UAVs are able to adjust their motion in order to achieve the network demands and hover/loiter over a specific hot spot for data collection, they offer a distinctive capability to data gathering efforts [2].

In keeping with the context of WSN and its role in UAV-based data gathering, adopting several technologies and paradigms can enhance the structure of ground network and communication strategies for improved data gathering performance. The concept of self-adaptive and re-configurable network has gained substantial momentum to offer a new level of flexibility for the WSN-enabled data gathering approach. This, in effect, requires highly functional sensing, communication, and computational capabilities compared to conventional network systems. This encourages the utilization of cloud computing that facilitates computational capability in the Base Station (BS). Figure 1-1 indicates the architecture of proposed data collection method via the UAV and distributed SNs for trap use cases in the conservation. According to the proposed used-case scenario, the static SNs are chosen for data gathering from the distributed traps as it is not necessary to adopt mobile nodes for data collection from traps in the forest. The reason behind that is that using the mobile nodes in the forest may lead to communication blockages that may happen for the mobile SNs. Moreover, due to the rough terrain shape in some areas of the forest, the mobile nodes may get stuck in some areas resulting in losing those nodes.

The UAV plays a crucial role in facilitating the exchange of information as it can establish a connection with BS via its own messaging system so called Micro Air Vehicle Link, ‘MAVLink’ [3] which is a long-range frequency hopping binary telemetry for communication among drones and BS. MAVLink follows a modern hybrid publish-subscribe and point-to-point communication design pattern. The UAV autopilot uses MAVLink to communicate with the software installed on the BS to control the UAV’s movement over the Radio Frequency (RF) protocol. RF communication is widely used in the drone industry due to its long-range capabilities and resistance to interference. Hence, it can be used to transmit the instant data such as GPS location, orientation and speed of the UAV to the BS. MAVLink is deployed in two major versions: version 1.0 and version 2.0. The BS whether it is a stationary or a mobile terminal should have the ability of communication with external network to match user preferences and QoS requirements.

According to the proposed strategy, the UAV communicates with the BS depending on the round of UAV travel and the collected data from SNs on the UAV processor are not repeatedly transmitted to the BS. If it is assumed that the UAV flies over distributed SNs in the first round called scanning phase, the UAV acquires the SNs control data during the scanning phase. Then, within its own processor the election process for updated ground network formation can be done without needing to offload the raw data to the BS. While in data gathering phase, where the UAV gathers the sensing data from SNs, the UAV offloads the collected sensing data to the BS via the Wi-Fi network.

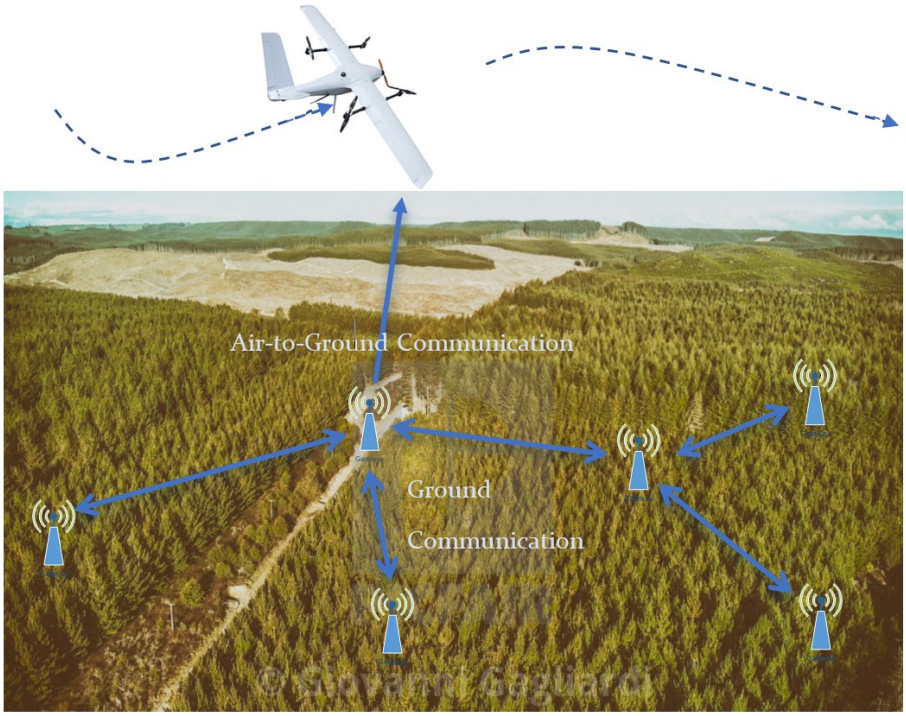


Figure 1-1 The architecture of proposed data collection system model.

The suggested communication protocols amongst the network components are suggested as: dual Zigbee and Bluetooth for the communication among the ground network entities and Wi-Fi for enabling the communication between the UAV and the gateway nodes.

1.3 UAV-based WSN Data Gathering: Approaches and Challenges

Data gathering via WSNs typically consists of several distributed SNs which have the cooperation capabilities to forward the sensing data to a central station. However, the area presents numerous research challenges that should be tackled. Radio pollution is critical once it relates to the ground communication [4, 5]. This is an obvious issue apparent in

long distance communication of SNs in the conservation such as the impact of radio pollution on living organisms in a forest. Alternative approaches are to use vertical communication schemes such as communicating with UAV or satellite [6, 7]. Communication utilising satellite is regarded as an energy-hungry method for ground sensor network while UAV as a mobile data agent is introduced as an efficient potential driver to collect data from the distributed ground SNs. Hence, the UAVs are employed to offer a reasonably less polluted solution for data gathering from the dispersed ground SNs. This also allows the WSN to pass their data through SNs' representatives on the ground and then forward the buffered data to the UAV with higher percentage of LoS connectivity over vertical air-to-Ground communication rather than having plain horizontal on-the-ground communication with the high possibility of NLoS connectivity. Data collection by utilising the UAV from ground sensor network also offer more flexibility and manoeuvrability. Hence, this not only enhances the WSN coverage and throughput, but it noticeably reduces energy usage of the ground network entities by scheduling their communications based on the UAV's presence. Hence, scheduling of the ground network representatives to have an energy-conserved data transmission with the UAV in order to improve the packet delivery in the UAV-Ground communication has become another challenge in recent years.

Since modern UAVs are capable in manoeuvring and loitering over a given spot for data collection, dynamic variations in motion speed and acceleration may cause a substantial energy cost. Hence, minimising the variation of travelling velocity and acceleration taking advantage of designing low-complexity and crisp trajectory that enables the UAV to interact with ground network during its travel is another hot topic amongst the research topics [8].

From the communication perspective, there are additional advantages to plan for an appropriate coordination between the UAV and the related ground data collection devices in a way that minimises the energy needed for communication among them by utilising stable communication among the UAV and ground SNs [9]. Here alongside the UAV trajectory, identifying and scheduling of the in-line ground SNs to set the flow of data from the most appropriate ground to the UAV router is also an important factor for enhancing the collection of ground data with least cost. Hence, the proposed solution for distributed sensors data collection using UAV comprises of three main sub-solutions. These are: energy-efficient ground WSNs representing traps grouping, reliable UAV-

Ground communication approach, power-efficient UAV path-planning model. These three major areas have not been looked at as a full package in enhancing the performance of data collection in the conservation. Each of the main sub-solutions is represented by the main factors that may influence the structure of the proposed system as discussed in section 4. The proposed scheme can be scalable, suitable for medium-large environments with a large number of SNs and can provide a pervasive and ubiquitous computing system based on an adaptive WSN. As a result, our proposed model, which is applied in each of the three parts, can be flexible in terms of environmental scaling and taking into account the energy aspect of the UAV and SNs on the data gathering effort. In addition, the major performance measures are highlighted to pave the way to the main metrics that could affect the proposed system. The outcome of the whole system architecture can effectively contribute to minimize the UAV energy consumption and the sensory data packet loss that results in prolonging the lifetime of the SNs.

1.4 Motivation for Driving Towards SDWSN Dynamically Flexible Grouping Approach

Based on the conventional UAV-based WSN data gathering arrangement, the UAV path is restrained to limited options for collecting the data once physical topology of each group of ground sensor nodes is set. Setting the gateways to be selected dynamically offers adaptability to the ground network global topology taking the service demands into account. This allows for the groups topologies to be flexibly re-organise through software. The proposed concept of creating what we refer to as a ‘Fuzzy flight Path’ utilizes the capability to software redefine the groups representatives or network topology to align with the network requirements. With regard to offering the network dynamicity to the ground network in UAV-enabled data gathering use case, Software-Defined Network (SDN) approach can offer a promising solution, where the control plane is separated from the data plane using a central controller such as a cloud computation processor. The separation of these two planes enables intelligent traffic routing and optimal re-orchestration of network structures to support WSN in terms of network energy usages [10]. This thesis intends to take advantage of fuzzy path to allow the UAV to communicate with updated ground representatives using SDN re-orchestration. As shown in Figure 1-2, the proposed network communication structure plan is based on dividing the entire system into three core communicating sub-units: Ground Network, Drone, Cloud stations. Ground Network may contain three types of nodes that are identified as

three main roles in WSN. These are Leaf, Router and Gateway roles, each of which has its own responsibility to pass the sensing data to the remote cloud server.

Hence, an SDWSN is a viable solution for seamlessly resolving dynamic re-orchestration and service demands in real-time by relying on a software solution rather than hardware ones through introduction of an enabling technology such as virtualization [11]. Chapter 4 contains a detailed discussion of the SDWSN network.

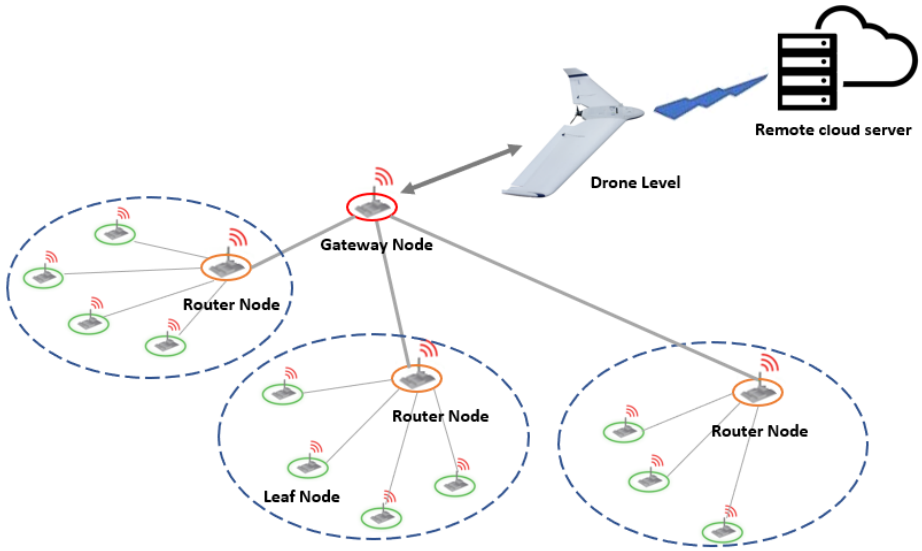


Figure 1-2 The architecture of proposed SDN-enabled data collection system model.

The concept of UAV path adaptability with the ground sensor nodes offers a flexibility for the entire network ensuring either UAV or the sensor nodes' energy efficiency improvement. Hence, timely distributing the workload of collected data among a variety of ground representatives via multiple ground network structures can offer a host of advantages to the whole system. This includes improving the energy efficiency of the ground sensor network and allowing the UAV path design to be selected from multiple routes rather than being restricted to one route. This ideology can eliminate the use of a single fixed architecture by dynamically distributing the workload of data across multiple ground network entities rather than a single one. This also allows for designing a crisp path with lower number of sharp edges to lower the UAV energy consumption for each planned travel. The proposed UAV-enabled WSN communication takes the main factors such as ground network parameters (SN's distribution, density, topology, etc.), UAV parameters (length of the path, UAV velocity and acceleration, number of operations,

etc.) and communication parameters (Bit rate, transmission range, etc.) into account for its modelling.

1.5 Objective, Research Questions, and Research Scopes

The objective of this research work is to develop an energy-efficient and pervasive approach that supports UAV smooth path planning and facilitates ground network orchestration for data collection from spatially dispersed wireless sensors over a large space.

The above-mentioned background data and knowledge about UAV-assisted WSN data collection prompt the following questions to achieve the objective, which served as the inspiration for this study:

'What are the structural and operational organizations required for a viable UAV-based data collection of distributed sensing points in a large space?'

'What makes a UAV-based data collection of distributed sensing points within large space efficient?'

'What is the involvement of UAV flight path and how does it improve the travelling path and energy costs?'

'What is involved in UAV-Ground data communication and how does this relate to improving the opportunity for data capturing at a lower cost?'

'How can we establish an energy efficient wireless communication among ground nodes that makes the opportunistic interaction with the UAV more efficient?'

'How to integrate the UAV flight path with softwarization concept to benefit the UAV-based data collection?'

The following are the research scopes towards accomplishing the above aim:

- To define the generic 'Fuzzy Travel Route' concept as the UAV path span that enables the UAV flight path to be chosen from a wider range of alternatives rather than being fixed in one defined path. This organisation allows the UAV path to be dynamically adjusted in accordance with the updated ground network topology.

- To use a mathematical modelling for the propulsion energy consumption of fixed-wing UAVs as a function of the UAV's flight speed and acceleration, based on which the crisp and low complexity UAV path is identified within the specified fuzzy path.
- To establish the software defined network re-orchestration concept on ground network and exploring its impact on data collection process's robustness.
- To identify a comprehensive solution for integration among the UAV path design, air-to-Ground connectivity, and ground network communication within a large field.
- To identify, justify and implement the proposed data gathering approach to evaluate its robustness in terms of UAV movement and ground network energy efficiency.

1.6 Thesis Organisation

Chapter 1 gives an overview of the UAV-enabled WSN data collection, its recent challenges and approaches including the integration of SDWSN with the UAV path design. The background area outlined in this chapter includes the software-defined enabled ground network and the full vision towards the network architecture and its relationship with the UAV route planning. The research motivation, research problem statement, research aim and objectives, and research contribution are also discussed.

Chapter 2 provides an insight into the related literature to the important aspects associated with this research including the UAV-assisted WSN data collection path planning approaches, as well as the UAV energy efficiency aspect involved in the literature. Also, other publications on ground network communications systems including SDWSN-enabled networks based on the ground energy efficiency factor and dynamic re-orchestration and virtualisation are considered. Additionally, the literatures on the air-to-Ground communication among nodes and the UAV and the relevant performance measures have been discussed within this chapter.

Chapter 3 discusses the modelling and implementation tools used in concept development and testing. It provides an insight into the capability of various tools, including software and hardware, to investigate the concept of UAV-enabled WSN data gathering.

Chapter 4 provides the development of the proposed UAV fuzzy route data collection via SDWSN approach through the phases of the proposed concept including UAV routing, UAV-Ground Communication, and Ground Network Communication. This chapter presents the ideology of the core aspects of the proposed fuzzy path approach and its major components such as network density, distribution, etc. The formulation of UAV's path energy expenditure including crisp path and Bezier Curve path is laid out. The proposed SDWSN concept is presented which is based on dynamic ground network re-orchestration/orchestration phases taking the ground network energy model into account. Finally, the proposed air-to-Ground communication via utilising window of time has been highlighted.

Chapter 5 presents the modelling and simulation of the proposed approach. The ground and air-to-Ground network testing and simulation taking the core components of the communication model into account has been performed in this chapter. In addition, the UAV smooth path designs taking the UAV propulsion energy factor into account have been tested and analysed, followed by a real-world validation of the proposed paths. This aims at providing a high degree of flexibility with effective real-time validation mechanism. A proposed co-simulation between Contiki-Cooja, MATLAB, CupCarbon and SITL Mission Planner is presented in this chapter. The concept is tested to be aligned with the defined parameters of a generic use case including data gathering from trap sensor nodes.

Chapter 6 presents a physical network organisation for the UAV-based data collection effort to demonstrate the capability of ground node re-configurability. Additionally, a large-scale network testbed of SensorTag CC1352R LaunchPad, development board and Raspberry Pi modules considering various densities of the network is tested and evaluated to compare the network performance of real-world scenarios with simulation testing.

Chapter 7 concludes the research work, along with the future work.

1.7 Research Outcomes- Publications Based on the Thesis Work

Three papers have been published over the course of this PhD research work as listed below:

1. Karegar, P. A., & Al-Anbuky, A. (2022). "UAV-assisted data gathering from a sparse wireless sensor adaptive network". Wireless Networks, 1-18.

2. Karegar, P. A., & Al-Anbuky, A. (2022, October). “UAV as a Data Ferry for a Sparse Adaptive WSN”. In 2022 27th Asia Pacific Conference on Communications (APCC) (pp. 284-289). IEEE.
3. Karegar, P. A., & Al-Anbuky, A. (2021, July). “Travel Path Planning for UAV as a Data Collector for a Sparse WSN”. In 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS) (pp. 359-366). IEEE.

1.8 Conclusion

Development of a UAV-enabled data gathering over software-defined wireless sensor network organization that allows for smoothness and flexibility for the UAV path design has been deemed as a viable solution in this chapter. This chapter has presented a general overview of the research topic as well as typical data gathering approaches and challenges to justify the use of the UAV as a data ferry to ensure reliable communication between the ground network entities and the UAV. It then delved into the ground network structure and the challenges that the network faces as a result of UAV dynamics. Additionally, the challenges within the proposed dynamic ground re-orchestration structure and subsequent UAV path design to provide the complete data gathering architecture were identified and proposed. In this chapter, the motivation for driving towards the formulation of a flexible ground network structure capable of adapting to the UAV path design and re-orchestrated network demands utilising softwarisation concept was highlighted. The research question was then laid out. The latter stages of the chapter were devoted to brief discussions on the specified objectives. The research output, in the form of publications, was then listed, followed by the chapter conclusion.

Chapter 2 Literature Review

2.1 Introduction

With the development of Wireless Sensor Network (WSN) system architecture, permanent monitoring and processing of the measured data collected from the distributed monitoring nodes impose significant challenges, including stable connectivity, energy efficiency within the network set-up, timely data delivery and radio pollution in the field. These necessities new requirements for a robust and efficient data collection infrastructure following the most recent system design approaches of combining ground Sensor Nodes (SNs) with other technologies such as Unmanned Aerial Vehicles (UAVs).

Utilizing the drone as a regular data ferry that collects data from the SNs on the ground provides a reliable solution since it facilitates Line of Sight (LoS) upward communication on WSN-UAV connectivity. This approach can also improve network performance including the packet delivery performance and energy usage on a per device by reducing the need to horizontal communication with the sink and replace that with individual nodes or local group of nodes vertically communicating with opportunistically accessible UAV. Using the UAV as a mobile data agent also targets reduction of radio pollution over a large sensitive area like bird habitat or natural reserve. Given the highly dynamic nature of UAV-Ground link connectivity, data gathering by UAV necessitates physically communication management on the UAV-Ground link that must provide efficient and stable routes for the UAV via a reliable data gathering approach from the ground network. Since the air-to-Ground communication scheduling is tightly coupled with the UAV's trajectory, a relaxation of the UAV trajectory is required for efficient data collection. This could also facilitate modifying the common functionality of the ground network to allow for a more efficient network structure.

This chapter offers a study and analysis of the existing state of the art related to the UAV-enabled WSN data collection research elements like UAV path designs, UAV-Ground communication, and Software Defined Wireless Sensor Network (SDWSN) where the ground network re-orchestration can be investigated. Specific focus is laid on the aspects of network structure in-line with the UAV path design wherein the UAV movement restrictions and air-to-Ground network challenges are integrated within the approach. Based on the analysis, the novelty of the proposed UAV-based WSN data gathering in

contributing to the improvement of the state-of-the-art is highlighted including identification of the key conceptual and operational elements.

2.2 UAV-based WSN Data Collection for Conservation Monitoring Use Case

Practical development and implementation allow validation of the proposed theoretical approach in scalable fashion with physical ground nodes. As the use-case of this thesis is focused on the UAV-based data gathering modelling from distributed traps on the ground, the most recent literature in this regard needs to be reviewed in order to observe the parameters involved in data acquisition from real distributed traps on the ground. One parameter involved in data collections from traps is the distribution of sensor nodes in the conservation. In this regard, the area of woody vegetation is divided into four parts based on [12] (Claverley, North Taupo, Leader Valley, Purakaunui). In these four areas, the number of effective traps density equals to 2~4 traps per hectare. According to the New Zealand Department of Conservation [13] spacing between traps depends on the species that are targeted, and it estimates the distance between traps for rat, stoat, possum in a trap line. According to [13] the spacing between possum traps is 20~40 m initially, or up to 100 m at low possum densities. The work in [14] has utilised 108 sites for brushtail possum monitoring in an attempt to find out the possum occupancy rates in wildlife and the traps transect is recognized in distance of 200 m from each other. As a result of evaluating these literatures, the distance between possum traps is required to be determined. The field level data is then aggregated at the node level by trap sensors before being relayed to the central gateway via the UAV systems. According to Faial et al. [15], the single-board Raspberry Pi computer is used in the embedded hardware as gateways to gather and process the data coming from each sensor. In paper [16] the researchers employed TelosB ultra low-power nodes (see Figure 2-1) as ground sensor nodes distributed in the field to improve pesticide spraying accuracy while lowering the risk of human exposure to these products. A precise and scalable data collection scheme is designed to enable long-distance communications at low bit rates in order to reduce energy consumption. In [17, 18], the authors used the data gathering implementation using Arduino boards for both WSN and UAV and also they used Raspberry Pi at base station to collect the environmental data.

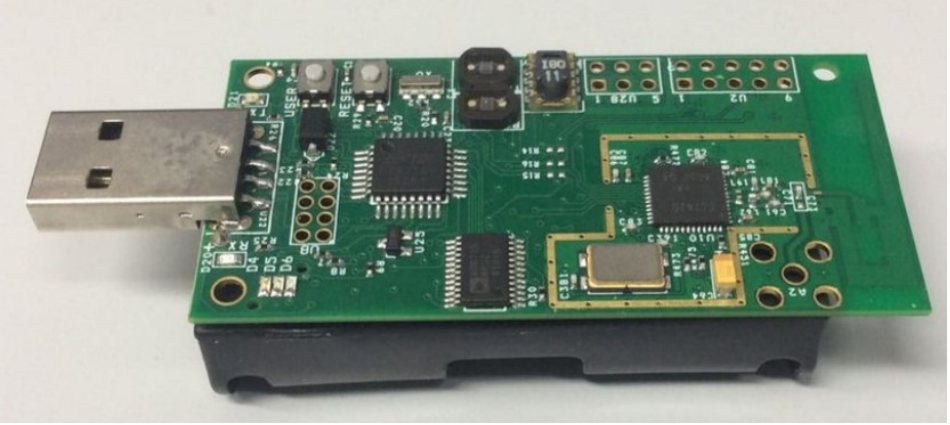


Figure 2-1 TelosB WSN node [16].

Other factor involved in UAV-aided data collection from traps use case is the choice of UAV type which plays a crucial role in the efficiency of data gathering approach. According to the literature [19, 20], in many small to medium scale applications (e.g. precision agriculture, sensing weather and soil conditions, pest and weeds management, animal attacks and crop morphology), the UAVs which are employed in data gathering are quad (or multi) rotor UAVs. Large scale applications (e.g., pesticide spraying [21], crop monitoring, etc.) are mostly handled by fixed-wing UAVs. The former can track complex trajectories (where both position and yaw angle are controllable outputs), whereas the later are usually restricted to a constant-height flight plane (where the velocity and heading angle are controllable). Overall, fixed-wing UAVs have a higher cruising speed and a longer operation time whereas multi-rotor UAVs are more flexible and require less maintenance but have lower autonomy.

Although considerable works have been done for the development of UAV-enabled physical WSN data gathering schemes, the aspects that affect the system performance viz., UAV's energy in relation to the ground communication cost have been given less attention.

2.3 An Overview of Various UAV-enabled WSN Communication Methods

Wireless communications via flying unmanned vehicles such as drones has recently gained popularity due to its numerous advantages, including on-demand and swift deployment, ease of application with entirely-controllable mobility, and high probability of having line-of-sight (LoS) communication links with the ground SNs [22]. There are three various use cases for UAVs-aided wireless communications including UAV-

enabled mobile relaying [23], UAV-enabled base station [24], UAV-enabled data acquisition [9]. One special application for the UAVs is through the use of that as a mobile relay. Zeng et al. [25] have proposed UAV-enabled multicasting systems in which a UAV is used to disseminate a common file to a set of ground terminals. The main goal of the research is to minimize the UAV mission completion time, while ensuring that each ground terminal can successfully recover the file with a targeting probability. In their study, a set of optimal waypoints for the UAV trajectory is found, and then the instantaneous UAV speed is optimised along with the path connecting these waypoints. The UAV also serves as a flying base station in UAV-based aerial base stations use case, offering a reliable communication particularly in emergency situations to the ground users [26].

Another use case of UAV-based wireless communication is reflected in data acquisition leveraging the UAVs from the ground sensor networks. The main issues in creating a scalable, energy-efficient and delay tolerant UAV-capable WSN data gathering model are the high mobility, frequent movements of the UAV over the ground network and disruptions in the communication network [27].

2.3.1 UAV-assisted WSN Data Gathering Path Planning Approaches

Numerous studies have focused on data gathering application utilising the drone from a distributed wireless network where the drone approaches each data point individually. To serve the most SNs within an operational connectivity during a limited flying period, the shortest route should be identified. The UAV trajectory designing domain is suggested based on a wide variety of ever-changing methods such as the geometric based path planning and heuristically trajectory planning. Herein, selecting the proper UAV movement pattern may lead to a more scalable and energy-efficient flying route design. One method is to impose geometric constraints on the UAV path, such as circular, spiral, strip-based and zig-zag ones to heuristically improve UAV energy usage pattern [27, 28]. To this end, Wu et al. [27] assess the plain circular route considered as an analytical procedure with low design complexity. The velocity of the UAV is set on a fixed value and the UAV path is defined as a circular one with the same radius. While the circular path is an approach with low complexity computation, the area of interest cannot be effectively covered by the drone. As an alternative approach of employing geometric path to tackle the UAV routing challenges, Yue et al. [29] have arranged the data points in concentric shells and mapped a spiral pattern passing among them. Strip-based and zig-

zag patterns are outlined in [30] having similar geometry, but with the difference in that rectangular strips are replaced with zig-zag lines. Additionally, the authors in [31] evaluate impact of UAV mobility patterns on the quality of the data collection from ground nodes. The UAV flight paths at different altitude levels are examined over the area of interest to discover and enable the sensor network. Four mobility patterns are subsequently defined as: tractor, angular, square and circular patterns (shown in Figure 2-2). In this study, the metrics for model evaluation include the UAV's overall coverage and mission time efficiency. However, shapes of the suggested UAV paths, as shown in Figure 2-2, have a negative effect on uncovered nodes and leave coverage gaps between data points.

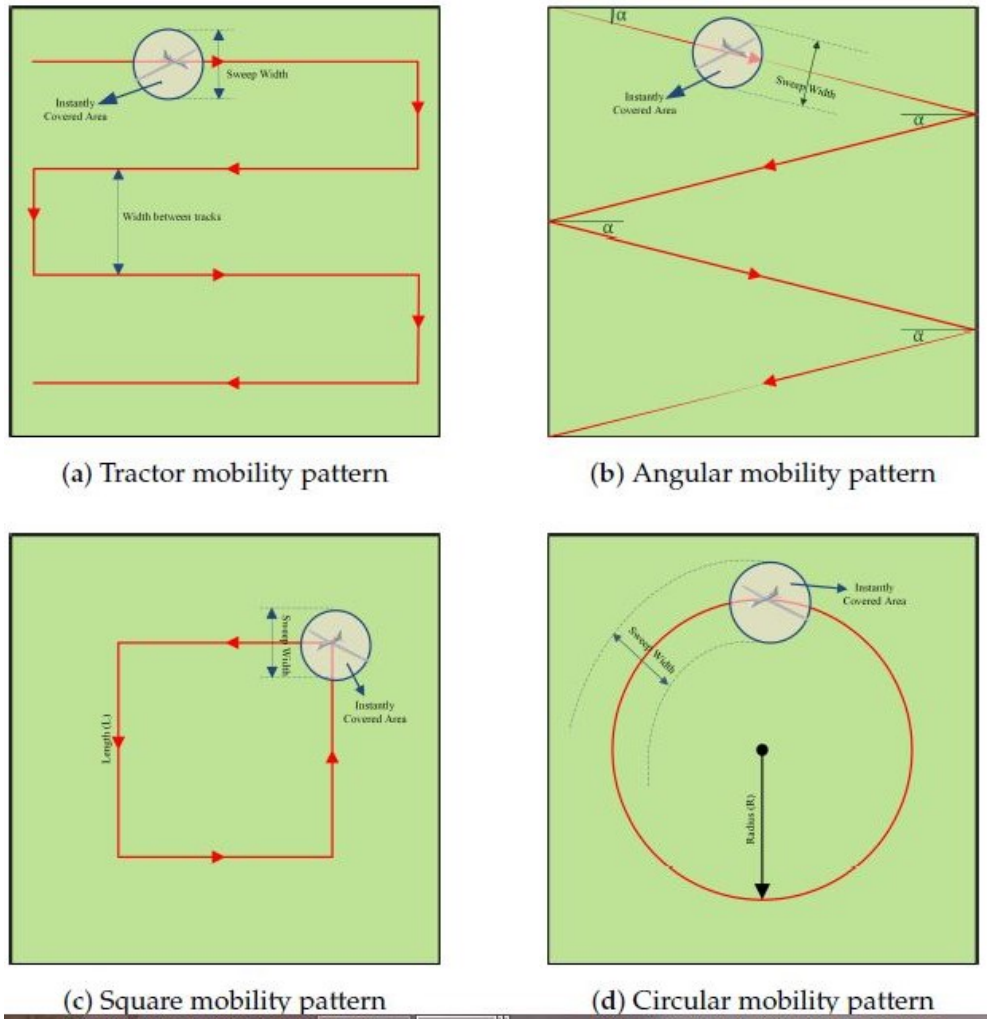


Figure 2-2 Mobility patterns which can be used in data acquisition process [30].

There are other protocols and algorithms which are pointed out to solve the UAV path problem heuristically such as iterative genetic algorithms (GA) [32], ant colony optimization (ACO) [33]. In the literature [34, 35], the authors use Traveling Salesman

Problem (TSP) solution and Pick up-and Delivery Problem (PDP) solutions for the UAV trajectory design design as preliminary routing for the UAVs, respectively. The authors of [36, 37] also discuss the issue of optimal wireless sensor network coverage via UAV platforms in terms of an optimization problem. This is formulated by means of heuristic approach of travelling salesman problem (TSP) to find the optimal routing of the UAV for data collection while minimizing the energy needed for data transmission of the SNs. It is proved that the proposed TSP algorithm outperforms particle swarm optimization (PSO) for various flying heights, speeds and network sizes [36]. When employing the TSP algorithm, the results for the tested configuration in this paper show less than half total energy consumption at the gateway level. TSP solution is also used in [34] to improve connectivity between the UAV and nearby nodes compared to the current geometric UAV route planning. Chao et al. [38] have suggested data gathering form virtual clusters considered traversing from one cluster to another by designing waypoints selections using $K-Means$ clustering algorithm to reduce the latency of mobile data gathering in a WSN. However, in all of the preceding literatures, the UAV service area during its travel is regarded as concentration points between sensors, lowering the UAV's energy efficiency in data acquisition. Also, Trajectories like zig-zag or rectangular ones, in particular, introduce sharp and sudden changes in the UAV's speed and acceleration in the concentration points. To this end, another route planning solutions for traversing over curves is through the use of curves such as so-called Bezier curves [39-41]. A smooth curve connects the start and end points in this case, whereas the flight waypoints serve as control points to navigate the curve. [40]. Since the Bezier curve path essentially does not pass through all given waypoints, this approach can be suitable for addressing WSN data collection from a distance. As shown in the benchmark comparison in Section 5.4.1, the Bezier curve TSP (BTSP) requires more energy to pass through the curves which lowers the UAV performance in terms of energy efficiency per mission. As a result, the UAV energy efficiency is a critical performance metric in UAV-enabled WSN data gathering [42]. The limited battery capacity makes the UAV to land for recharging, which severely restricts the flight time of UAV. Note that, unlike conventional terrestrial systems, the UAV needs to consume propulsion power to remain aloft in addition to the power required for the communications [43]. Therefore, having an energy-efficient trajectory design for UAV communication systems is of paramount importance. In this research, an energy-aware UAV trajectory design is suggested regarding the air-to-Ground connectivity taking UAV's energy efficiency into account.

Other researchers, on the other hand, have proposed UAV trajectory optimisation solutions to overcome mobility constraints on the UAV movement. This includes the optimisation of the UAV's trajectory parameters such as altitude, velocity, and energy usage of the UAV while ensuring the reliable communication network. In this regard, Li et al. [35] have proposed a UAV-enabled data collection design to gather data from numerous ground entities in which the UAV's trajectory, altitude, velocity, and data links with ground networks are optimised to minimize the total mission time. The problem is decomposed into three optimization subproblems, altitude, velocity, and link scheduling optimisation problems. The simulation results show that the proposed UAV time minimization design achieves significant performance gains over existing algorithms. The authors in [44, 45], utilise a communication-oriented UAV trajectory design to gather data from subsets of the sensors as cluster heads to minimise the total energy consumption of the UAV tour for both travel as well as hovering for data collection regarding the communication requirements amongst the SNs and the UAV. The authors of [33] assume a limited buffer capacity for the given SNs and solve the path generation problem concurrently using multi-hop routeing within the WSN. The authors propose a new approach in which nodes send their data via a multi-hop path of reduced length by allowing nodes to buffer data while waiting for the UAV to approach, thereby exempting more nodes from relaying these data. This strategy allows for energy savings while also ensuring that no data is lost due to buffer overflow. Furthermore, this solution has a longer lifetime and allows for more data to be generated and transmitted to the mobile sink. It is also obvious from this study that the proposed scheme better balances the load among nodes. Zeng et al. [42] have suggested a theoretical model for energy efficiency of a fixed wing UAV that relates its propulsion energy consumption with the flying velocity and acceleration. The aim of the paper is to propose an efficient design for enhancing the UAV's energy efficiency considering general constraints on the UAV movement ensuring maximal communication bit rate. The model is the first theoretical model that relates the UAV's energy consumption with its velocity and acceleration, whereas other literatures [46, 47] mostly uses heuristic energy consumption models only considering the speed parameter. Numerical results have shown that the proposed designs in [42] achieve significantly higher energy efficiency for UAV path as compared with conventional methods [46], [47]. However, none of these studies take the energy aspect of the ground network entities at the same time as UAV energy consumption into

consideration. Integrating the energy efficiency of the ground network with the air-to-Ground scheduling and UAV path planning can offer more realistic outcomes.

On the other hand, Zhan et al. in [9] have focused on the energy efficiency of the ground SNs by developing an optimization problem for SNs' wake-up schedule and UAV's trajectory to minimize the energy consumption of all SNs while ensuring that a target amount of data is forwarded from each SN to the UAV. The proposed scheme achieved significant energy savings for the SNs as compared with static data collector. Ebrahimi et al. [48] have also outlined the issue of energy-efficient data collection in dense WSNs using minimising the length of UAV flight path and SNs' transmission power. Since flying the UAV over all nodes in the network to collect data is not an efficient way of data gathering since it makes long travel trajectory, high data collection delay and frequently recharging period of the UAV's battery. One solution is that the data which are generated by the sensors be compressed in some transform domain, recovered at the sink by using the help of a UAV. Hence the authors in [48] have targeted this issue using compressive data gathering (CDG) approach. This means that a deployed UAV has followed an optimized trajectory over an area of interest to collect aggregated data from intelligently selected sensor nodes called cluster heads to be delivered to a central sink node. The key differentiating novelty of this paper was on the joint modelling of compressive data gathering (CDG) and UAV trajectory. Authors in [49] discusses UAV flight time optimisation in data collection missions that support ground sensor networks while the UAV is cruising or hovering. The aim is to achieve optimal, non-overlapping data collection intervals, UAV speed and sensor transmit power by using a dynamic programming approach. The main finding highlights how the UAV speed should be proportional to the energy levels of the sensors and the inter-sensor distance. While cruising, it is found that even if the data rate of the nodes decreases, the reduction in the mission time is able to compensate for this in comparison to hovering. However, the authors in these papers focus on the energy efficiency of data gathering in WSNs and minimising the length of UAV path or mission time than conducting a detailed analysis of UAV path energy efficiency while considering the propulsion energy required for drone movement. Table 2-1 lists some research papers that use optimisation to solve the path planning design problem. A comprehensive model that considers the UAV and SN's energy efficiency and secures the maximal communication packet delivery rate either in-ground or air-to-Ground communications is missing in all above literatures. Moreover,

all such heuristic methods share similar characteristics and flaws. They may become stuck into local minima despite providing quick and near-optimal solutions.

Another aspect of UAV energy efficiency in UAV-enabled wireless communication is through taking the minimum number of UAV mission operations required by the application into account. In some practical use cases, one-time operation UAV fly is suggested in the literatures since UAV needs to collect the data from the SNs with only one single fly and then it can leave for another mission. Alternatively, Zhang et al. [34] have studied UAV-enabled radio access network considering periodic operation and have compared that with one-time operation scenarios to minimize the UAV flight time. In periodic operation, UAV serves the ground SNs in a periodic manner by following a certain trajectory repeatedly, while in one-time operation, the UAV serves the ground SNs with one single fly for an intermittent service request. Furthermore, for both operation scenarios, the UAV's initial trajectory is designed using the Traveling Salesman Problem (TSP). The numerical results show that the proposed TSP trajectory designs save significant UAV flight time and improve user throughput when compared to the conventional circular initial trajectory for both one-time and periodic operation methods. Proposing the periodic data gathering operation enabling dynamic ground network reorchestration once per multiple UAV missions is a novel technique that can be viewed state-of-the-art in this context.

Since most existing works assume that the UAV flies at a fixed altitude and thus only the two-dimensional (2D) UAV trajectory is considered. In studies [50, 51], authors attempt to design the optimized trajectory in UAV-enabled WSNs which exploits the vertical trajectory of the UAV and presents a new design framework of three-dimensional (3D) UAV trajectory to further improve the rate performance in UAV enabled WSNs. As the UAV at a sufficiently high altitude has a high likelihood to establish a LoS link with the ground node, the deterministic LoS channel based on the free space pathloss model has been widely used in most of the existing works on the UAV trajectory design due to its convenience for optimization. Considering shadowing, signal propagation can be obstructed by obstacles (e.g., trees) in the environment. Hence, the UAV-ground channels can be largely categorized into either LoS or non-LoS (NLoS) link at different locations with varying features. To overcome the impact of NLoS links, the researchers in [50, 51] have designed the 3D UAV trajectory in the angle dependent Rician fading channel.

However, taking the suggested fading model into account, the proposed approach is not always realistic.

To gain the most essential insights, the current UAV flying data collection methods have not accounted for the dynamic topological formation and the flexibility of a network architecture to be adjusted based on the UAV flying route. The UAV data collection approach can be improved via the analysis of the UAV route planning and the integration with softwarization. Herein, one approach to lower the UAV energy consumption per mission is to design a framework to constrain the UAV velocity by employing the topological sorting concept and boosting the UAV path shape to enhance the UAV energy efficiency as well as the fairness of data communication on air-to-Ground connectivity. The idea of fuzzy path technique is studied in [52-54]. In these papers, the proposed concept considers spatial grouping structure with one or more nodes that can act as a ground data capturing representative. Nomination of the active collection nodes are based on the proposed fitness model. This results in a more efficient and relaxed UAV flight path design. Hence, the performance of the approach is critical to be assessed based on the UAV energy usage while traveling the visiting points. The UAV relaxed path should be mapped based on the proposed network communication analysis offering a higher percentage of served sensor nodes. Furthermore, while all the preceding literatures focused on data gathering from uniformly distributed SNs, one of the other recent challenges is UAV-enabled data gathering using non-uniformly distributed SNs. To overcome this issue, this thesis considers the topic of developing the network into groups of SNs with topology organisation facilitating the relaxed path for UAV through a technique so-called fuzzy path.

Table 2-1 A Range of Related Works.

Parameters in Optimisation	**Methods**	**Application**	**Miscellaneous**	**Reference**
Path Length	GA	Crop Monitoring	Experimental Results	[55]
WSN-based path update, communication radius, energy consumption	TSP	Pesticide spraying, crop monitoring	multi-pass, grid sensor deployment, experimental results	[56-58]
Communication radius, energy consumption	Non-Linear Programming, TSP	data gathering	forward and axial flight	[28, 59]
path length, energy Consumption	greedy algorithm	data gathering	Experimental results	[60]
path length, communication range, energy consumption	heuristic methods, greedy algorithm	data gathering	Spiral, zigzag, strip-based paths	[29, 30]
path length, energy consumption	heuristic algorithms, TSP	target tracking, area coverage	Experimental Results	[36]
Path length, sensor lifetime	linear programming	data gathering	Limited sensor buffer capacity	[61]
path length, travel time	heuristic methods	sensor node localization, data gathering	Multi-pass, experimental results, zig-zag path, sensor deployment	[16]
path length, energy consumption, communication throughput	Mixed integer non-convex programming	data gathering	Multiple ground terminals	[45]
communication radius, path length, GW selection	Non-linear programming, heuristic methods	precision agriculture	Spline parametrization	[62]
path length, area coverage	GA, PSO	pesticide spraying	Experimental Results	[15]
energy consumption	Non-linear programming, GA	data gathering	TDMA, cyclic path	[27]
Multi-objective utility function	GA algorithm	data gathering over large areas	heterogeneous IoT devices	[63]
path time, energy consumption	TSP	data gathering, recharging of depleted IoT devices	IoT devices	[35]

2.3.2 UAV Path Design Real-World Validation

There are numerous publications available discussing potential solutions for real-world validation of proposed designs utilising real-time simulators. For instance, the authors in [64] simulate and test their proposed method for facilitating pattern formation among UAVs via a virtual IP based Software In The Loop (V-SITL) environment. The proposed V-SITL environment offers scalability on account of the entire UAV system being simulated in a single computer. The authors in [65] use Dronekit SITL for real-time simulation of multiple UAVs performing multiple tasks cooperatively and assigning specific tasks to each vehicle without the need for the real world implementation. In our previous work [53], the SITL Mission Planner is used to provide testing of the proposed fuzzy path, including the crisp path and Bezier Curve TSP path, as discussed in Section 5.4.2, before arming the vehicle and running the mission in real world.

2.4 Wireless Communication among Ground Sensor Nodes

Due to the considerable flight height of the UAVs, direct connection of UAV with all nodes is not energy-efficient way of data gathering. This problem can be solved by dividing the nodes into clusters so that only the cluster heads (CH) can communicate with the drone. The clustering process involves two steps: selecting the CHs and constructing the cluster. According to the literatures, building the state-of-the-art topology and data transmission routes in clusters and devising an efficient CH selection algorithm are crucial elements of the research in this topic. As discussed in [66-69], the selection criteria for the CH election process includes the remaining energy of the nodes, the position density of the nodes, and their distance from the drone's path. Furthermore, the clustering model that are chosen are based on improving different performance merits such as increasing cluster size, enhancing WSN functioning time and energy usage, minimizing data collection time, and reducing the latency in network-based clustering to deliver real time data with less delay to the drone.

2.4.1 Energy Efficiency Factor for Ground Sensor Network

In a typical ground wireless sensor network data gathering effort, due to data traffic concentrating towards the sink, the batteries of sensors near the sink deplete faster than those of other sensors. To address this issue, the authors of [67, 68] employ the mobile sinks in the network. However, considering a mobile data ferry to gather the data may arise additional challenges in the mobile data gathering with wireless sensor networks

due to the frequent link changes or failures which make the network topology unstable; hence, Karunanithy Kalaivanan et al. [67] have proposed a mobile clustering data collection protocol which produces a highly stable and reliable routing path for data transmission. In this paper, a fuzzy logic-based CHs selection is utilised considering the node speed, number of neighbour nodes and average connection time. It is observed that this algorithm enhanced the packet delivery ratio, throughput, energy consumption, end-to-end delay. In this regard, Can Tunca et al. [69] have also outlined a novel, distributed and energy-efficient mobile sink routing protocol called Ring Routing, which is beneficial for time-sensitive applications. It is worth emphasizing that this routing protocol can be applied on large scale WSNs deployed outdoors with stationary sensor nodes and a mobile sink which could establish a virtual ring structure that allow the data to be easily delivered to the ring. The outcomes indicated that Ring Routing has offered an energy-efficient protocol which extends the network lifetime. However, in all of the preceding research, they have used mobile nodes as sinks, limiting the mobile sink's ability to maintain a stable link with CHs due to obstacles in the ground.

As employing the UAV as a data ferry assisting the ground network data collection, the authors of [70] focus on the clustering methods for the ground SNs in a single UAV scenario for environmental monitoring and data gathering. The energy model for the SNs is composed of the energy required for packet reporting and packet forwarding in the case of mesh networks where each node can serve as a router for neighbour nodes towards the sink. UAV path planning uses a heuristic path, which is a local search algorithm to solve TSP for visiting the WSN clusters. The metrics involved in the optimisation problem are the UAV and sensor energy consumption versus the maximum number of network hops. Tarighi et al. [71] and Dan Popescu [37] have also investigated UAV as a mobile agent to collect data from SNs while taking into account the effects of clustering parameters on WSNs, implying that their methods are useful for environmental monitoring applications. The results indicate that the collaborative UAV assisted WSN method enhances the performances in both precision agriculture and ecological agriculture use cases. In another study as discussed in section 2.3.1, the UAV platforms act as data mules between the cluster heads of the network in the UAV-enabled WSN data aggregation expressed in [27]. The routing optimization is based on genetic algorithms (GA). The energy consumption of data transmission is minimised using a standard energy-per-bit transmission model. The evaluation is based on three other protocols: The genetic algorithm of centre-based,

greedy-based, and clustering-based, and the proposed approach improves system-wide energy consumption compare with other benchmarks.

The other challenge of the topic of WSN data gathering using the UAV as a mobile sink is to drive a balanced trade-off between the length of the UAV path and the energy consumption of the ground SNs. As discussed in section 2.3.1, Ebrahimi et al. [48] have outlined a forwarding tree structure and CH selection using a novel projection-based method in order to gather the data from the SNs and transferring them to the CHs which are responsible for gathering the data from all other nodes in the same cluster and sending the aggregated data to the UAV as a mobile sink to be delivered to the base station. The UAV then passes through all the selected cluster heads once and arrives at the destination with minimized total trajectory distance. The clustering problem which is used in the research is based on an updated version of $K-means$ clustering [72]. In their proposed algorithm, after applying $K-means$ clustering method, the clusters are updated to be within almost equal size cells. The simulation results have proven that by the UAV assistance and gathering data at CHs instead of the sink not only decreases the number of relaying transmissions, but also the needed energy consumption is reduced notably. However, the approach used in this paper is not based on a realistic scenario for the ground network formation.

To investigate and test the role of clustering on energy efficiency of the network, Alagirisamy et al. [66] have utilised unequal cluster sizes in their WSN's design to reduce the extra energies which are consumed by the CHs once the data is routed to the sink. The simulation indicates that considering unequal cluster sizes in WSN design enhances network lifetime of SNs significantly. In many application scenarios, relaying the raw collected data from the local sensors to the central aggregation point of the integrated system poses a considerable burden on the computing and communication resources. In these cases, some form of processing is necessary to compress, reduce or intelligently extract information from sensor measurements in order to improve energy efficiency of each SN. The analysed articles propose a wide range of methods ranging from basic processing methods (min/max thresholding, averaging and basic statistical features) to higher level distributed approaches such as fog computing. For instance, authors of [73] point out the key areas of active research for data aggregation within IoT systems of which WSNs are a core building block. The potential for improving the energy efficiency of the network through clustering mechanisms for data aggregation is acknowledged through

tree-based and centralized approaches. The advantages and disadvantages of each method are identified in terms of energy, traffic load, accuracy, security, scalability, and fault tolerance. There are many literatures in which Cloud-based support infrastructure for UAV enabled WSN mixed data collection platforms. As discussed in [60], increased availability of cloud resources contributes to the improvement in the flying parameters and information extraction from WSN data. Distributed schemes, on the other hand, such as consensus algorithms, can contribute to intelligent data reduction and reduce the burden on upper UAV layers by passing processing burden to the ground sensor level [74]. This method is appropriate for use cases such as event detection where the integrated system must ensure the timely observation of mission-specific events. In this manner, each of the ground sensors communicates its findings with neighbouring nodes on a regular basis in order to reach a joint decision, such as the event present, in as few communication instances as possible. Local analysis can also help to reduce overall detection latency. As the primary focus of this research is on energy efficiency of whole system, utilising local analysis may reduce the battery level of ground SNs, hence, in this thesis, we assume that the required computation analysis, such as fitness model computations, should be executed in cloud to benefit from an energy aware system model.

To sum up, considering the issue of data gathering from multi points reflects the idea of using dynamic clustering methods in which only the updated CHs communicate with the UAV. Hence, designing an innovative dynamic network clustering architecture enabling the network re-orchestration regarding UAV path and proposing an energy efficient data communication approach among ground SNs are some of the research components in the UAV-aided WSN data collection topic.

2.4.2 Dynamic Ground Network Re-orchestration and Virtualisation

To examine the communication link prior to the real-world implementation testing, there are numerous methods to virtualise the network behaviour within the cloud. The efficient utilization of WSN deployments as multiple applications will be feasible to co-exist on the same virtualized WSN [75]. Many researchers recently employ WSN virtualization as a key enabling technology in their WSN deployments [76-81]. Acharyya et al. [77, 82] propose the WSN virtualization unit which utilises Cooja network simulator running the virtual sensor cloud model that supports software definition of the network. The virtualisation approach for the vehicular network application is also highlighted in [78-81] in which the authors offer a modern concept that WSN functions can be controlled

via the remote cloud. Virtualization approach here is used as an approach for testing possible re-orchestration scenarios of the vehicular network organization. This includes separating the vehicular node's functions so that they can be loaded as software functional components in the remote cloud. The method enables the re-organisation and integration of the vehicular network, allowing for dynamic re-orchestration. For example, by implementing any of the defined functions, a node's functional role could be switched via software re-formulations at the virtual level. As a result, the operational behaviour of vehicular network can be modified to meet service requirements. Furthermore, Samir et al. [83] have employed a novel virtual design for UAV-aided vehicular networks to collect and process fresh information prior to forwarding that to the UAV.

To the best of our knowledge, the current UAV flying data capturing methods have not considered the dynamic topological formation and the flexibility of a virtual network structure to be aligned on the UAV flight path. The UAV data collection approach can be improved via the analysis of the UAV travel route and its integration with softwarization and network virtualisation. Herein, one solution to lower the UAV energy consumption per travel is through designing of a framework of UAV path range by employing the concept of topological sorting and boosting the UAV path shape in order to improve the UAV energy efficiency and data communication fairness. In [52-54], the concept of fuzzy route is studied in which the proposed approach takes geographical grouping structure with one or multiple nodes that can act as ground data collection representatives into account. This leads to a more efficient and smooth UAV travel planning. Herein, the performance of the scheme is evaluated in terms of the UAV energy consumption while traversing the visiting points. The proposed network communication analysis along with the UAV smooth flight path reflect a higher percentage of served SNs and improved ground SN power usage.

2.5 Air-to-Ground Communication on UAV-Ground Connectivity

From the connectivity perspective on the air-to-Ground network communication, in some regions including heavy vegetation, the communication link may be weakened or lost due to signal attenuation which imposes additional restrictions on the UAV movement. Hence, in such dynamic aerial network, both the high communication latency and packet loss between SNs and the UAV impact the data collection effort deeply. Some critical parameters in designing an efficient UAV-Ground link are data packet size, frequency of UAV mission operations and the aggregated data packet size over the entire trip. In this

regard, Karunanithy et al. [84] have proposed intelligent data collection by UAV for water irrigation use case in which a number of SNs are distributed randomly in a field with the data packet size of 512 bytes per node communicating with the UAV above. To overcome signal attenuation, they propose an air-to-Ground communication diagram based on broadcasting a beacon signal to the affected SN once the UAV reaches the SN's place. The affected SN then sends the acknowledgement to the UAV prior to initiating to send the collected data to the UAV. Park et al. in [85] have proposed a UAV-enabled monitoring system in order to supervise an agricultural environment. The defined communication between SNs and the UAV is conducted via client/server connection in which supports data rates between 0.3kbps and 50kbps. Rao et al. in [86] have also designed and implemented a UAV-assisted smart agricultural management platform with three layers: perception layer, network layer and application layer. During the implementation phase, four SNs are distributed with two SNs equipped with humidity sensor and two SNs equipped with temperature sensor. According to the proposed strategy, once the UAV reaches each land, it hovers for 20 seconds to ensure that the SN has successfully accessed the UAV network channel and collected the data stored in the SN. To guarantee that the relevant SNs' data packets are properly offloaded to the UAV in the UAV-Ground communication, Samir et al. [17] has suggested a data collection algorithm from time constrained IoT devices by optimising the trajectory of the UAV utilising successive convex approximation (SCA) algorithm. However, with raising the number of dispersed nodes in the field, the complexity of the system will be exceedingly high that leads to degrading the data capturing fairness among the sensor nodes.

Almost all the surveyed papers consider a necessary time for data transmission between CHs and the UAV. To ensure that the UAV is within the CH communication range until all the information from that fixed agent is retrieved imposes additional restrictions. These may be implemented either explicitly by computing the time in which the UAV lies within the communication radius of the CH [30, 45] or implicitly by proposing a path guaranteed to spend enough time in the neighbourhood (e.g. hovering [20, 55]). Furthermore, simply passing through a waypoint or through its vicinity is almost never sufficient. Hence, some papers consider an explicit communication range when imposing hovering conditions [20, 55]. Most applications necessitate a period of hovering (such that there is enough time to gather the required information). In such cases, the difference between multi-rotor and fixed-wing UAVs becomes important, the former can hover easily and for an arbitrary amount of time, whereas the latter must enter a mode known as loitering, in which they

orbit around the current waypoint. However, requiring UAVs to hover may result in an additional time spent gathering data from CHs. For multiple CHs system, it is important to note how the UAV chooses to communicate with which CH in each time slot. Hence, the CH scheduling is another factor to improve the UAV's energy consumption [45]. The authors in this research paper propose a solution to the problem of the UAV's energy efficiency maximization considering CH scheduling which is solved in an optimization problem [45].

Employing communication window of time on the air-to-Ground connectivity is considered as a promising solution in UAV data gathering effort. Say et al. [87] have also employed a frame selection framework to transfer the data to the UAV. Based on the proposed data collection method, the SNs inside the UAV's coverage area are divided into different windows based on their locations with each window associated with different transmission priorities. Employing window of connectivity in air-to-Ground interaction are also studied in [88] and [89]. While the former assumed that contention window selection is based on receiving signal strength (RSS), the latter supposed that different transmission priorities are associated with different windows based on the location of the ground SNs. The concept of UAV-Ground communication interaction while the UAV has the minimal distance with SNs within the communication window is not viewed in the above papers yet.

2.6 Conclusion

This chapter presents a UAV-based WSN data collection approach as well as a literature review pertaining to the main components of the research, which include UAV path planning, ground network communication, and air-to-Ground connectivity. There is clearly a lack of research in the area of constructing an adaptive and flexible ground network using dynamic topological formation aligned with the UAV path design. Herein, the performance of the proposed method, particularly its energy and packet delivery rate, can be significantly affected by various network configurations.

Therefore, the research aims to create a solution for a UAV-aided WSN model that provides an energy efficient and scalable data gathering method and supports forest predator's traps application. The proposed model incorporates three parts, the UAV trajectory planning, UAV-Ground connectivity, and ground network communication. The proposed UAV path relaxation design is based on the so-called UAV fuzzy path, in which

the UAV velocity and acceleration are two parameters that are included in the trajectory planning problem. Additionally, within the ground network, the topic of developing the network into groups of SNs with topology re-arrangement capability is addressed. Then, with regard to air-to-Ground communication, designing a communication window of connectivity for the UAV that enables a virtualized communication mechanism with ground representatives is an important topic addressed in this thesis.

The performance of the scheme is crucial to be analysed in terms of the efficiency of the flight path in collecting data from all relevant nodes while accounting for data loss and communication costs. The UAV path could also be validated via the real-world simulation prior to arming the UAV for the actual mission. The ground network could be tested and analysed on the virtual machine to offer the best performance for physical network testing.

Chapter 3 Modelling and Implementation Tools

3.1 Introduction

The objective of this chapter is to provide insight into the research methods used for this work as well as the hardware and software tools employed towards construction of the proposed testbed. This involves tools relevant to real-world UAV path validation, physical ground sensor network, virtual ground sensor network, air-to-Ground communication network, and relevant analytical and operating system tools required for software/hardware development. The chapter justifies the significance of tools employed to test the experiments considering various aspects of the developed concept that is in-line with the research methods outlined in Chapter 4.

This chapter is inspired by three main objectives. First, the software tool required for validating and verifying the proposed UAV paths in real-world scenario is introduced. Second, two network simulators are detailed to provide a testbed for analysing air-to-Ground and ground communication networks in order to meet the data gathering performance requirements of a distributed sensor network. Finally, a physical WSN system is established by employing a hardware controller capable of provisioning the required processing and code storage memory to host the protocol for incrementally trailing the re-orchestration scenarios.

3.2 Related Work on Simulation of UAV-enabled WSN Data Gathering

The study of UAV-assisted WSN data collection, particularly in the domain of UAV energy efficiency and ground network performances, are mostly performed using simulators. Simulators are typically employed since they can simulate thousands of scenarios prior to testing them over actual hardware. Furthermore, the simulators facilitate to run complex simulations via high performance computers since the simulations only depend upon the machine's computational resources.

In this thesis, the simulator tools are mainly divided into two main categories:

- Communication network simulators including ground or air-to-Ground networks
- UAV path design simulators

3.2.1 Communication Network Simulators

Network simulation is the most useful and common methodology used to evaluate different network topologies without real world implementation. Network simulation is used in different areas, academic researchers, industrial development, to analyse, design, simulate and verify the performance of different network concepts and hypotheses. There are a number of network simulators for instance, NS-2, NS-3, OMNET++, OPNET, Contiki-Cooja and CupCarbon, etc.

Network Simulator-2 (NS-2) is an open-source discrete event network simulator designed for the simulation of network protocols enabling different network topologies. NS-2 is based on C++ for network topology designing and Tcl (Tool command language) to provide the script-based simulation interface [90]. NS-2 is the most common and widely used network simulator for research work particularly in the domain of UAV-aided WSN data gathering [91-93]. The performance parameter obtained are power consumption [91, 92], packet success rate [93]. However, NS-2 is not a good choice for more scalable networks including more than 100 nodes. One of the disadvantages of NS-2 network simulators is the lack of real-world implementation providing virtualisation functionality. Another drawback is that it lacks a native graphical editor for deployment scenario, which is critical for WSN study.

The NS-3 simulator is another open-source discrete-event network simulator for WSN systems, targeted mainly for research and educational use. It offers support to languages such as: C++ and Python. NS-3 becomes a more scalable and easier network simulator compared to the others and the function of NS-3 visualizer is more powerful than network animator of NS-2 simulator. However, not all of NS-2's modules are included in NS-3 and some of them need to be ported from NS-2 to NS-3. It is widely used in UAV-enabled data gathering WSN for measuring various parameters such as consumed energy, end-to-end latency, and packet delivery ratio [94].

OMNET++ (Optical Micro-Networks Plus Plus) is a component-based, modular and open-source, discrete event simulator framework written in C++. It has an extensive graphical user interface (GUI) and intelligence support. The most common use of OMNET++ is for simulation of computer networks, but it is also used for wireless network simulations. One drawback of this tool is that the mobility extension is relatively incomplete and missing a variety of protocols. OMNET++ has been extensively used in

data gathering from WSN by the UAV as Garraffa et al [95] has employed OMNET++ to simulate the received packet rate for a swarm of drones.

OPNET (Optimized Network Engineering Tool) is an event-based network-level simulation tool. It provides a powerful graphical environment supporting the modelling of network topology and distributed systems from application layer to physical one. OPNET simulator is also useful once working with complex networks with a high number of devices and traffic flows particularly for the network latency analysis. Luo et al. [96], have measured the time delay performance considering a proposed cloud-based multi-UAV based on OPNET simulator. However, OPNET has a drawback of limited number of nodes connected to a single device.

CupCarbon [97] is another open-source discrete-event network simulator employed to design, visualize, debug and validate distributed algorithms for environmental data collection. CupCarbon is most commonly used in mobility scenarios within the data gathering projects. It also allows for the testing of various wireless topologies, protocols, etc. Networks can be prototyped by an easy-to-use GUI using the OpenStreetMap (OSM) framework to deploy sensors directly on the map. It is based on a script called SenScript, which allows to program and configure each sensor node individually. The energy consumption can be analysed and displayed as a function of the simulated time. This allows for clarification of the structure, feasibility, and realistic implementation of a network prior to its actual deployment. Within our research, CupCarbon serves to visualise the network scenarios in the air-to-Ground connectivity to analyse the energy consumption and received packet rate performances.

Contiki-Cooja is a network simulator employed to create virtualisation concept that enable same programs to be used in both hardware and simulation environments. Contiki is an operating system (OS) developed particularly for the wireless microprocessors [98]. The Instant Contiki supports firmware development for a wide range of low power wireless hardware platforms such as 8051, MSP 430, AVR, ARM Cortex- M3, and z80. Contiki's in-built java-based tool of 'Cooja' [98, 99] provides virtualization at both the hardware as well as software levels. Contiki-Cooja also supports node mobility. By patching the mobility plugin within Contiki-Cooja, it enables controlling over the paths and speed of the virtual nodes. Using an editable 'positions.dat' file, the coordinates as well as the corresponding time-instants of a node or a group of nodes can be pre-defined with Contiki-Cooja to set the path of a particular mobile node. Within our research,

Contiki-Cooja is used to deploy the ground network scenarios and measure the energy consumption and packet delivery per sensor node.

3.2.2 UAV Path Design Simulators

To improve the UAV flight path energy efficiency, the main parameters that generate UAV flight path (including UAV velocity, acceleration, and power usage) can be characterised based on a real-world validation. There are a range of ground station flight simulators available to run flight simulations without any hardware. This facilitates the UAV path design enhancement by testing the behaviour of the UAV without running the real-world test.

Mission Planner [100] is an open-source full-featured ground station application for the ArduPilot powered vehicles. It is a ground control station for Plane, Copter and Rover and can be used as a configuration utility or a dynamic control supplement for an autonomous vehicle. It enables to setup, configure, and tune the vehicle for optimum performance and load the similar missions into the autopilot with waypoint entry on Google or other maps. ArduPilot Mission Planner includes a wide range of vehicle simulators built in and can interface to several external simulators. This allows ArduPilot to be tested on a wide variety of vehicle types. The SITL (software in the loop) simulator embedded in Mission Planner allows to run simulations without any hardware. It is a build of the autopilot code using an ordinary C++ compiler, generating a native executable that enables testing the behaviour of the code without hardware. It allows downloading and analysis of mission logs generated following a virtualised test. It also offers an appropriate telemetry visualisation that allows monitoring of the UAV's status while in operation. SITL Mission Planner is used in this study to specify the UAV crisp and curved paths and it allows us to evaluate real-world energy performance based on multiple real-time parameters such as current consumption and flight velocity that are determined based on the logfiles obtained from each mission planning.

QGroundControl [101] is another ground station application that provides full flight control and vehicle setup for ArduPilot and Pixhawk powered vehicles. It provides easy and straightforward usage for beginners, while still delivering high end feature support for experienced users. It also facilitates a software in the loop (SITL) simulation to do experiment on a wide variety of vehicle types without hardware. However, Mission

Planner SITL offers more parameters than QGroundControl once it comes to testing and simulating the Ardupilot powered vehicles.

3.3 Selected Tools

3.3.1 Air-to-Ground Network Model Implementation in CupCarbon

CupCarbon is the network simulation software used in this study to analyse communication interactions via air-to-Ground connectivity. CupCarbon is a well-suited tool for simulating data collection from distributed gateways via the UAV because it enables visually testing various WSN architectures through a discrete event simulation of the conducted testbed and allows the design of mobility scenarios by simulation of mobiles such as UAVs. CupCarbon supports a variety of consumption models, and it is also possible to write a personal model as any equation for the transmitter and receiver with predefined variables. To configure each sensor node, the coding is based on SenScript in CupCarbon which is close to Java coding as shown in Figure 3-1.

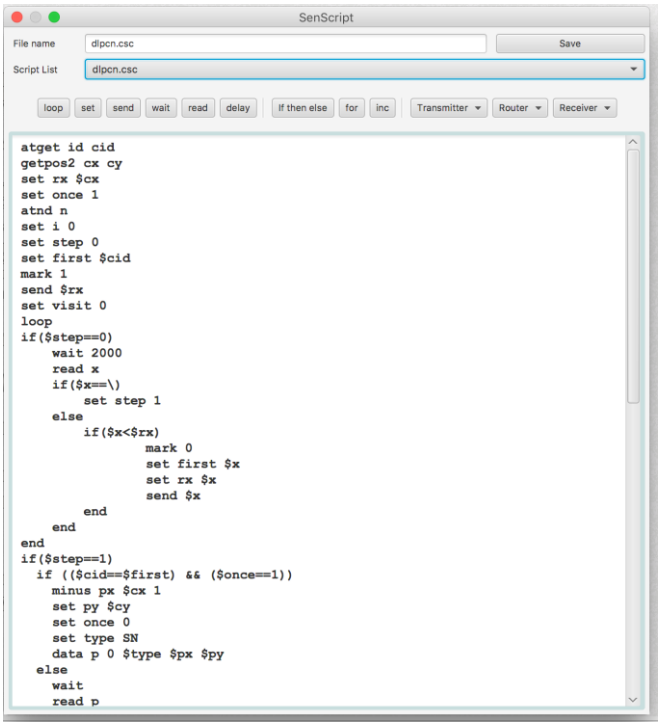


Figure 3-1 A sample of sensor nodes' configuration in CupCarbon based on SenScrip.

CupCarbon also allows for some modifications to a number of parameters such as the type of radio module, channel, and the transmission radius range of the selected radio module. Also, other parameters such as sending, receiving and sleeping energy

consumption, as well as sending/receiving data rate can be identified in CupCarbon. Some available configurations in radio parameters in CupCarbon is shown in Figure 3-2.

CupCarbon also allows for generating multiple route shapes for mobiles (UAVs). The proposed UAV path designs (discussed in Chapter 5) are generated in CupCarbon to evaluate the air-to-Ground performances such as receiving packet rate and energy consumption of communication among the UAV and ground network. Figure 3-3 is shown a defined shape for a UAV moving towards the gateway node, to gather the data from the gateway node which is already connected to its neighbour node.

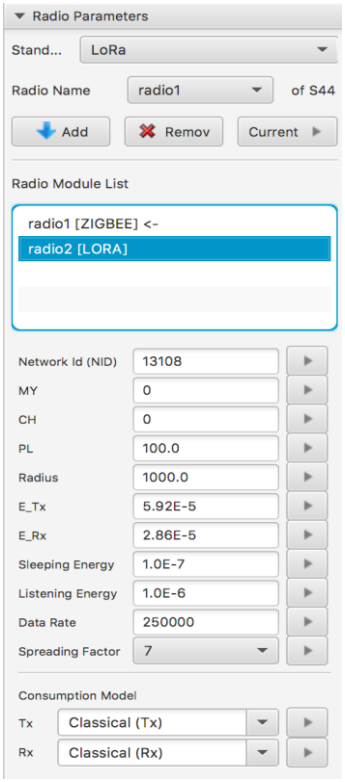
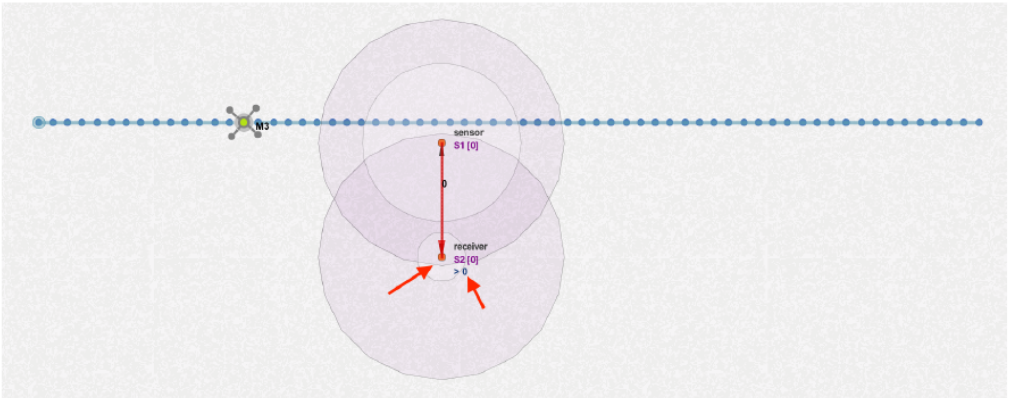


Figure 3-2 Available configurations in radio parameters in CupCarbon.



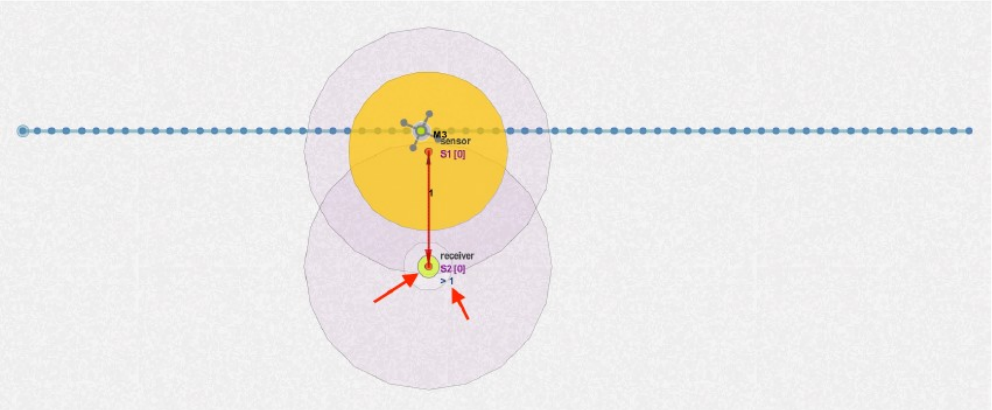


Figure 3-3 A UAV mobility over a designed path.

Note that following termination the simulation, the network performances such as the energy consumption and the number of packets received will be saved in a logfiles in the same directory of the project. Figure 3-4 shows a logfile table of energy consumption performance based on time for an experiment involving three static sensor nodes as transmitters and a mobile UAV as a receiver. Moreover, during the simulation process, some useful information such as errors is displayed in the console for user.

Time (Sec)	S3	S6	S9	S21
0	19159.9998	19159.9998	19159.9998	19160
0.1	19159.99776	19159.99776	19159.99776	19159.99908
0.2	19159.99573	19159.99573	19159.99573	19159.99816
0.3	19159.99369	19159.99369	19159.99369	19159.99724
0.4	19159.99166	19159.99166	19159.99166	19159.99631
0.5	19159.98962	19159.98962	19159.98962	19159.99539
0.6	19159.98759	19159.98759	19159.98759	19159.99447
0.7	19159.98555	19159.98555	19159.98555	19159.99355
0.8	19159.98352	19159.98352	19159.98352	19159.99263
0.9	19159.98148	19159.98148	19159.98148	19159.99171
1	19159.97945	19159.97945	19159.97945	19159.99078
1.1	19159.97741	19159.97741	19159.97741	19159.99078
1.2	19159.97538	19159.97538	19159.97538	19159.99078
1.3	19159.97334	19159.97334	19159.97334	19159.99078
1.4	19159.9713	19159.9713	19159.9713	19159.99078
1.5	19159.96927	19159.96927	19159.96927	19159.99078
1.6	19159.96723	19159.96723	19159.96723	19159.99078
1.7	19159.9652	19159.9652	19159.9652	19159.99078
1.8	19159.96316	19159.96316	19159.96316	19159.99078
1.9	19159.96113	19159.96113	19159.96113	19159.99078
2	19159.95909	19159.95909	19159.95909	19159.99078
2.1	19159.95706	19159.95706	19159.95706	19159.99078
2.2	19159.95502	19159.95502	19159.95502	19159.98986
2.3	19159.95299	19159.95299	19159.95299	19159.98894
2.4	19159.95095	19159.95095	19159.95095	19159.98802
2.5	19159.94892	19159.94892	19159.94892	19159.9871

Figure 3-4 The logfile obtained from a test in CupCarban reflected the energy usages performance for each terminal based on time.

3.3.2 Ground Network Model Implementation in Contiki-Cooja

Contiki-Cooja is chosen as the network simulation software to model the communication on the ground network. In the prosed model (discussed in Chapter 5), multiple tests are conducted for the ground network with multiple functionalities (Leaf, router, and gateway) where the topological structure mode can be dynamically reconfigured based on star, single hop, multihop networks as a compound network topology. The ground network can be structured in Contiki-Cooja considering various network formation scenarios depending on the election of gateway nodes out of the predefined gateway-capable nodes process. As discussed in Chapter 5, the WSN nodes consumed energy and packet delivery rate performances are examined for the proposed model assessment in Contiki-Cooja. Furthermore, Contiki-Cooja provides an input for the subsequent simulation implementation for evaluating the air-to-Ground connectivity performance in CupCarbon, by outputting the accumulated buffered overhead in each gateway. There are several features available in Contiki-Cooja that facilitate the simulation analysis for our proposed software defined-capable model prior to hardware implementation.

a) 'Power-Tracker' Feature

'Power-tracker' is a software-based 'power-profiling' mechanism embedded in Contiki-Cooja which keeps track of the estimated energy consumption of each sensor node without the need for additional hardware. The 'Power-tracker' feature within Contiki-Cooja is accessible via the 'Tools' menu dropdown list. It returns the duty cycle of each node including instantaneous 'Radio TX' (i.e., Radio Transmission), 'Radio RX (i.e., Radio Reception)', 'Radio on' as well as the 'average' of the overall network duty cycles. As represented in Figure 3-5, a multi-hop communication protocol of 7 nodes including four leaf nodes, two routers and one coordinator is structured in Contiki-Cooja. The communication between three leaf nodes to router level-two, router level-one and gateway is demonstrated through the multi-hop communication protocol. The leaf node number four is also connected to router level-one, and router level-one finally forwards data buffered from four leaf nodes to the gateway. The amount of instantaneous duty cycle per network component is obtained based on microseconds in Power-Tracker toolbox (See Figure 3-5).

Figure 3-5 Power-Tracker feature in Contiki-Cooja.

b) Node Mobility

The mobility feature can be added to Contiki-Cooja by patching the mobility plugin. Following enabling the mobility patch, a mobile node path can be set by defining the instantaneous coordinates and the corresponding time-instants of node in a ‘position.dat’ file. Mobility feature of Contiki-Cooja allows us to visualise a homogeneous ground or air-to-Ground network.

As will be discussed in Chapter 5, Contiki-Cooja is used for one-to-one communication between the UAV and each sensor node via air-to-Ground communication.

c) ‘Mote Output’ Feature

Another key feature of Contiki-Cooja is the 'Mote Output' feature, which displays the communication transaction messages between each WSN mote along with the time stamp of the communication. The mote output for the multi-hop network shown in Figure 3-5 is depicted in Figure 3-6. The packet frame design for communication between each network components including leaf, router and gateway nodes is based on the suggested packet frame design in Chapter 4. The communication transaction is initiated from the leaf nodes to the router Level-2 as shown in Figure 3-6. The suggested packet frame design identifies the remaining battery, coordinates, and RSSI level fields on leaf node packet frame configuration (as discussed in Chapter 4). As a result, the values of each leaf node are forwarded to the router Level-2, where they are merged with the router's own packet and forwarded to the upper-level node (router Level-1). Data from three leaf nodes plus the most recently connected leaf node (leaf node four) is collected and

transferred to the gateway via router Level-1. Finally, the entire network data is accumulated within the gateway, as shown in mote output in Figure 3-6.

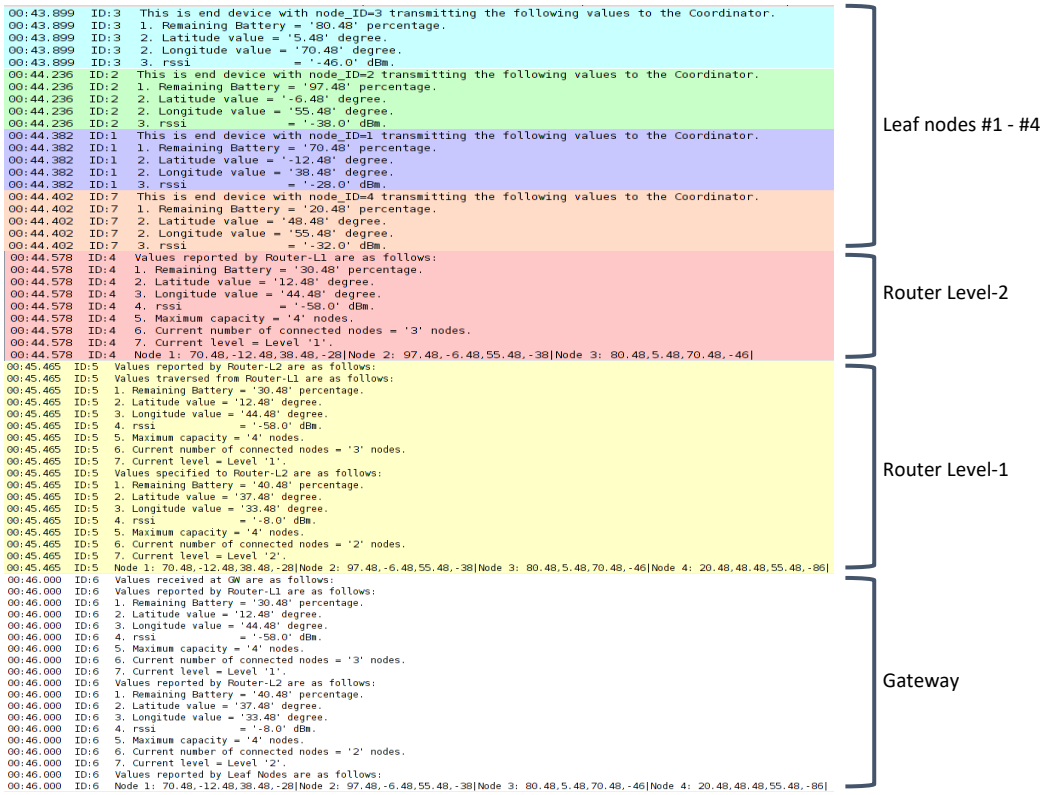


Figure 3-6 The mote output of communication among each network component.

d) Scalability

Another distinguishing feature of Contiki-Cooja that makes it suitable for this research is its scalability or the capability to test the behaviour of models and communication protocols on scaled-up networks. As shown in Figure 3-7, a number of 50 nodes plus one coordinator (UAV) is distributed in Contiki-Cooja environment to test the UAV-Ground behaviour in a more scalable way. As demonstrated in Figure 3-7, the mobility path is used to enable the UAV mobility. The trajectory of the coordinator (UAV) in Contiki-Cooja is exactly represented in accordance with the designed trajectory in Chapter 4 and while the UAV traverses from each sensor node, the communication is executed through the point-to-point connection.

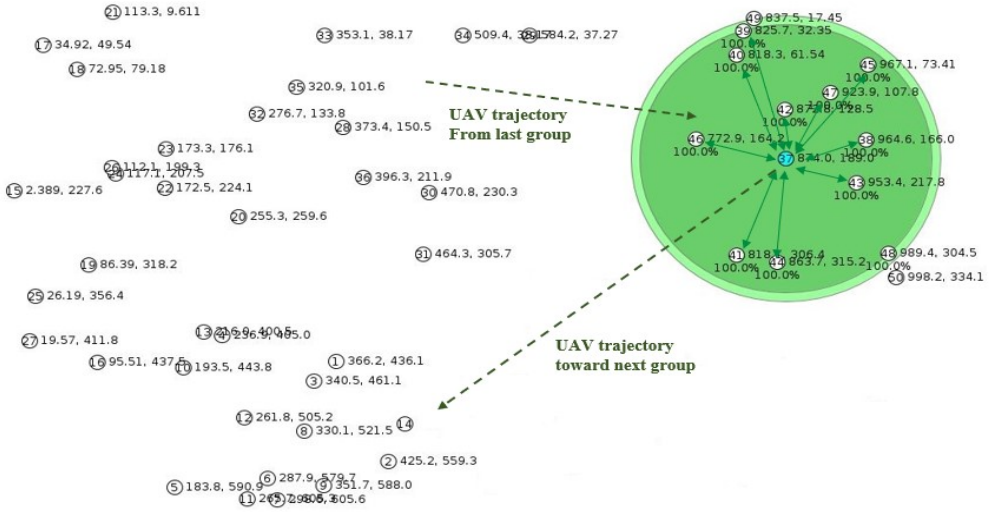


Figure 3-7 The scalability feature of WSN on the communication between the UAV and multiple sensor nodes.

3.3.3 Mission Planner Real-World Validation Tools for UAV Smooth Path

The simulation tool used for the energy efficiency validation of the proposed UAV path based on a real-world scenario is ArduPilot Mission Planner. The designed path in MATLAB is transferred to an embedded software in Mission Planner called (Software in the Loop) SITL flight simulator. SITL enables ArduPilot to be run directly on a PC without the need for any additional hardware. The connected PC is simply another platform on which ArduPilot can be built and run. Once running SITL, the sensor data comes from a flight dynamics model in a flight simulator. Hence, SITL can simulate a wide variety of optional sensors, such as Lidars and optical flow sensors in a simulation environment. SITL also provides access to the full range of development tools available for desktop development, including interactive debuggers, static analysers, and dynamic analysis tools. This makes developing and testing new features in ArduPilot much simpler.

SITL can also simulate devices such as plane, rover, multirotor and helicopters. Among the planes, the quadplanes (broadly known as Vtol) are the targeted UAV for simulation that is used in this research, since quadplanes are hybrid quadcopter-aircraft planes which can take off and land vertically. Once it reaches a desirable height, it moves forwards like a normal plane [102]. Also, because they are able to cover larger areas on a single full battery compare with quadcopters and have less difficulty landing and taking off compare with fixed wings and can hover on top of targeted areas steadily, they are considered as

more reliable options. Figure 3-8 shows SITL environment prior to selecting a desirable aircraft.

Figure 3-8 SITL environment prior to aircraft type selection.

Following the selection of an appropriate aircraft, Mission Planner downloads the required SITL packages for simulation purposes. Then, the flight mission can be defined based on various waypoints localisation and the definition of multiple actions for each waypoint including stabilize, auto, loiter, Return to Launch (RTL), zigzag, circle, etc. Finally following running the simulation, the outcomes are stored in a logfile accessible for post-processing flight analysis.

An example of waypoint planning along with the obtained logfile for an elliptical path are shown in Figure 3-9.

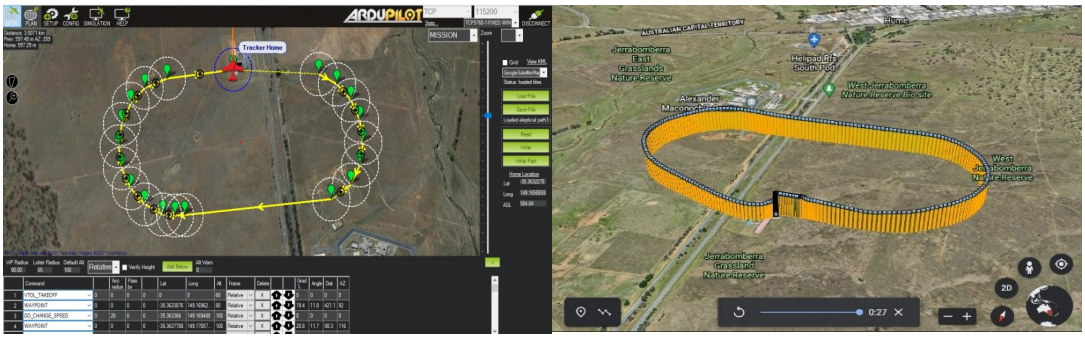


Figure 3-9 Waypoint planning for an elliptical path based on SITL.

In our research, SITL has the responsibility to analyse the instant behaviour of the UAV flight path performance including instant power and velocity variants. Hence, the MATLAB output of the UAV path design module, as discussed in Chapter 5, is used for real-world validation purposes in SITL Mission Planner. To define the simulation outlines on SITL, multiple waypoints are required to be generated on the UAV mission plan. Then, the UAV path is defined in SITL Mission Planner. The proposed UAV path performance

is validated by defining multiple parameters, including volumes of real-time consumed current and speed for the entire path, which are logged in SITL after the simulation is terminated (discussed in Chapter 5).

3.3.4 Co-Simulation between MATLAB and Other Tools

To conduct testing and evaluation for the proposed model, several sequential processes utilising multiple simulation tools must be followed. As shown in Figure 3-10, simulation through each software tool has composed of multiple computations that might get triggered in the event of either a defined component within the same simulation tool or an outputted parameter from other simulation tool. For the proposed model performance analysis in this thesis, four different software tools are used: MATLAB, Contiki-Cooja, CupCarbon, and Mission Planner.

The simulation step process is composed of four modules as shown in Figure 3-10. In MATLAB, the main model entitled ‘UAV fuzzy path’ is defined which determines the UAV smooth path and communication window of connectivity (discussed further in Chapter 5). The module outputs in MATLAB enable the air-to-Ground communication scheduling and ground network communication models in CupCarbon and Contiki-Cooja respectively. Note that the MATLAB outputs also trigger mission planner software to validate the UAV path design. While in Contiki-Cooja and CupCarbon, performance analysis such as packet delivery rate and sensor network consumed energy for enabling communication between the UAV and ground network are examined, the UAV path design in real-world Mission Planner SITL software is validated in terms of energy consumption. Interrelations between modules are two-sided since the model parameters including the energy cost and data-capture dependent parameters need to be characterised to improve model efficiency. Chapter 5 contains a more in-depth discussion of the modules' interrelationships.

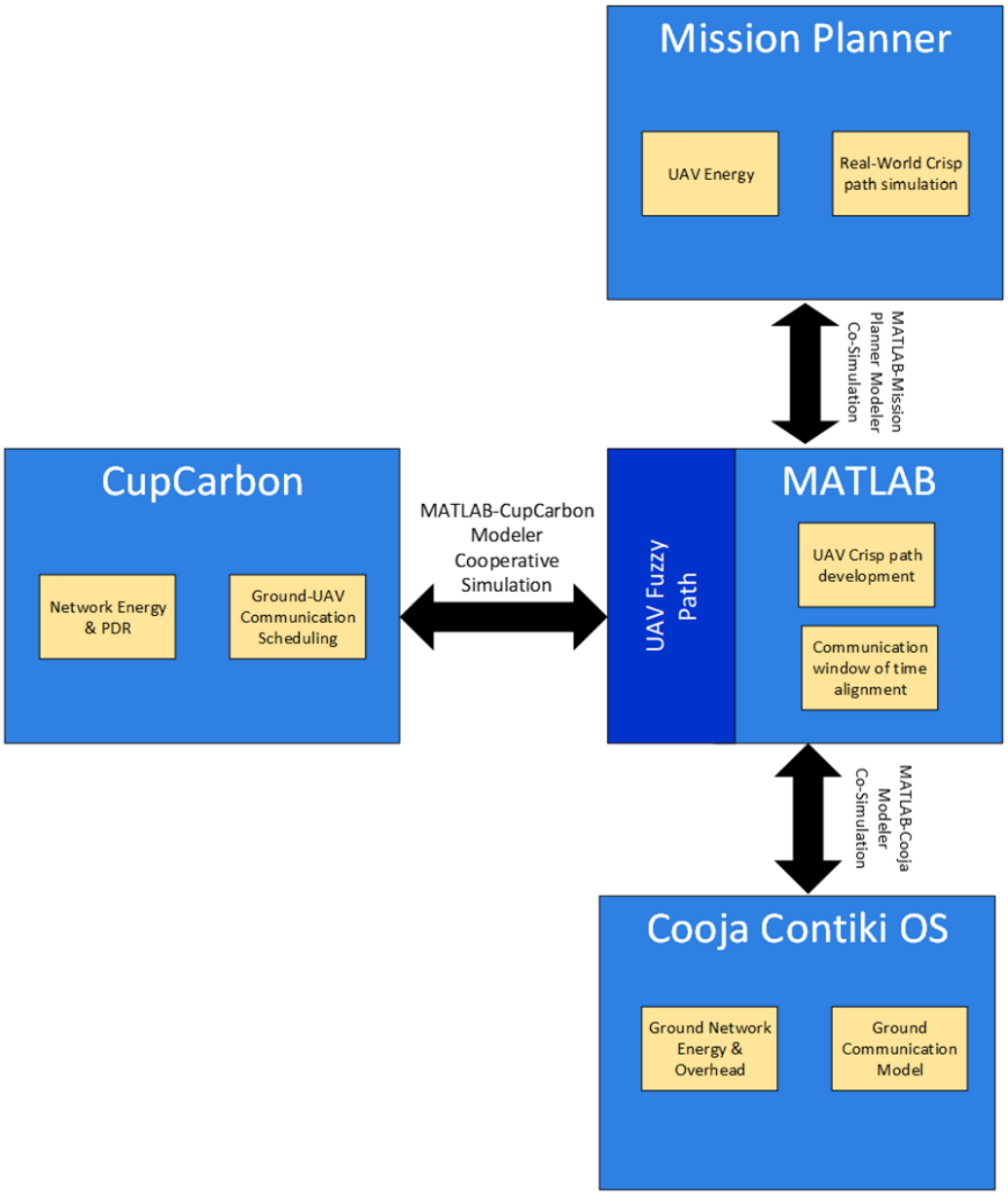


Figure 3-10 Simulation software modules and their relations.

3.4 Texas Instrument CC1352R Target Hardware

A ground network physical layer is typically comprised of a number of wireless sensor-transceiver target modules, each of which is configured with either sensing, routeing, or gateway functionalities. The Launchpad SensorTag kit CC1352R, or 'LPSTK-CC1352R' for short is a powerful Cortex-M4F MCU with integrated environmental and motion sensors, multi band wireless connectivity and simple software for prototyping connected applications. The Launchpad modules include environmental sensors, such as humidity and temperature sensor, ambient light and inertia sensors, Hall-effect switch, and 3-axis accelerometer that can be employed in trap use cases. The Launchpad modules offer a

dual-band low power radio kit which enables Sub-1 GHz (868 MHz/915 MHz) and 2.4 GHz to be run with Bluetooth Low Energy (BLE) concurrently in a single chip solution. Hence, Texas Instruments (TI) 15.4-Stack supports various topology networks for either Sub-1GHz or 2.4GHz applications. The Sub-1 GHz implementation offers several key benefits such as longer range and better protection against in-band interference, as well as the ability to send Bluetooth Low Energy (BLE) beacon packets while operating on a Sub-1GHz TI 15.4-Stack network when using dual-band mode. Herein, the dual-band functionality allows us to modify the functionality of a given SensorTag via over air debugging and configuration functionality using BLE. Alternatively, the SensorTag can be reconfigured by an external programmer-debugger called 'CC1352R LaunchPad Development Board'. Figure 3-11 and Figure 3-12 depict the Launchpad SensorTag CC1352R and LaunchPad Development Board CC1352R respectively.

One advantage of using SensorTags is on their mobility capabilities by using two AAA batteries or a CR2032 coin-cell mounted on the module. These batteries can power up the transmission of their readings in different environments.

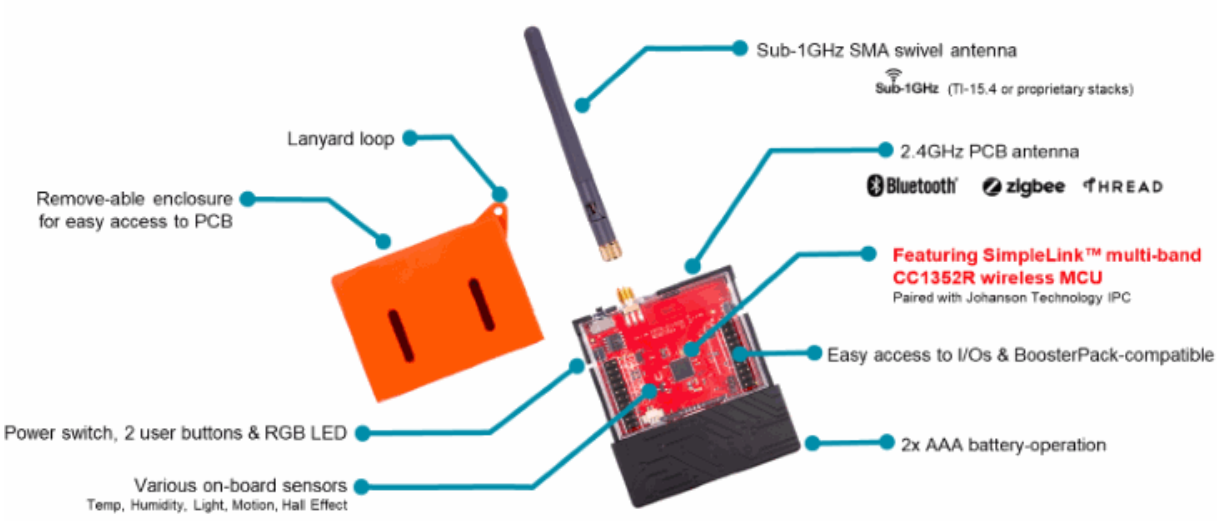


Figure 3-11 The Launchpad SensorTag kit CC1352R [105].

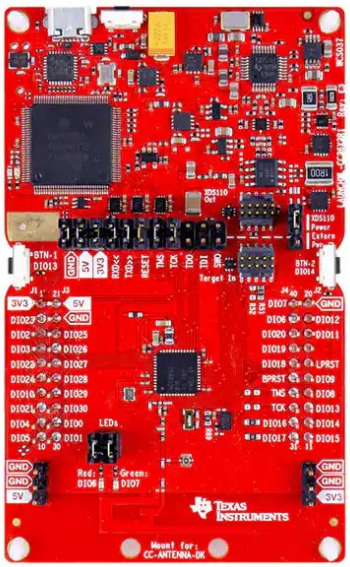


Figure 3-12 The Launchpad Development Board CC1352R [105].

To enable debugging the targeted module, Code Composer Studio (CCS) software is used as an integrated development environment (IDE) that supports TI's microcontrollers. The CCS enables running, debugging, and modifying the functionality of the LPSTK modules through a suite of tools including C/C++ compiler, source code editor, project build environment, debugger, profiler, etc. Herein, each LaunchPad SensorTag requires a LaunchPad development kit for debugging purposes via utilising an ARM 10-pin JTAG cable and a 2-wire pair for UART. The SensorTag can be connected to the development board allowing for full debugging, programming, and UART communication via CCS. Figure 3-13 depicts the connection between LaunchPad SensorTag and LaunchPad development kit to enable debugging of the LaunchPad SensorTag.

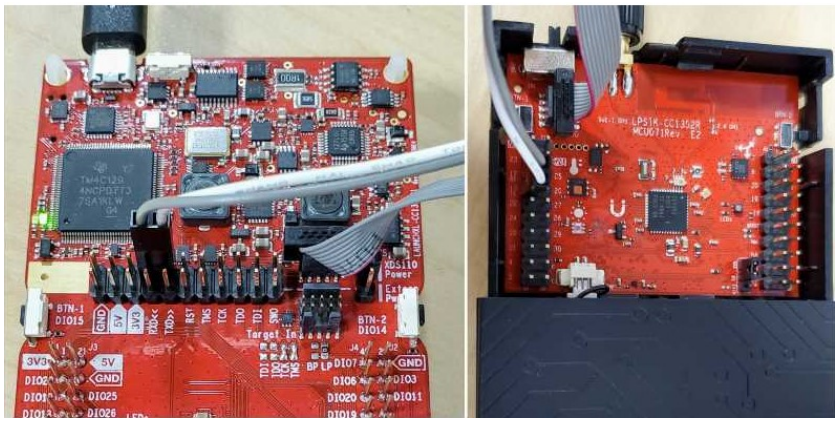


Figure 3-13 Debugging the LaunchPad SensorTag CC1352R using Development board CC1352R [105].

Within the CCS, there is an embedded GUI called ‘SysConfig’ which is an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, radios, subsystems, and other components of CC1352R. The output involves C header and code files that can be used to configure custom software. The configuration via SysConfig for the targeted hardware is shown in Figure 3-14. As demonstrated in this figure, some configurations are available in SysConfig ranging from Radio to Network, Application, MAC, Power, and Security. Some Physical layer configurations such as Frequency Band (for instance Sub-1 GHz frequency) and data bit rate can be performed within the Radio part. Under the network menu, channel selections of the network and the maximum number of children devices that can be supported by the coordinator can be configured. Under the application menu, configurations on reporting intervals (how often the device will report data) and polling intervals (how frequently a sensor device polls its parent for data) can be set. Setting the minimum and maximum back-off exponents under the MAC configuration is possible, which is determined by the backoff time of the collision avoidance mechanism in the CSMA-CA algorithm. If the minimum value is set to 0, for example, collision avoidance is disabled. Within the Power menu, the transmit power in dBm can be configured. Lowering this value helps to reduce energy consumption when nodes in the network are close to each other. Security and other dependencies sections menus offer some settings on security configuration of targeted module and some modifications related to other external modules like their interrupt priorities, etc.

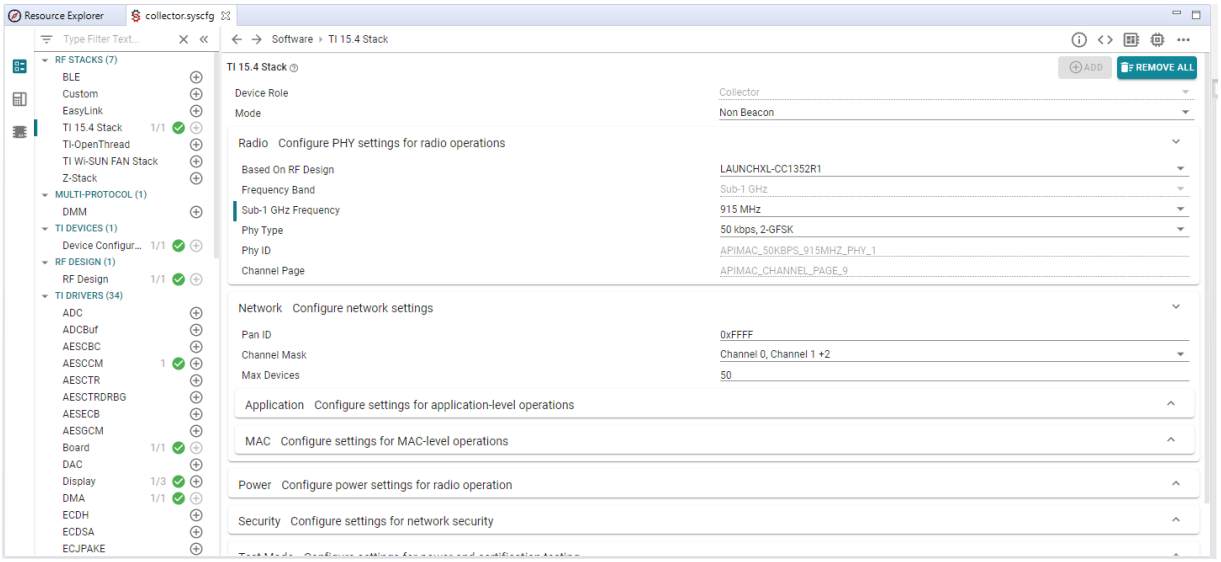


Figure 3-14 The SysConfig GUI for configuring the target hardware.

3.5 Conclusion

The envisioned UAV-enabled WSN data gathering model involves a data flow process from distributed sensor nodes to the UAV while taking required performance metrics into account. This chapter provides an overview of the research methods and tools used in this study. The viability of employing the open-source Contiki-Cooja, CupCarbon as tools for sensor network modelling in Ground or air-to-Ground network are discussed. Moreover, an overview of available network simulation tools such as NS-2, NS-3, OMNET++ and OPNET is provided. Later, the SITL Mission Planner visualisation software tool for characterising the required parameters for the UAV path design is described. The interconnection of the chosen simulation software tools is also expressed in order to look into the required step processes towards testing the comprehensive model. By providing some examples, certain key features of these tools have been highlighted. Finally, a physical WSN system is highlighted using the suggested hardware platform of Texas Instrument CC1352R modules that can provide the necessary processing and code storage memory to enable testing, debugging, and configuring the physical nodes.

Chapter 4 UAV Fuzzy Route Data Collection through SDWSN

4.1 Introduction

This chapter provides insight into the concept developed towards designing a comprehensive strategy for UAV-based data collection from wireless sensor networks (WSNs). The proposed approach aims at enabling an energy-efficient UAV-aided WSN data collection that allows for mutual adaptability among UAV relaxed path design and the ground network topological arrangements. Thereby, in this chapter, the terms 'UAV fuzzy travel path' and 'software defined wireless sensor network (SDWSN)' are jointly used to support UAV crisp path design and facilitate network topology shifting respectively. This research examines the emerging role of SDWSN in the context of UAV path planning improving network performance and UAV energy efficiency through dynamic orchestration of wireless ground sensor nodes. This also enables a more flexible framework of ground network that can effectively be restructured based on the planned path of the UAV incorporating software defined network concept. Herein, the objectives of this research are to determine the comprehensive solution on UAV-aided WSN data gathering model taking into account three crucial areas, namely UAV path planning model, ground network structure and UAV-Ground communication incorporating the SDWSN functionality.

This chapter is organised as follows: The first section begins by laying out the proposed data collection system organisation by defining the main sub-components. The data collection parameters' involvement is studied in section 4.3. The overall structure of the UAV fuzzy path takes the form of forth section. Also, to establish SDWSN concept on the ground network, section 4.5 highlights ground communication aspects in detail. Next, section 4.6 is concerned with the methodology used for air-to-Ground communication. Finally, section 4.7 concludes the chapter.

4.2 Data Gathering System Organization

With rapidly declining wildlife populations, viable wildlife management is a crucial concern across the world. Wildlife monitoring is a vital activity to sustain the populations of animals in danger of extinction. Hence, the need for animal tracking has arisen in scientist research. Trapping predators can be a promising solution for monitoring their

population by laying out a network of traps and servicing their operation on a large scale. However, due to the vast areas these traps cover, and the desire for a sizeable amount of maintenance initiated by rangers, traps monitoring could be a challenging process. Utilising advanced technologies for data gatherings from traps using Wireless Sensor Networks (WSNs) and Unmanned aerial vehicles (UAV) could offer an effective solution with minimal human intervention.

Wireless sensor networks (WSNs) are typically made up of several low-cost distributed trap sensor nodes (SNs) that are typically powered by finite energy sources such as batteries, which are difficult to recharge once depleted. Hence, energy-efficient sensing and communication techniques among SNs are crucial to prolong the network's lifetime.

The UAVs provide cost-effective and interesting solutions for surveillance applications. Given the maximum payload the aircraft can carry, an improvement in energy efficiency can directly increase the UAV's flight time, allowing it to communicate with more WSNs before being recalled for recharging. Hence, designing an energy-aware UAV path planning that takes into account the communication aspects with the ground network is a crucial factor in improving the network's quality of service.

In the proposed method, UAV-based data acquisition of WSNs can be categorized into three major sub-sections: UAV path-planning model, group network communication, air-to-Ground communication (as shown in Figure 4-1). Each area is represented by the main factors affecting the structure of the proposed system, which will be discussed in the following sections. The proposed model applied in each of these three parts should offer scalability and mutual adaptability in both ground network and UAV path design.

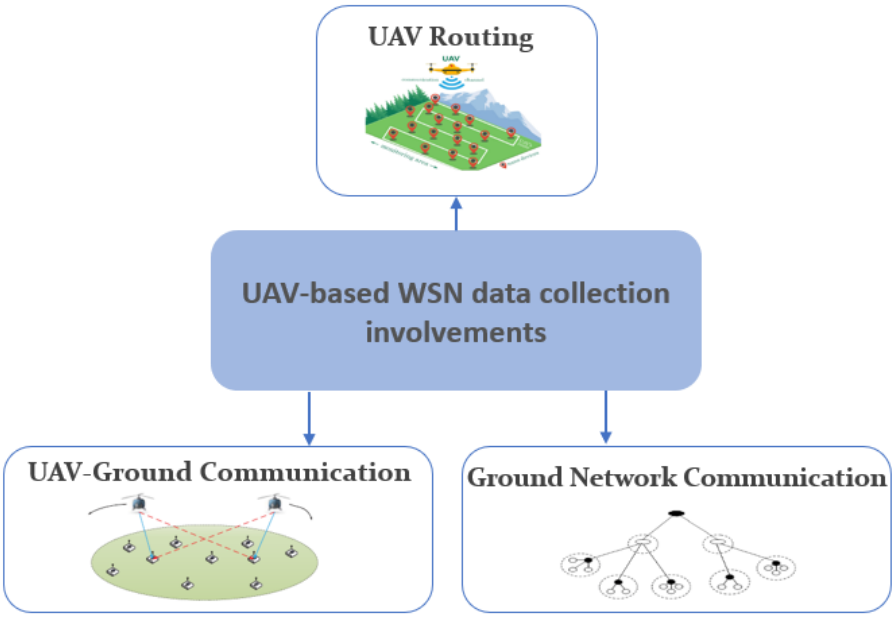


Figure 4-1 Data gathering system organisation.

4.3 UAV-based WSN Data Collection System Architecture

The organisation of the proposed network system supported by the cloud environment reflects the overall structure of a UAV-Ground network operating system. The proposed architecture for the UAV-Ground network system consists of two main parts: the virtual UAV-Ground network operating system and the physical UAV-Ground network operating system. As illustrated in Figure 4-2, the virtual UAV-Ground operating system contains the resources that are available in the cloud such as data storage, network virtualization unit and other capabilities for processing and intelligence which offer intelligent interaction with the physical network. While the physical UAV-Ground network operating system contains the actual real-world implementation of a ground network of distributed SNs wherein the physical network is connected to the virtual UAV-Ground operating system via the Internet through a UAV. In both virtual and physical network operating systems, ground network contains three types of nodes that are identified as three main roles in WSN. These are Leaf, Router and Gateway roles (will be discussed in detail in section 4.5). A given node may have one or more of these roles at the same time. The UAV also plays a crucial role in facilitating the exchange of information as it can establish a connection with remote cloud computing centres through the Internet. Hence, the drone station serves to capture the data from the actual ground network and relay that to the cloud over the Internet for real-time post-processing

assessments and virtualization of main stations. Cloud resources, mainly network virtualisation unit, could offer intelligent interaction with the physical network. Virtualisation unit also allows the virtual testing of a node or a network of nodes before their actual physical implementation, preventing any major modification. Hence, network operation can be tested, modified, and structured to meet the requirements of the physical network prior to the real-world implementation. The WSN virtual cloud also enables us to process the post-processing assessments such as fitness model for the upcoming election operations enabling ground network re-orchestration functionality. Further explanations on these operations are expressed in detail in the following sections.

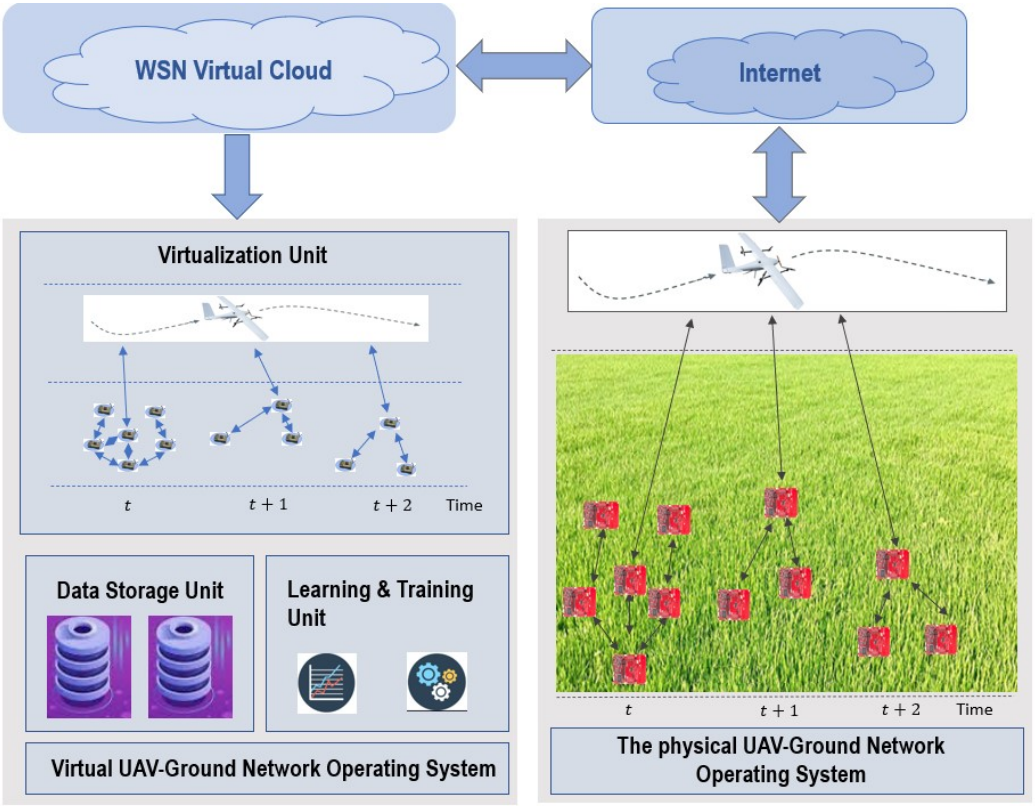


Figure 4-2 Organisation of the proposed network system.

4.4 UAV Energy Efficient Path Planning

Conventionally, UAV-assisted data collection is based on either direct connection with individual nodes or indirect communication with group representative nodes (referred to as cluster head). Figure 4-3 depicts an example of how such a formation might appear. While grouping in this formation allows for more efficient data transmission, it still limits the UAV path for data gathering to the limited options once the actual topology of each

group is structured. The main question is, can we relax the UAV route options by allowing the cluster heads to be elected to match the data collection path?

Flight path relaxation allows the UAV to lower its energy expenditure in turning points once the UAV is within the sharp edges. This facilitates the minimum variations in speed and acceleration required at turning points reducing the UAV flight propulsion energy consumption. This method also allows for the dynamic redefining of group topologies via softwarisation. In this case, the CHs are software redefined to meet the requirements of the flight route.

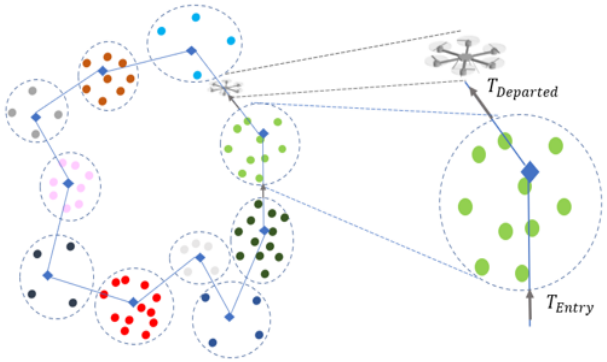


Figure 4-3 Conventional data gathering organisation from the distributed WSN.

4.4.1 Fuzzy Path Concept

The term ‘Fuzzy Travel Route’ is used in this thesis to describe the ability to software redefine the network organisation to align with the UAV flying smooth route. It is necessary here to clarify exactly what is meant by UAV fuzzy path. Fuzzy travel route concept is defined as a UAV path span that enables the UAV flight path to be chosen from a wider range of alternatives rather than being fixed in one defined path. The choice of the UAV flight’s dynamic parameters such as path shape, flight speed and acceleration variations could provide more options and be designed to maximize the efficiency of travel while reducing data loss. This organisation allows the UAV path to be dynamically adjusted in accordance with the updated ground network topology. The main contribution of this research is on employing the software defined network orchestration concept applied on ground network and exploring its impact on data collection process’s robustness and evaluating its effect on the UAV’s movement and ground network’s energy efficiency and jointly analysing the UAV path energy consumption and ground network energy expenditure.

The fuzzy route approach is based on the ability of modifying a given node functionality from leaf node role to a gateway one or vice versa. These nodes are known as gateway-capable nodes. Due to the UAV path relaxation arising from a larger fuzzy range, this method can enhance the overall system robustness and efficiency. Herein, the proposed topology structure eases the UAV path by incorporating efficient and crisp route designs within the UAV route fuzzy region and providing the resilience for adaptive gateway election closer to the UAV path.

For generality, the proposed model considers the random distribution of ground network, considering both dense and sparse nodes' distributions over a given testing area. The goal is to identify the model by taking the distribution and density of ground network into account as two key factors (discussed in the following section). According to the proposed model, neighbouring nodes that are within LoS of each other form a WSN group that is controlled by one or more ground representatives acting as cluster head(s).

The design of fuzzy path following with the inserted smooth paths for each distribution and density scenarios are shown in Figure 4-4. The non-uniform path planning method via Bezier curves in sharp edges is one solution for path alignment within fuzzy route to overcome higher energy expenditure in sharp edges (See Figure 4-4- I-III). This UAV path design, on the other hand, has variable speed, which causes inflexibility in real-world path planning and leads to UAV energy efficiency degradation (discussed in Chapter 5). Hence, an example of a relaxed UAV path fitted within a fuzzy flight route using circular arcs/lines geometry is suggested as shown in Figure 4-4-IV. This is demonstrated using two semi-circular routes with constant velocity and two linear and level flight paths with variable velocities (discussed in Section 4.4.4 in detail). The proposed geometric semi-circular routes provide a better performance in terms of energy efficiency allowing for constant velocity travels for the UAV throughout the curves. The velocity variation relaxation can improve the UAV energy consumption for the entire tour and at the same time lower the complexity of the flying path. According to the simulation results, using the fuzzy smooth path based on multiple linear and curved portions of the overall path not only improves the energy efficiency of the entire tour but also provides the simplest real-world flight path implementation (discussed in Chapter 6).

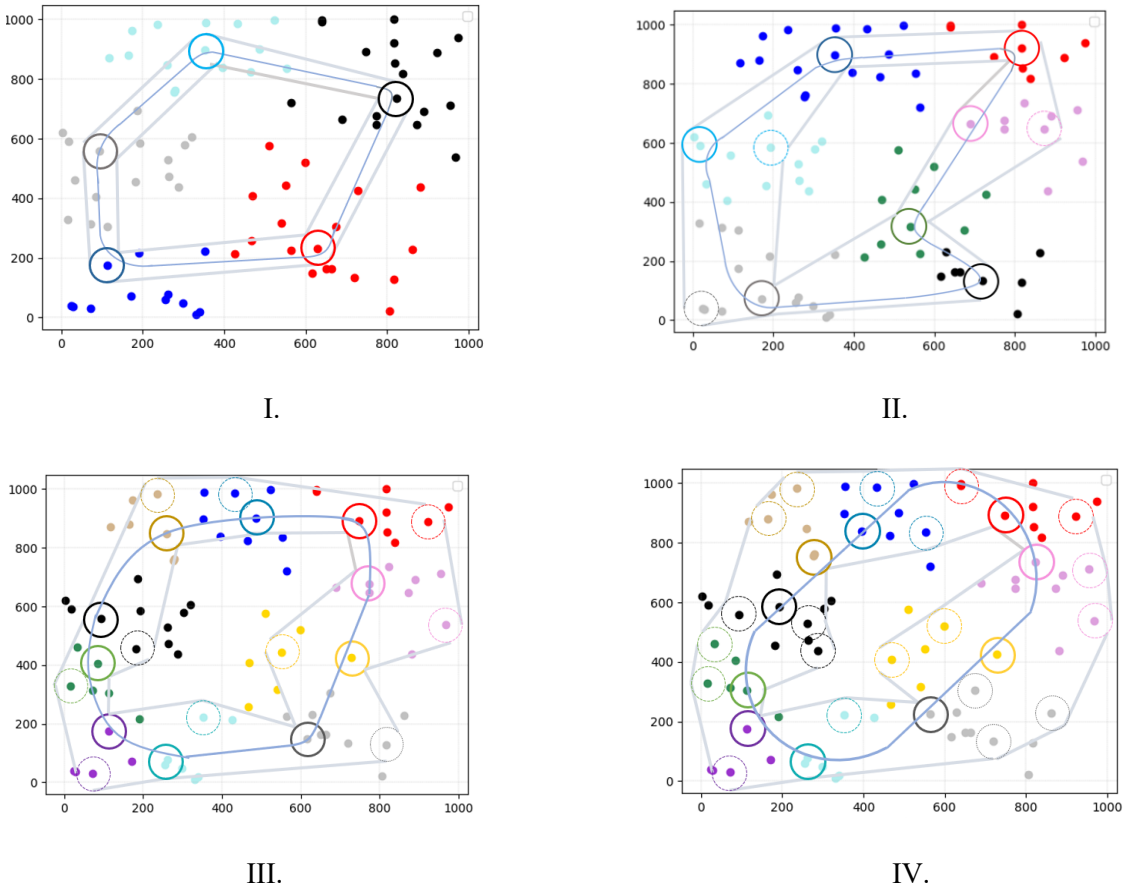


Figure 4-4 The design of fuzzy path and smooth route with various distribution of gateway-capable nodes and densities (from figure I to figure IV. the distribution of gateway-capable nodes and densities raised).

As shown in Figure 4-4 I-IV, the elected gateways are depicted by solid circles per group while the remining potential gateways are demonstrated by dashed circles in the network structure. The UAV fuzzy route is represented by the bounded space between the grey lines.

In short, the fuzzy flight route approach presents the potential for relaxing the specified options for the flight tour over a given group without jeopardising the communication LoS. Depending on the capability of the WSN group members to presume the role of gateway, the more spread out the gateway-capable members, the better the smooth flight space.

4.4.2 Fuzzy Path Operational Parameters

To begin with, based on the proposed topological set-up, we assume that the gateway-capable nodes are spread such that one or more of them are not isolated from the remaining gateway capable nodes (there should not be any isolated gateway-capable node

in the ground network). The proposed fuzzy path is highly dependent on multiple parameters, one of which is the percentage of gateway-capable nodes population. This metric specifies the percentage of gateway-capable nodes population out of the total sensor node population. Figure 4-4 illustrates the potential for relaxing the special options for the flight route over a given group without disadvantaging the communication line of sight. Depending on the ability of the wireless sensor group members to assume the role of being cluster head, the more capable members, the better the relaxed flight space. As shown in the Figure 4-5, three plots have demonstrated the impact of increasing the number of cluster-head-capable sensor nodes on the UAV coverage fuzzy path. For example, with growing the number of capable cluster heads to 3 and 5 nodes within a 10 nodes cluster, the flight region over the cluster widens without affecting the LoS. Hence the number of contributors within each cluster (leaf nodes or cluster heads) have major impact on the width of the proposed fuzzy path. Notice that from this figure, the potential cluster heads relate to the available space that offers consistent line of sight. Hence, the proposed topology structure allows for the UAV's crisp path design by minimising the variations of flight velocity, acceleration, and length of the path. This approach could also present the flexibility of CH election closer to the UAV route.

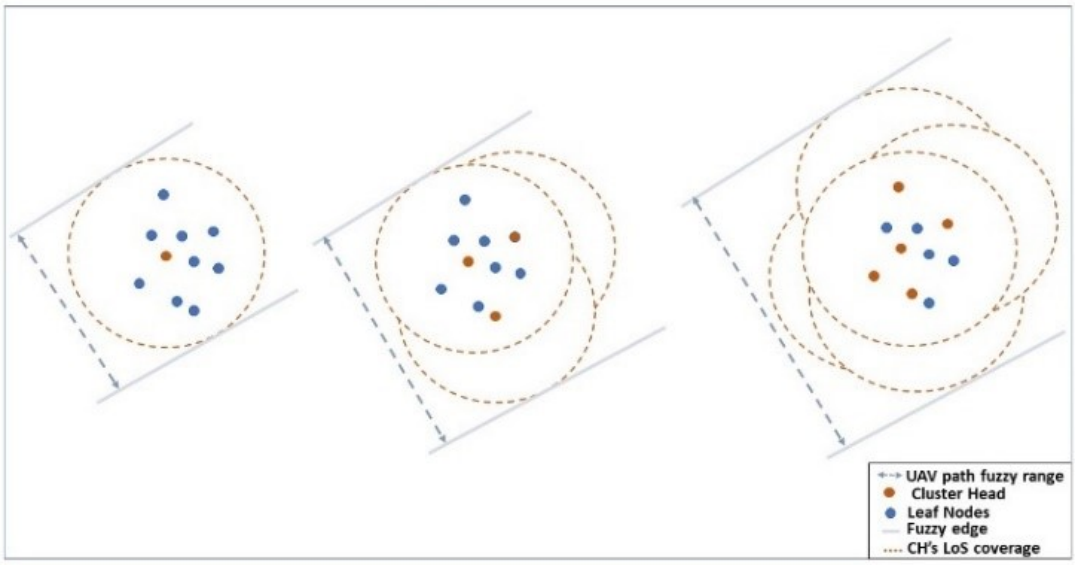


Figure 4-5 The impact of raising the number of gateway-capable nodes on the UAV coverage fuzzy path.

The proposed model is also reliant on other defined metrics, including spread factor ratios and gateway election factor. The spread factor ratio identifies the dispersing factor of the gateway-capable nodes for each group within the network. Spread factor is utilised in

characterising the width of the path which is defined based on the given specific geographic dimension and gateway capable nodes distribution expressed in (4-1):

$$\sigma_{gc} = \sqrt{\frac{\sum_{i=1}^{N}(q_0^i - \mu)^2}{N}}\,. \tag{4-1}$$

Proof: See Appendix A.1. In which σ_{gc} is the spread factor, q_0^i is the coordinates of gateway-capable nodes, μ is the average coordinates of total gateway-capable nodes, N is the number of gateway-capable nodes and m is the number of groups in the ground.

The latter one represents the percentage of the gateway capable nodes populations that have been configured as gateways in each configuration election phase (discussed in section 4.5 in detail). The gateway election factor for each configuration election process is calculated in (4-2):

$$\xi_i = \frac{n_i}{m} \tag{4-2}$$

Wherein n_i is the number of gateways elected in ith configuration election process. While m stands for the number of predefined gateway-capable nodes. These three aforementioned factors are the main parameters in defining the distribution and density of gateway-capable and elected gateway nodes among the total ground network entities.

As shown in Figure 4-4, the effect of increasing the percentage of the gateway-capable nodes and spread factor ratio while maintaining a constant amount of gateway election factor on the fuzzy path region and hence UAV flying route is demonstrated. Once the percentage of the gateway-capable nodes and the spread factor ratio are limited to the specified threshold of $D_{gc}(th)$ and $\sigma_{gc}(th)$, the UAV path shape loses its fluidity. On the other hand, this can be improved to a more relaxed shape with less sharp edges by utilising the Bezier curve enhancement concept (according to Figure 4-4 I-III). Conversely, once the percentage of the gateway-capable nodes and the spread factor ratio are higher than the given threshold, the UAV fuzzy region is expanded, facilitating more flexibilities for the UAV path definition, and hence the UAV path gets closer to the crisp path with minimal sharp edges (See Figure 4-4 IV). Hence, the proposed topology organisation allows for the UAV's crisp route capability by minimising the variations in velocity,

acceleration, and length of the path. This approach also provides the flexibility of gateways election closer to the UAV route design. The proposed fuzzy software-defined capable relaxed path topology organisation is illustrated in Algorithm 1.

Algorithm 1 Fuzzy software-defined capable Smooth Path Designing

1: **Inputs:** the percentage of distributed gateways-capable nodes $D_{gc}(\%)$ with a set of gateway-capable nodes N with their positions $U_{gc(i)} \in R^{2\times1}$, the spread factor of gateway-capable nodes σ_{gc} the set of $K-Means$ cluster heads centroids V

2: **Initialize** 'Centroid-based trajectory enhancement approach (CTEA)' trajectory design utilizing conventional Travel Salesman Problem (TSP) connecting the centroids

3: Workout the fuzzy-path covering the defined CTEA trajectory and transmission range of all gateway-capable nodes

4: **for** every cluster head centroid V **do**

5: **if** $D_{gc} \leq D_{gc}(th)$ **OR** $\sigma_{gc} \leq \sigma_{gc}(th)$ **then**

6: apply Bezier curve to smooth the path and calculate the UAV energy expenditure using eq. (2) & (5)

7: **else**

8: apply circular arcs/lines geometry to smooth the path and calculate the UAV energy expenditure using eq. (3) & (4) for each segment of the path.

9: **end if**

10: **end for**

11: **for** each cluster of nodes **do**

12: select the gateways based on update fitness model for the next round

13: **end for**

14: **Output:** the energy efficient data gathering path

Finally, as represented in Algorithm 1, the elected gateways are chosen based on the fitness model including level of proximity to the aligned path, energy consumption and remaining capacity of the capable nodes (discussed in Section 4.5.3). Then, the ground network structure will be reconstructed based on the nominated gateways out of the gateway-capable nodes per group.

4.4.3 Formulation of UAV Energy Expenditure Model

Firstly, to present the enhanced UAV energy consumption within the UAV flight path fuzzy range, the proposed circular arcs/lines crisp route versus Bezier curve route performances taking instantaneous UAV's speed, acceleration and power usage are highlighted. The UAV circular arcs/lines path is mainly split into two geometric parts including circular curved and linear flight paths. While the Bezier curve one is divided to linear and Bezier curved paths which are defined to mitigate the sharp turning points in the UAV path design. In UAV curved movement model, there are two parameters that define the acceleration vector a.

Centrifugal acceleration $a_{\perp}$: the acceleration parameter that is perpendicular to the UAV speed vector and causing heading (direction) changes without any variation in the UAV velocity (orange arrow in Figure 4-6).

Tangential acceleration $a_{\parallel}$: the acceleration parameter that is in parallel to the UAV's speed vector and is defined based on the variable speed in the path.

These two acceleration vectors are depicted in Figure 4-6.
The UAV energy consumption considering variable velocity $v(t)$ and acceleration vectors $a(t)$ is expressed in (4-3) [42]:

$$\bar{E}(q(t)) = \int_0^T [c_1 \|v(t)\|^3 + \frac{c_2}{\|v(t)\|}(1 + \frac{\|a(t)\|^2 - \frac{\left(a^T(t)v(t)\right)^2}{\|v(t)\|^2}}{g^2})]dt \qquad (4\text{-}3)$$

$$+\frac{1}{2}m(\|v(t)\|^2 - \|v(0)\|^2)$$

in which c_1 and c_2 are two parameters related to the aircraft's weight, wing, air density, etc., g is the gravitational acceleration with nominal value 9.8 m/s^2, m is the mass of the UAV including all its payload.

It is obvious from (4-3) that the UAV energy usage is dependent on the instant UAV speed and acceleration vectors rather than the variation of these vectors.

The tangential and centrifugal accelerations can be aggregated to the acceleration vector a in which each component has its own impact on the UAV energy expenditure. In the

Section 5, the impact of tangential and centrifugal accelerations on the UAV energy consumption is analysed.

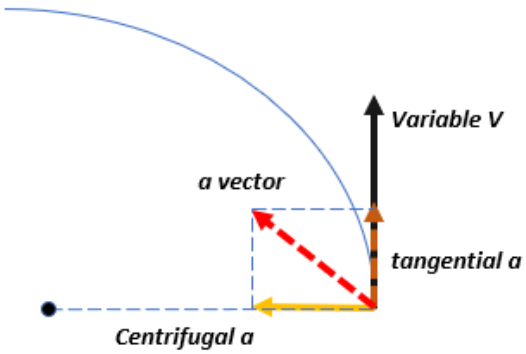


Figure 4-6 The speed and acceleration vectors in UAV movement while traversing over the curve.

4.4.4 Utilization of Fuzzy Path for Defining UAV Crisp Paths

a) UAV circular arcs/lines geometric path

A crisp circular/linear UAV route that is fitted within the fuzzy route is expressed in this section. This is constructed using two semi-circular paths with constant velocity along with two linear and level flight paths with variable velocities.

The value of centrifugal acceleration is zero at the linear path. Herein, the tangential acceleration is responsible for altering the UAV velocity. The energy consumption highlighted in (4-3) for linear flying route is reduced to (4-4):

$$E_{straight\ flight} = \int_0^T (c_1 \|v(t)\|^3 + \frac{c_2}{\|v(t)\|}) dt. \quad (4\text{-}4)$$

The optimal acceleration and velocity for the linear flying route is obtained as $a^*(t) = 0$ and $v^* = ({}^{c_2}/_{(3c_1)})^{1/4}$.

For the semi-circular flight paths, the acceleration is perpendicular to the velocity and the path direction to ensure that there is no velocity variation. Moreover, the value of the acceleration vector for maintaining the circular path is constant and is obtained by $a(t) =$

$\|a(t)\| = v^2/r$. This is the only acceleration component for the heading change. Herein, the energy expenditure in (4-3) has been reduced to (4-5):

$$E_{cir}(v,r) = T\left[\left((c_1 + \frac{c_2}{g^2r^2}\right)V^3 + \frac{c_2}{V}\right]. \quad (4\text{-}5)$$

From (4-5), the optimal velocity in the curves is obtained which is dependent on the value of curve radius as $v^* = (\frac{c_2}{3(c_1+c_2/(g^2r^2))})^{1/4}$. This allows for the minimum energy usage per curve. The proposed approach including multiple linear/semi-circular segments has been depicted in Figure 4-7. Based on this figure, the UAV crisp path is divided into 5 segments:

- At the first part (segment-1–Figure 4-7), once the UAV initiates its tour from the starting point to travel through the curve, its velocity changes from zero to the optimal constant velocity $v^*_{curve} = (\frac{c_2}{3(c_1+c_2/(g^2r^2))})^{1/4}$. This means that the UAV has the variable speed throughout this traveling part. The energy that the UAV consumed is calculated using (4-4).
- During the second part (segment-2– Figure 4-7), the UAV flies over the first circular path with its fixed velocity but variable heading. The UAV energy is obtained using (4-5).
- At the next part (segment-3– Figure 4-7) of the linear flight route, UAV changes its speed from the constant value within the curve to the optimal one for the linear route $V^*_{Straight} = (^{c_2}/_{(3c_1)})^{1/4}$. As the UAV gets closer to the next curved part, it adjusts its speed to the specified speed value for the curves. Herein, the value of velocity is variable in this part.
- The second curve is subjected to the same procedure as the first (segment-4– Figure 4-7).
- Finally, in the distance between the end of second curve and ending point, the UAV brakes to land on the ground (segment-5– Figure 4-7).

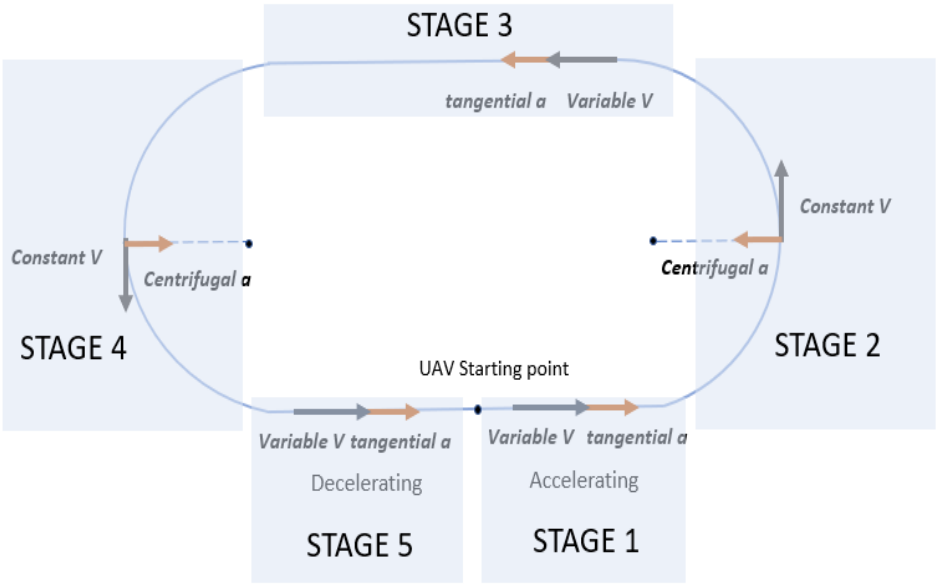


Figure 4-7 UAV circular arcs/lines path design within the UAV fuzzy range.

b) Bezier Curve and path smoothing

Bezier curve is initially published by Pierre Bezier in [103] for shape design of automobile bodies. Unlike other type of curves such as polynomials or cubic splines, Bezier curve is totally contained within the convex hull of its control points utilised to identify the Bezier curve.

Bezier curve does not traverse all the control points except the first and the last control points. According to [39], Bezier curves specified with four control points offer UAV relaxed paths. Herein, four control points are involved in defining each curvature over the TSP route [39]. Accounting for four control points, P0, P1, P2 and P3, for each curve in which P0 is the position of the initial point at the beginning of the curve and P1 and P2 are the points within the closest vicinity of the given visit point within curvature and P4 is the ending visit point, they uniquely create the third-order Bezier curve as shown in Figure 4-8. We propose a crisp route called Bezier curve TSP (BTSP) to travel over a given set of n visiting points (n is the number of gateways) by traversing along the route that meets the restrictions of the proposed fixed-wing UAV. This provides a route to connect a sequence of Bezier curves, where n Bezier curves (one curve per vising point) are connected as the final trajectory. The path vector third-order Bezier curve is expressed in (4-6):

$$X(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3 \quad (4\text{-}6)$$

wherein $0 \leq t \leq 1$ and P_k stands for the k-th control point. Also, the velocity and acceleration vectors $v(t)$, $a(t)$ are defined as derivation of the path vector $X(t)$ and $v(t)$ with respect to the normalized time t respectively. Consequently, the curvature κ at each point on the Bezier curve is obtained based on the first and second derivatives with respect to t as calculated in (4-7):

$$\kappa(t) = \frac{\left|\dot{X}(t), \ddot{X}(t)\right|}{\|\dot{X}(t)\|^3} \quad (4\text{-}7)$$

wherein $\dot{X}(t)$ and $\ddot{X}(t)$ are first and second derivatives of $X(t)$. It should be noted that the curvature is a function of the parameter t, that it is not essentially constant along the curve, and that it changes as the UAV moves. The curvature also defines the UAV's maximum possible tangential speed along the trajectory [39].

The proposed BTSP is initially highlighted based on the predefined TSP path. Next, the Bezier curve is used to enhance the smoothness of TSP path within the direct changings. This path is employed as the benchmark scheme for comparison with the proposed linear/semi-circular crisp path. The goal here is to analyse the value of energy consumption per UAV flight time using (4-3) based on the benchmark scheme and compare it to the proposed geometric UAV path. The simulation outcomes in Section 5 have supported the robustness of proposed path.

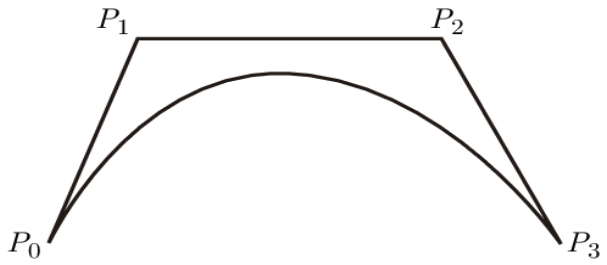


Figure 4-8 Curve fitting for cubic Bezier curve.

4.4.5 UAV Path Design Validation through The Real-World Simulation

For validating the simulation model, Ardupilot mission planner is employed to support the efficiency of the proposed linear/semi-circular path over BTSP flight path and assess the instantaneous behaviour of the UAV performance involving instant power and speed variants. Within Ardupilot Mission Planner, a simulation embedded software entitled software-in-the-loop (SITL) is used which offers simulation prior to arming the vehicle

and running the mission in real world [100]. Also, the VTOL quadplane UAV type is chosen for simulation analysis. The detailed validation of the proposed path designs is outlined in Section 5.

4.5 Ground Network Communication

4.5.1 The Proposed SDWSN Concept on Dynamic Ground Network Orchestration

The proposed ground network communication architecture is based on splitting the entire system into multiple network components: Ground Network, Drone, Cloud stations. Leaf, Router and Gateway roles are three major roles in ground network formation as shown in Figure 4-9. A node may be assigned to one or more of these roles. The drone terminal is also a mobile vehicle that collects data from the ground and sends it to the cloud via the Internet for instantaneous post-processing evaluations and virtualization purposes.

Mainly, the network communication messages in SDWSN are split into two main messages: sensing data information messages and control information messages. As the main goal of data acquisition effort is on passing the sensing data information messages, the control information messages have the responsibility to convey the ground network topological orchestration data such as election data from the cloud-level to the ground network and assigning the optimal functionalities to each component of the network in the initial scanning topological pre-orchestration and notification phases (will be discussed further in Sub-section 4.5.2). Each network component's functionality is explained further as listed below.

a) Leaf node

Leaf nodes are the lowest functional roles in the network, predefined as prior to the UAV traveling; Leaf nodes are responsible for the data acquisition from connected sensors and passing sensing data and the critical control information to the higher-layer routers or gateways. They are not in direct contact with the UAV during the topological pre-orchestration scanning or data collection post-orchestration phases. Leaf nodes can also be software redefined to act as fully functioned router nodes or to enable or disable a given query or data processing function.

b) Router-Capable node

The router-capable nodes also have the capability to be software defined as the high-layer fully functioned router nodes or reduced-function leaf nodes via the control packets received through the gateways. Once elected as routers, they can transfer information from other leaf nodes to upper-layer routers or active gateways using multi-hop communication protocol. They are presumed not to have direct communication with the UAV. Their roles as routers/leaf nodes are assigned by the UAV in the notification phase.

c) Gateway-Capable node

The gateway-capable nodes can be software defined to operate as gateways or dropped to lower-level routers and leaf nodes. Depending on their updated softwarized roles and their distributions in random positions within the ground network, they can present a variety of network topologies such as tree or star ones. They also determine the UAV fuzzy path. Their roles have been notified by the drone during the notification phase. Once selected as gateways for a given network of sensors, they are responsible for collecting the data coming from the lower-layer nodes and instantaneously forwarding that to the drone during the data collection post-orchestration phase. The spatial arrangement in a large-scale network distribution may include more than one node defined as a gateway. The presence of multiple gateways on a large-scale network facilitates the prevention of rapid energy draining of gateway-configured nodes and improves reliability by mitigating the risk of the failure of a single gateway. It also provides a broader range of UAV fuzzy path where the route is dynamically adjusted to benefit either the drone energy efficiency or the ground network’s energy consumption balancing.

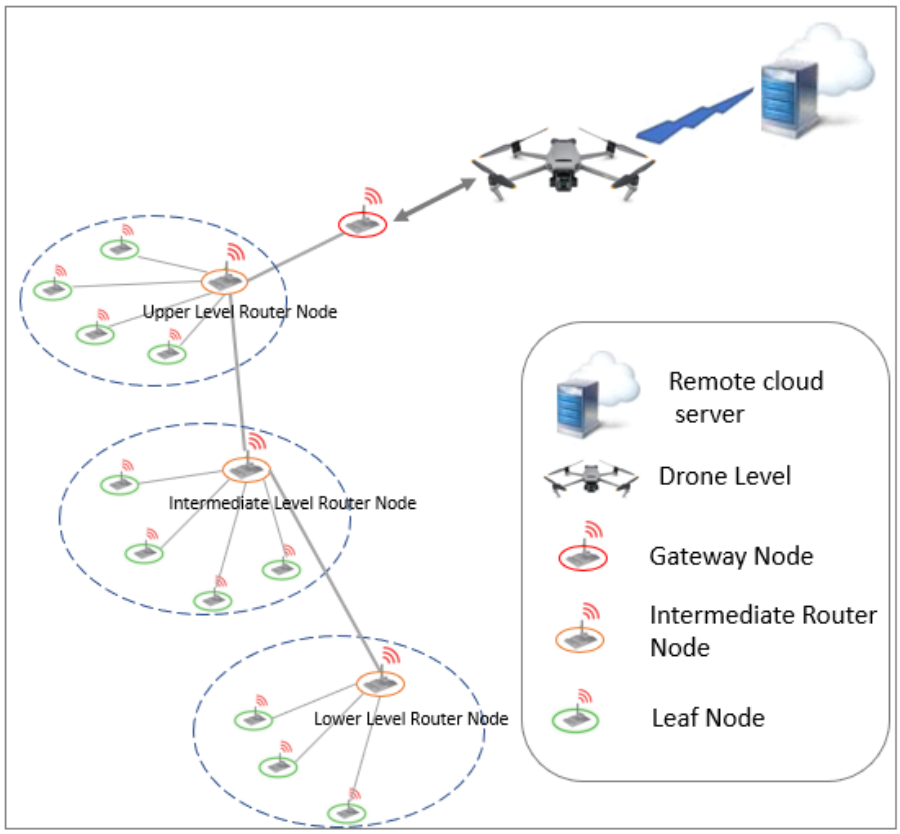


Figure 4-9 Topological arrangement for WSN data gathering.

d) Drone-level

The UAV serves as an access point to the remote cloud server via acting as an upper-level router between the gateways and the cloud platform (for both upstream flow of control data of the topological setting phase relevant to available regional nodes and the normal operational process of data collection post-orchestration phase and downstream flow of configuration of stations). Once the ground networks are defined, the UAV can resume regular ground sensing data collection.

e) Cloud-level computation

The cloud platform is the central station for post-processing real-world data passed by drone and virtualising network stations. Also, the UAV transfers the status of the ground nodes capability data to the cloud where the decisions for gateway and router elections have been made by utilising a proposed fitness model. Once the decisions have been made, the UAV fuzzy region as well as the elected gateways and routers for each group are assigned to each node's ID within the remote cloud server. Later, the decisions return

to the elected routers and gateways as well as their specified leaf-nodes through the notification messages to orchestrate the ground network configuration.

4.5.2 Network Topological Orchestration Strategies

The communication among the UAV and ground in the proposed SDWSN is based on a set of control/sensing data information messages that needs to be passed over the cloud to either get the network configured during the topological pre-orchestration scanning phase or gather the sensing data through data collection post-orchestration phase. The communication among the three acting sub-units which involves in both topological configuration planning and data collection phases aims at:

1. Collecting the control data by the UAV on available gateways and the relevant nodes that can access them and passing the control data to the Cloud.
2. Cloud level analysis of the collected control data and identifying the elected gateways and related network structure and operational parameters.
3. Passing the outcomes of the analysis from cloud to the ground networks within control data information offering the updated ground networks set up.
4. Collecting the ground sensing data through the data collection post-orchestration phase as a subsequent phase running the UAV movement within the designed updated fuzzy route.

4.5.3 Introduction to A Generic Model for Ground Network Energy Consumption

The proposed cost-effective energy model for evaluating the communication cost in both SDWSN topological scanning pre-orchestration and process data collection post-orchestration phases has been calculated in (4-8):

$$Cost_{total} = Cost_{Ground-network} + Cost_{air-to-Ground} \tag{4-8}$$

in which the overall communication cost equals with the aggregation of communication costs of both sensors on the ground and access points ground communication with UAV. The amount of each network components' power consumption for enabling the ground network communication is expressed in (4-9):

$$P_T = P_{Tx} + P_{Rx} + P_{LPM} + P_{Idle} \tag{4-9}$$

that means the average power consumption of each node is the summation of the average power consumption of node in four modes: Idle mode, Low power mode (LPM), receiving and transmitting modes. The Idle mode (P_{Idle}) is activated whenever the node is listening (the time interval that the CPU is non-active prior to the radio transmitter or receiver gets active). LPM mode (P_{LPM}) is activated when the sensor node goes to low power mode. Rx mode (P_{Rx}) is activated in the radio receive mode and finally Tx mode (P_{Tx}) is activated in transmission mode. Further explanations on power modes are available in Chapter 6.

Sensor nodes operate in either active mode or sleep mode. The ratio of the time spent in active mode to a total data period is defined as duty cycle. In general, sensors consume energy mainly in data receiving and transmitting, and idle listening when they are in active mode.

The proposed energy model uses the Contiki powertracker to measure the time intervals that each node spends in these four modes [104]. Hence, the overall energy consumption per node can be calculated considering the equivalent consumed energy for these intervals as calculated in (4-10):

$$Cost = \sum_{i=1}^{4} P_i \times T_i \tag{4-10}$$

where P_i represents the value of consumed power within each power mode, T_i is the time spent during a specific mode i. The power calculation analysis has been conducted based on the information explored from CC2538 datasheet for values of current usages, once the module is in active receiving, transmitting, idle and low power modes as shown in Table 4-1.

Table 4-1 Power parameters for Cooja mote based on [105].

Active-Mode TX current consumption	24 mA
Active-Mode RX current consumption	20 mA
Idle Mode	13 mA
Low Power Mode current consumption	0.6 mA
Supply-Voltage Range	2-3.6 v

For enabling the air-to-Ground communication among the UAV and gateway nodes, we use the radio model [106], for modelling the energy consumptions of transmitting and receiving of data for b bits in each terminal as expressed in (4-11) and (4-12).

$$E_t = \begin{cases} bE_{elec} + b\varepsilon_{fs}d^2 & d \leq d_0 \\ bE_{elec} + b\varepsilon_{amp}d^4 & d > d_0 \end{cases} \quad (4\text{-}11)$$

$$E_r = bE_{elec} \quad (4\text{-}12)$$

E_{elec} stands for transmitting circuit loss and d_0 is the threshold distance. ε_{fs} and ε_{amp} are the energy for power amplification in the free space channel model and multipath fading channel model respectively. The number of transmitted/received bits for each node is denoted by b. It is clear from (4-11) and (4-12) that the consumed energies for transmitter E_t and receiver E_r are highly dependent on the received/disseminated bits as b and, the instantaneous distance among the transmitter and receiver as d. As discussed in Section 5, the same energy model is implemented in CupCarbon for calculating the energy consumption of air-to-Ground communication. We presume that the transmission distance d during the communication among the UAV and each gateway is less than the threshold distance d_0, and the free space channel model is adopted accordingly. As discussed in Section 5, the overall communication cost in both SDWSN topological pre-orchestration scanning and data collection post-orchestration phases have been analysed based on various distribution and density factors of ground network.

4.5.4 Development of Communication Dialogues among The Network Entities based on The Proposed Packet Frame Designs in SDWSN

The communication dialogue among the network components is based on definition of control/sensing data information messages passed over the cloud to either configure the network components during the scanning pre-orchestration phase or gather the sensing data through data collection post-orchestration phase. The sequence diagram of the messages' transmission within the control and data flow phases has been depicted in Figure 4-10. The communication messages transaction among each network component has been designed using the following sequences:

Once the gateway nodes receive M_{Hello_msg} (message number one in Figure 4-10) from the UAV, they pass the message to the routers and lower-level leaf nodes to inform about the presence of the UAV. Then the nodes switch their transmitter on to broadcast their initial predefined functionalities (Leaf node, potential router, or potential gateway), along with the sensor's position, the magnitude of $RSSI$, the level of consumed energy, maximum capacity, the current number of connected nodes and current level which are defined criteria on identifying the fitness model election over cloud computation (denoted as M_{C_UAV} within message number two in Figure 4-10). The proposed control packet frame format for M_{C_UAV} in UAV-Ground communication is suggested based on Figure 4-11. The first message field of the payload within the proposed frame involves the transferring time slot along with the ID of the corresponding Router/Gateway. The next three packet fields involve 'maximum capacity', 'current number of connected nodes' and 'current level' reflecting the updated ground network structure. The 'maximum capacity' packet field places a boundary on the maximum number of supported nodes (including leaf or lower-level router nodes) of each router/gateway node. The 'current number of connected nodes' field represents the current number of attached nodes (including leaf/lower router nodes) to the router/gateway. The 'current level' message field stands for the level of the corresponding node with respect to the higher-level gateway. Finally, the last three fields consist of three operational factors associated with gateway/router own control data including 'RSSI', 'consumed energy' and 'relative location' which are other key factors in fitness model decision making. The length of data frame body for M_{C_UAV} is highly dependent on the number of ground network components transferring their control data to the UAV in the topological pre-orchestration scanning phase, which is assumed variable here.

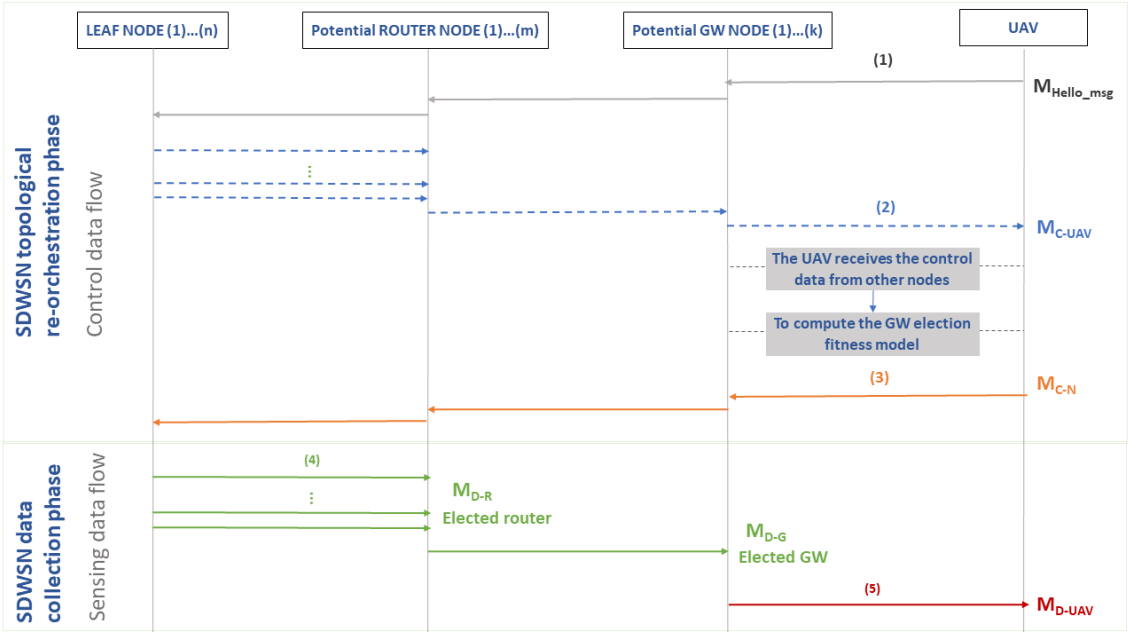


Figure 4-10 Sequence diagram illustrating the messages transmission through control and data information dissemination phases.

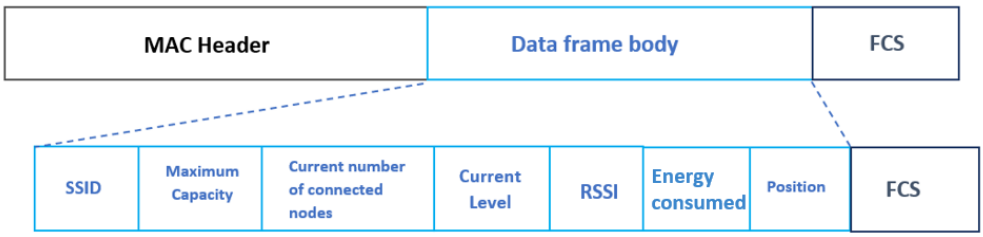


Figure 4-11 The control packet frame design in topological pre-orchestration scanning phase on ground and UAV-Ground network communications.

Then, once the election decision has been made within the remote cloud server, the UAV serves to notify the updated elected gateways and routers on their new roles by disseminating M_{C-N} messages (messages number three in Figure 4-10). The potential routers and gateways that are not selected as router and gateway have been switched over to the role of reduced functioned leaf nodes (M_{C-N} message notifies the updated function definition plus information about the SSID of the lower and upper-level nodes).

The suggested data packet frame for M_{C-N} on notifying on the updated ground functionalities is designed in Figure 4-12. Next to the 'SSID' segment of transferred packet field in the notification data frame, the 'functionality' packet field represents the updated functionality of the potential node (either Gateway, Router or Leaf node). This

packet field consists of 2 octets creating multiple functional messages (expressed in Table 4-2) by passing a binary code to each network entity to define the updated functionality. Furthermore, within the 'SSID Lower-Level' segments, the node IDs of involved leaf nodes or lower routers have been assigned. The 'SSID Higher-Level' packet specifies the ID of a higher-level node either a Gateway or a router. These two segments in the packet frame design reflect the number of updated upper level and lower-level connections per node. The designed packet frames are relayed to the potential router/gateway nodes in topological pre-orchestration scanning phase to redefine the functionality of the potential nodes plus to orchestrate the entire network structure based on the updated connections among nodes. For assigning the lower-level nodes to the higher-level ones (such as assigning a number of leaf nodes to a specified router), the Euclidean distance among the leaf nodes and the router has been considered for updated groups' formation. The length of data frame body for M_{C-N} is highly dependent on the number of potential routers/gateways distributed in the network, which is considered as variable here.

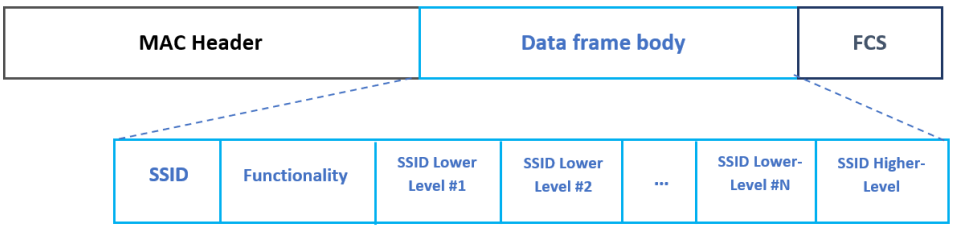


Figure 4-12 Packet frame design for notification messages coming from the UAV to the Ground (Routers and Gateways).

Table 4-2 The transmitted code within the functionality field of notification message frame traversing to the potential Router/Gateway nodes.

Transmitted code of functionality field	**Transmitted defined functionality**
[00]	Leaf Node
[01]	Router L1
[10]	Router L2
[11]	Gateway

Finally, in the software-defined data collection phase, once the UAV returns to the elected gateways, the normal flow of sensed data from the updated leaf nodes relaying to the elected router and gateway (represented as messages number 4 in Figure 4-10) resumes

and passes over the UAV (represented as message number 5 in Figure 4-10). The proposed data packet frame design on the ground and air-to-Ground network communications traversing from leaf-nodes receiving on the UAV in the data collection post-orchestration phase has been designed in Figure 4-13. The first four message fields of the proposed M_{D-R}, M_{D-G} and M_{D-UAV} messages involve the same required fields indicating the current configured state of each node including Leaf, Router and Gateway. The remainder of message fields consist of sensing data gathered from each leaf-node arranged to pass from gateway to the UAV within one accumulated large frame of multiple sensing data particularly on M_{D-UAV} message unit (expressed as message number 5 in Figure 4-10). The length of data payload for these three messages M_{D-R} , M_{D-G} and M_{D-UAV} is highly dependent on the number of leaf nodes distributed in the network, which is considered as variable in the proposed design.

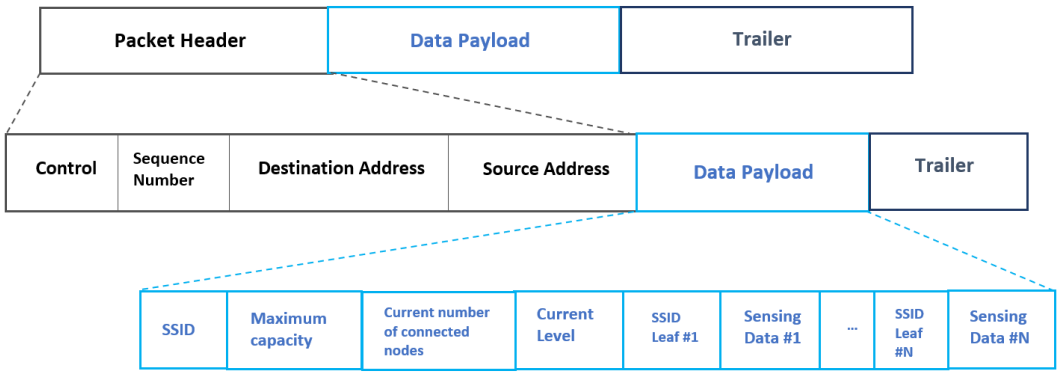


Figure 4-13 The data packet frame design in data capturing post-orchestration phase for ground and air-to-Ground network communications.

Fitness election model

The fitness model computation is presumed to be executed in the cloud environment enabling the updated structure of a ground network organisation. The proposed parameters in the fitness model election process for the involved potential gateways are briefly expressed as:

- **Link quality based on Radio Signal Strength Intensity (RSSI):**

The link quality between a UAV and its neighbour gateways is obtained by using the information of received signal strength intensity ($RSSI$) of received packets. The link quality between UAV and potential gateways, LQ can be expressed as (4-13):

$$LQ_{GWi-UAV} = \left(\sum_{k=1}^{N_{rssi}} \frac{R_k^2}{N_{rssi}}\right) - \left(\sum_{k=1}^{N_{rssi}} \frac{R_k}{N_{rssi}}\right)^2 \quad (4\text{-}13)$$

in which N_{rssi} is the total number of $RSSI$ samples received on the UAV from each gateway-capable and R_k is the $RSSI$ value of the k-th sample.

- **Energy consumption factor per node:**

The overall energy consumption per gateway-capable node $cost_i$ is another key factor in election process which is expressed in (4-10).

- **Capacity factor per node:**

The capacity factor per node H_i is another parameter in the proposed fitness model definition which is expressed as (4-14):

$$H_i = 1 - \left(\frac{M_i}{Q_{max_i}}\right) \quad (4\text{-}14)$$

where M_i is the current number of connected nodes to the gateway node ith and Q_{max_i} is the maximum capacity of the gateway node ith. These parameters have been uploaded to the cloud by the UAV via the control packet frame design shown in Figure 4-11. Note that once the number of connected nodes to the gateway equals with the predefined maximum capacity, the capacity factor equals with 0. This means that the gateway is connected to its neighbours' routers and leaf nodes with its full capacity, and it consumes higher energy than other gateways. This implies that this gateway should have a lower preference over other gateways with higher capacity factor.

Thus, the fitness model has been defined as (4-15):

$$W\_election_i = \alpha \times LQ_{GWi-UAV} + \beta \times Cost^{-1}{}_i + \zeta \times H_i \quad (4\text{-}15)$$

Wherein $LQ_{GWi-UAV}$ is the link quality factor of the gateway-capable node ith received on the UAV, $Cost_i$ is the accumulative energy cost of gateway-capable node ith. Higher the LQ and lower the $Cost$ parameters are, the higher likelihood of electing as gateway node, H_i is the capacity factor of the gateway-capable node ith. According to the proposed fitness model, the ground network should orchestrate in favour of the gateway-capable nodes election with better link quality, lower energy consumption and higher capacities, α , β and ζ denote three weights assigned to the three parameters based on their priorities

in the updated network structure. Following the execution of election process computation in the cloud, the gateway-capable node with the higher value of $W\_election_i$ will be elected as the updated gateway node and those that are not elected as gateways will be dropped down to leaf node functionalities. The updated ground network architecture will be restructured based on the locations of the current gateway nodes.

4.6 Air-to-Ground Communication

The high mobility as well as the physical constraints which are imposed by the UAV may cause some degradation to the performance of the wireless links on air-to-Ground communication. In addition, the three critical flight dynamics parameters known as pitch, roll and yaw may also incur the same degradation and have large implications on the communication network. Herein, an effective communication mechanism amongst the UAV and ground network is designed to control the channel access and at the same time has the capability to be adjusted based on the changes in the aircraft position.

Mainly, the proposed air-to-Ground communication design can be done based on two approaches:

- Air-to-Ground communication with each individual sensor nodes
- Air-to-Ground communication with each group representative nodes as cluster heads

The second approach enables the concept of fuzzy path and is considered as an energy efficient approach for UAV path design, while the first approach allows the UAV to serve more sensor nodes at each operational round.

4.6.1 Communication Scheduling through Communication Window of Time Definition

One way of improving the efficiency of the interaction between the UAV and ground SNs is through facilitating a UAV moving window of connectivity along a large space. This window considers dynamically allocating certain sensor nodes/gateway-capable nodes that are within the UAV's prespecified communication range. Note that with the proposed air-to-Ground connectivity designs, the UAV could pass through group representatives within the fuzzy range benefiting from a larger movable communication window of connectivity to connect to SNs either indirectly or directly (depending on the air-to-Ground approach that is selected). Hence, to initialize the formulation of an efficient

UAV-Ground interaction, the proposed initial trajectory is first modelled in the classical TSP, which has been manipulated as 'Centroid-based trajectory enhancement approach (CTEA)'. The CTEA strikes an effective performance by computing UAV visit orders of centroids and their locations, dividing all SNs into groups by utilising K-Means clustering scheme. The nodes of interest communicate with the UAV while traveling along the field. To this end, the opportunistic time of connectivity is designed for the drone trajectory based on the determined path within fuzzy region. The step process of communication scheduling among the UAV and ground using the proposed connectivity window of time has been sketched in Figure 4-14. Based on the designed step processes of the proposed communication scheduling, the model has been categorized into the major sub processes in which firstly, the initial UAV path has been designed based on CTEA within the fuzzy region to enable sketching the relaxed path. Following the path relaxation over one of the proposed methods that already mentioned in Section 4.4, the communication on UAV-Ground connectivity has been established based on the proposed communication window of time. Finally, an efficient and discrete time data gathering scheduling based on the received RSSI measurements has been highlighted regarding the outcomes of communication window and UAV smooth path designs.

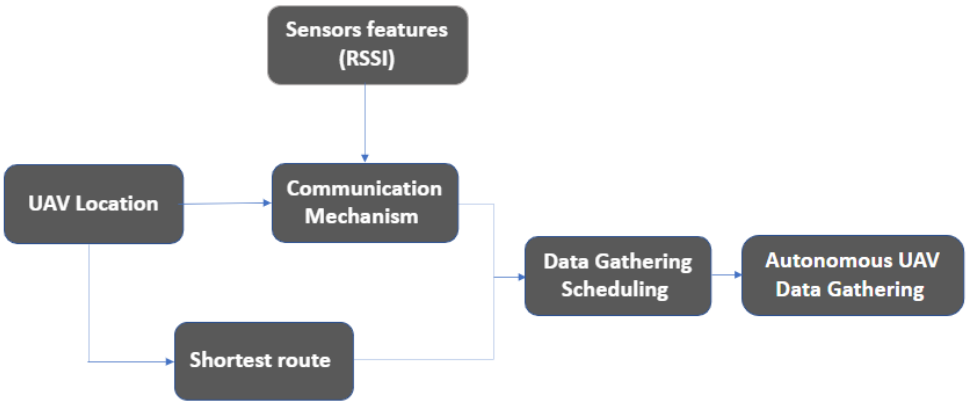


Figure 4-14 Illustration of the proposed processes for UAV-Ground communication scheduling.

4.6.2 Priority-based Communication with Individual Sensor Nodes

In this section, a reliable and time-discrete communication scheduling mechanism based on obtained communication window of time providing the virtualisation capabilities among the UAV and individual SNs while improving network performance has been proposed. Following the UAV entrance to the communication window of time per group, the impacted nodes communicate with the UAV while traversing along the ground. To

compute the communication window for each group, the minimum Euclidean distances among each step of the smooth path and the members of each group are obtained individually (point by point distances). Next, the steps of the path which are in the closest distances from the affected SNs have been specified. The entry point of the UAV's connectivity window for a group is the first UAV's location on its path where the UAV has minimal distance from the first SN, and the departure point is the last UAV's location that has minimal distance from the last SN under the same group (See Figure 4-15). Based on this figure, 'X' represents the centroid of each group while the red dots are the SNs belonging to a specified group. The first arrow is the entry point to the communication window of connectivity while the second arrow is the departed point. The distance between these two points is defined as the communication window of time for this specified group.

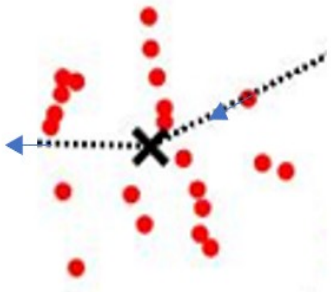


Figure 4-15 The proposed connectivity widow of time design for a group of SNs.

Hence, the UAV window of connectivity is identified as the time once the UAV is in between the entry point and the departed point that is calculated as (4-16):

$$T_{Connectivity} = T_{Departed} - T_{Entry} \tag{4-16}$$

The designed communication window of time, $T_{Connectivity}$ is highly dependent on the average speed of UAV V_{avg} as calculated in (4-17):

$$T_{Connectivity} = \frac{x_{connectivity}}{V_{avg}} \tag{4-17}$$

wherein $x_{connectivity} = x_{Departed} - x_{Entry}$ is a piece of trajectory with lowest distance from the affected SNs belonging to the same group. Note that the connectivity windows of time will be variant for each group depending on the density of that group.

Next, to enable time-discrete communication mechanism between affected SNs and the UAV within each communication window, the TDMA MAC protocol based on the time-discrete communication mechanism has been utilised due to the less packet drops performance based on different time slot allocation to each SN for communication interaction. In the proposed model, let specify a time varying communication scheduling variable $a_k[n]$ in which $a_k[n] = 1$, once UAV has a communication link with the SN and $a_k[n] = 0$ once the communication on the air-to-Ground link is not feasible. The proposed communication scheduling model is based on the distance between the impacted SN and the UAV which could be enhanced by the $RSSI$ measurements to revise the scheduling variable per time slot. This can be proposed based on applying a sensible constraint on the $RSSI$ measurements, $RSSI_{th}$ received on the UAV from each SN. Therefore, the scheduling variable is calculated as (4-18):

$$\begin{cases} a_k[n] = 1 & RSSI \geq RSSI_{th} \\ a_k[n] = 0 & otherwise \end{cases} \quad (4\text{-}18)$$

wherein node k is the interest node that initiate communication with the drone at the time slot n where the received $RSSI$ on the UAV is higher than a $RSSI_{th}$ constraint. Once the UAV is within each communication window of time and the condition (4-18) has been satisfied, the UAV will establish its communication with the SNs. Considering establishing a connection to the UAV, the affected SN that is within the closest distance from the UAV has been assigned a time slot to connect to the UAV. Hence, affected SN with RSSI signal below a particular threshold value on the UAV's receiver has not been successful to send its packet to the UAV correctly.

4.6.3 Air-to-Ground Communication Model based on UAV Connectivity with Group Representatives

This section emphasises a communication scheduling mechanism based on the obtained communication window of time among the UAV and gateway-capable nodes. The prioritisation for nominating the most favourable gateway nodes out of the potential gateways is employed based on the fitness model that is proposed in Sub-section 4.5.4. Once the appropriate gateway has been chosen for each group and the updated ground structure has been orchestrated, the window of connectivity for the communication among the UAV and the elected gateway is formulated. The entry point of the UAV's connectivity window for each group is the first UAV's position on its trajectory where the UAV receives the RSSI variant equal or higher than RSSI constraint from the elected

gateway, and the departure point is the last UAV's position that the received RSSI variant is higher than the RSSI threshold. Hence, UAV window of connectivity is defined as the time once the UAV is in between the entry point and the departed point.

Once the UAV is within the defined communication window of connectivity, the UAV initiates its communication with the gateway to gather the already buffered data in gateway. We presume that the gateways use a consistent transaction of communication considering acknowledgements (Ack) and message retransmissions to operate the communication on the air-to-Ground network. The UAV initiates the connectivity with the gateway node of interest within its window of time and sequentially traverses over each group during numerous connectivity windows of time. Once the UAV is within the communication window of affected gateway node, it initiates its presence by transmitting a hello packet to wake the gateway up, then a designed control/data packet consisting of the $RSSI$ measurements is forwarded from the gateway to the UAV (expressed in Sub-section 4.5.4). If the UAV receives the $RSSI$ packet frame properly with the $RSSI$ measurement upper than the $RSSI_{th}$, it sends an Ack to the gateway. Whereas, once No Ack received on the gateway with the $RSSI$ measurement lower than the $RSSI_{th}$, it transmits the data/control packet frame multiple times to ensure that the packet is received in the UAV buffer by receiving the Ack properly in the gateway. The UAV is presumed to be in the given communication window associated with the identified gateway and the affected gateway transmits its buffered data to the UAV afterwards. The sequence diagram of UAV-Ground communication by employing gateways is shown in Figure 4-16. The proposed communication window of connectivity is highly dependent on the average UAV speed, herein, with raising the average UAV's speed, the connectivity window among the UAV and gateways decreases. This results in the UAV to depart the communication window prior to ensuring the successful receiving of the overall data/control packets buffered in gateways. Hence, the unsuccessful transmitted packets are assumed as packet loss in the UAV operational travel. In the Section 5, the effect of the UAV speed on the percentage of served SNs is demonstrated in detail. The proposed air-to-Ground communication network among the UAV and gateways is described in detail in Algorithm 2.

Algorithm 2 UAV-Ground communication strategy

1: **Inputs:** Communication scheduling variable $a_k[n]$, the received $RSSI$ from each gateway on the UAV receiver $RSSI_{received}$. Suppose the percentage of distributed Gateway-capable nodes meets the condition $D_{gc} \leq D_{gc}(th)$

2: **Initialize:** apply the selected smooth path based on Algorithm (1) within fuzzy region, assign multiple polling time channel access to each gateway

3: **for** each gateway-capable node nominated N **do**

4: **if** $RSSI_{received} \geq RSSI_{th}$ **then**

5: $a_k[n] = 1$ and calculate the connectivity window based on eq. (7)

6: **end if**

7: **if** (the UAV presence packet is received in the ith gateway receiver & $RSSI_{received} \geq RSSI_{th}$) **then**

8: the UAV is within the ith communication window associated with ith gateway and the gateway initiates its buffered data to the UAV

9: **end if**

10: **end for**

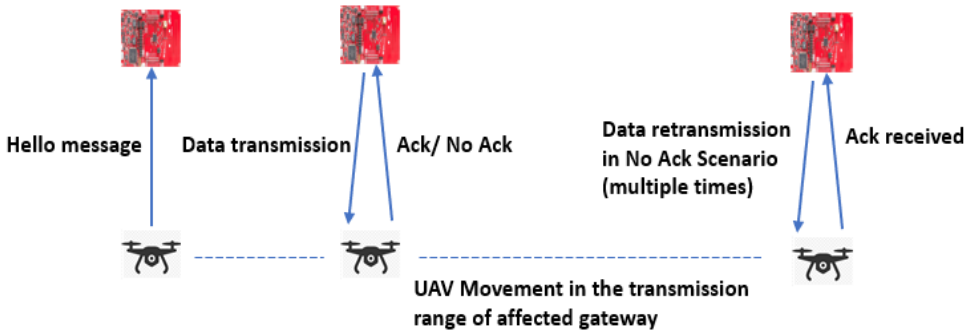


Figure 4-16 The sequence diagram of UAV-ground communication employing gateways.

As discussed in Section 5, the proposed communication method offers the close to actual scenarios on the air-to-Ground data communication along with maintaining a better network performance.

4.7 Designing an Analytical Formulation for Optimisation Problem

Following with multiple scenarios assumed in these three subtopics viz. UAV energy efficient path planning, ground network communication and air-to-ground communication, the output performance of the proposed approach includes the percentage of served sensor nodes, ground network energy consumption, and average UAV energy consumption. It is obvious that there is a trade-off between UAV propulsion energy consumption and ground network energy cost. As will be discussed in Chapter 5, while the proposed crisp path can offer an energy efficient path from the UAV cost point of view and serve a higher percentage of ground sensor nodes, it suggests an energy inefficient data gathering model from ground SNs perspective. The Bezier curve path design although results in UAV energy consumption degradation per mission and lowers the percentage of served sensor nodes, it benefits the ground network formation in terms of energy efficiency. Hence, it is required to define an optimisation problem to solve the optimal solution considering jointly minimising the UAV propulsion energy usage and ground SNs energy consumption while maximising the packet delivery to the UAV.

To this end, to enhance the packet delivery in the air-to-Ground communication, a statistical model for modelling the communication throughput amongst the UAV and SNs considering LoS communications needs to be developed. The air-to-Ground connectivity model for occasional link blockage due to NLoS links is not presumed in this thesis. Hence, the total amount of information bits transferred to the UAV over the duration T is a function of UAV trajectory expressed as [42]:

$$\bar{R}(q(t))_{air-to-Ground} = \int_0^T B \log_2\left(1 + \frac{\gamma_0}{H^2 + \|q(t)\|^2}\right) \tag{4-19}$$

where B stands for the channel bandwidth, γ_0 is the reference received signal-to-noise ratio (SNR) at $d_0 = 1m$. H is the altitude of the UAV while flying over the ground SNs.

To define the approximation optimisation problem, a cost function of multiple parameters is required to be specified as:

$$\underset{\substack{q(t) \\ s.t.\ List\ of\ Constraints}}{Minimise} \quad \frac{\bar{E}(q(t))_{UAV-propulsion} + Cost_{Total}}{\bar{R}(q(t))_{air-to-Ground}} \tag{4-20}$$

Where $q(t)$ is the UAV path design. $\bar{E}(q(t))_{UAV-propulsion}$ is the UAV propulsion power consumption obtained from (4-3), $\bar{R}(q(t))_{air-to-Ground}$ is the communication throughput obtained from (4-19) and $Cost_{Total}$ is the ground energy cost emerged from (4-8). According to our assumptions, ($Cost_{Ground-network}$) within (4-8) is static and not dependant on the location of the UAV, while ($Cost_{air-to-Ground}$) is highly dependent on the distance between the UAV and SNs and as a result, the locations of the UAV based on (4-11). Hence, the impact of the ground SNs energy consumption while communication with each other ($Cost_{Ground-network}$) is disregarded in our optimisation problem formulation by just taking the UAV-Ground communication energy usage ($Cost_{air-to-Ground}$) into account.

Identifying the constraint for the optimisation problem is dependent on the definition of the approximation for the proposed scenario. The outcome of optimisation problem should probably be an optimal and energy efficient network formation that can offer an energy saving UAV path design and at the same time supporting the requirement of maximised communication throughput and mitigating the high values of ground energy cost. By discretizing the time horizon T into $N + 2$ slots with step size δ_t , i.e., $t = n\delta_t$ $n = 0, 1, \cdots, N + 1$, the UAV's trajectory $q(t)$ can be well characterized by the discrete-time UAV location $q[n] = q(n\delta_t)$, the velocity $v[n] = v(n\delta_t)$, as well as the acceleration $a[n] = a(n\delta_t)$.

The constraints that should be satisfied on the UAV path optimisation problem are UAV initial/final location and velocity, the minimum/maximum speed and acceleration and minimum throughput requirement on the UAV-Ground connectivity. The list of the constraints in the UAV path optimisation problem is expressed in Table 4-3.

Table 3-3 the list of constraints in solving optimisation problem.

Parameter	**Constraint**	**The necessity of using the constraint**
UAV initial/final position	$q[1] = q[M]$	The UAV initial and final locations should be the same (the UAV needs to get back to the initial location after the horizon time T).
UAV initial/final velocity	$v[1] = v[M]$	The UAV initial/final speed should be the same.
UAV velocity in each time slot	$V_{min} \leq \|v[n]\| \leq V_{max}$	The UAV's velocity should be restricted between two amounts during the whole trip.
UAV acceleration per time slot	$\|a[n]\| \leq a_{max}$	The UAV's acceleration should not be passed over a maximum threshold during the whole trip.
Communication scheduling	$\alpha_k[n] \in \{0,1\}$ $\sum_{k=1}^{M} \alpha_k[n] \leq 1$	In each time slot, only one sensor node is scheduled for transmission based on the TDMA MAC protocol.
Communication throughput	$\sum_{n=1}^{M} \alpha_k[n] R_{k,n} \geq \boldsymbol{\eta}$	The throughput requirement for all sensor nodes should be more than a specific threshold $\boldsymbol{\eta}$

Since some of the constraints in Table 4-3 such as the minimum UAV speed constraint and communication throughput constraint are not convex, solving approximation optimisation problem in (4-20) is a challenging task due to the non-convex problem formulation. There are various mathematical algorithms to search the optimal results for non-convex problems. Herein, a specific mathematical solution for finding the optimal UAV path, velocity and acceleration with maximised communication throughput and minimised ground energy consumption is required. Sequential convex approximation (SCA) technique is chosen to solve the optimisation problem using slack variables to convert the problem into linear programming which is solvable by CVX MATLAB. The original non-convex problem can also be solved by iteratively optimizing (4-20) with the local point$\{q_i[n], v_i[n]\}$ updated in each iteration.

4.8 Conclusion

This chapter delved into the concept that was developed in order to design a comprehensive strategy for UAV-based data collection from wireless sensor networks (WSNs). The proposed method aimed to enable energy-efficient UAV-aided WSN data collection by allowing mutual adaptability between UAV relaxed path design and ground network topological arrangements. The above stated concept of creating what we referred to as a ‘Fuzzy Path’ utilized the ability to software redefine the groups cluster heads or network topology to align with the optimal global UAV flight path. This section also investigated the emerging role of SDWSN in UAV path planning and air-to-Ground communication, with the goal of improving network performance and UAV energy efficiency through dynamic orchestration of wireless ground sensor nodes.

Chapter 5 Modelling and Simulation of UAV-enabled SDWSN Data Collection

5.1 Introduction

This Chapter presents the modelling of the UAV-assisted Software Defined Wireless Sensor Network (SDWSN) data gathering ideology proposed in Chapter 4. This Chapter gives a detailed description of the modelling of key components of the proposed concept. The model implementation and the testing of the individual components are presented, followed by the results from the modelling and simulation activities.

In this regard, the UAV fuzzy path model and relevant components of the UAV energy efficiency are designed and analysed using MATLAB and SITL Mission Planner. These two software tools offer analytical environments for UAV path design and validation. Also, the communication network models including Ground and air-to-Ground communications are designed and analysed using Contiki-Cooja and CupCarbon network simulators. As explained in Chapter 3, Contiki-Cooja and CupCarbon are both effective discrete event-based network simulators. Herein, the key parameters of the UAV path model, air-to-Ground scheduling approach and software defined ground network design are gained through MATLAB to feed them into the SITL Mission Planner and CupCarbon/Contiki-Cooja in order to evaluate the performances involved in space coverage and communication network, respectively.

The ‘UAV fuzzy path’ concept is initially implemented in MATLAB to understand and investigate the impact of data-capturing dependent parameters on the proposed model. MATLAB generates visual outputs that can be fed into SITL Mission Planner in order to validate and debug the proposed paths. Moreover, Contiki-Cooja and CupCarbon are used for implementation of the ground and UAV-Ground communication networks.

5.2 Modelling and Simulation Phases

The modelling of UAV-enabled SDWSN data collection is separated into several phases. Figure 5-1 shows an overview diagram of the modules involved in the simulation planning phase following with the illustrations of relations among their outputs and inputs. The simulation analysis is based on four simulation tools including MATLAB, CupCarbon, Contiki-Cooja and SITL Mission Planner. As shown in this figure, the output

of one simulation tool can trigger the next level of simulation through the use of other simulation software. For instance, MATLAB output parameters can be applied as inputs for Contiki-Cooja, CupCarbon and SITL Mission Planner facilitating the network orchestration and smooth path design models at the same time. Each software module simulation is based on a sequence of multiple stages per module. The following section contains a detailed description of each task on each sub-module.

The proposed simulation model results in reduced ground sensor node energy consumption, and overall packet loss, and jointly minimised UAV energy expenditure during the smooth path design.

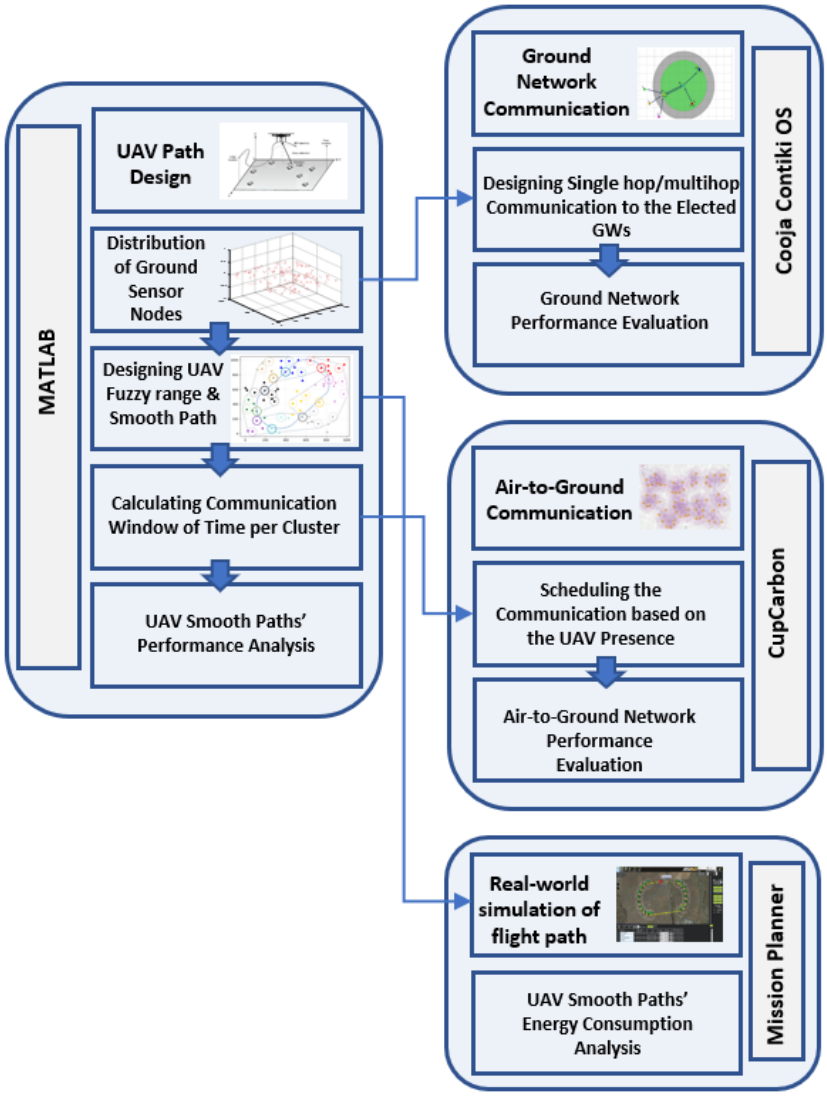


Figure 5-1 Collaboration among tools for implementation of the proposed model.

5.3 Key Parameters Involved in Modelling

5.3.1 MATLAB

The simulation software used for analytical design and evaluation of UAV path planning is MATLAB 2020b in which the UAV fuzzy region, UAV relaxed paths and the air-to-Ground window of connectivity for the spatially deployed SNs are identified. The performances including length of the path, mission time, instant speed and acceleration of the UAV and UAV energy efficiency have been assessed in MATLAB. Within MATLAB module, the first simulation module (See Figure 5-1) is through the spatially distribution of ground sensor nodes considering two given variants: the distribution of gateway capable nodes and their density spread factor. The outputs of this module can be employed for ground network communication analysis in Contiki-Cooja as the locations of dispersed sensor nodes have been transferred one by one from this module of MATLAB to the Contiki-Cooja network simulator. Then, within the second component of testbed simulation in MATLAB, the UAV fuzzy route and UAV flight smooth path are designed considering the gateway capable nodes placement on the ground. Following that, the communication window of connectivity enabling interaction among the UAV and the elected gateways is sketched as part of the third process. The outputs of these two modules (UAV crisp path design and communication window of connectivity design) support the air-to-Ground communication simulation in CupCarbon. The UAV crisp path design can also be validated in SITL Mission Planner SITL based on real world scenario visualisation. Finally, the UAV performance evaluation mainly on the UAV energy expenditure, length of the path and velocity of UAV is worked out here.

Figure 5-2 depicts the parameters involved in interrelations among the software tools as inputs and outputs. Initially, various scenarios of sensor nodes deployment could be simulated in MATLAB based on given densities and distributions of sensor nodes and then fuzzy route is aligned based on the predefined locations of gateway-capable nodes. As shown in Figure 5-2, two main parameters in defining the proposed model in MATLAB are the given density spread factor and the distribution of gateway-capable nodes among the whole ground sensors that identify the distribution of sensor nodes in the testbed design. Following the design of the fuzzy range and smooth path in MATLAB, four output parameters including UAV flight time, length and shape, instantaneous UAV’s velocity and acceleration and elected ground gateways from fitness model

analysis will be outputted and fed into CupCarbon, Contiki-Cooja and SITL Mission Planner.

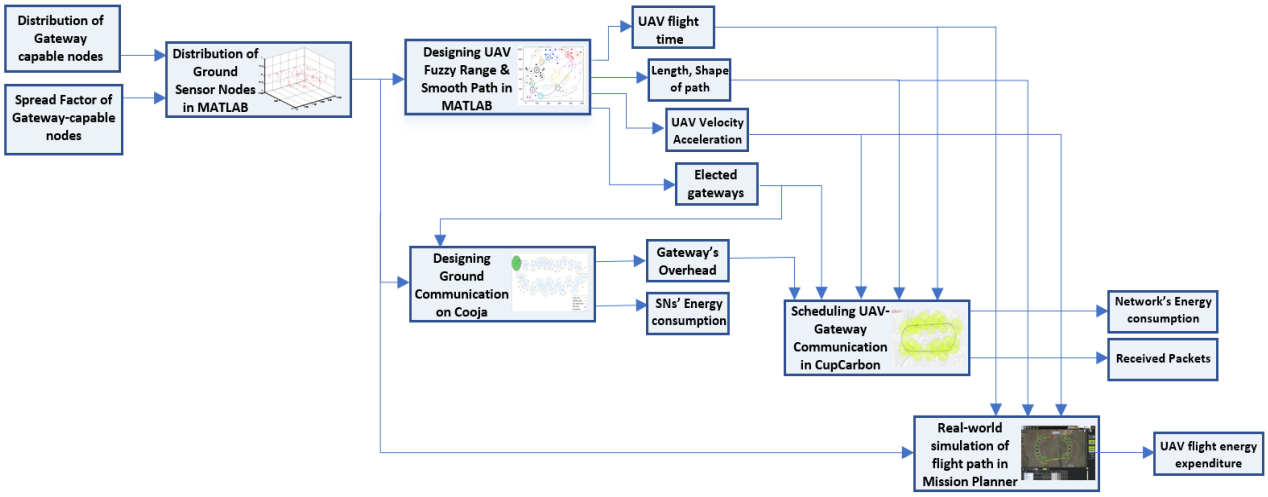


Figure 5-2 Simulation software correlations among outputs and inputs.

5.3.2 The Contiki-Cooja Tool

The network simulation software used to model the communication on the ground sensor network is Contiki-Cooja (as discussed in Chapter 3). In this study, Contiki-Cooja consists of multiple modules to analyse the proposed model. In the first module (See Figure 5-1), a test is conducted for ground network that has got multiple functionalities (leaf, router, and gateway) where the topological structure mode can be dynamically configured based on star, single hop, multi-hop networks as a compound network topology. The distribution of ground sensor nodes in Contiki-Cooja are aligned with that in MATLAB. Hence, the ground network is structured in Contiki-Cooja considering various network formation scenarios depending on the election of gateway nodes out of the predefined gateway-capable nodes process. The sensor nodes consumed energy vs. packet delivery rate performances are examined for the proposed model assessment. As shown in Figure 5-2, the accumulated buffered overhead in each gateway is an input for the next simulation implementation phase in evaluating the air-to-Ground connectivity performance in CupCarbon.

5.3.3 CupCarbon

CupCarbon is the network simulation software tool used in this thesis to simulate communication interactions via air-to-Ground connectivity (as discussed in Chapter 3).

As discussed in Chapter 3, CupCarbon enables visually testing the WSN topologies through a discrete event simulation of the conducted testbed and at the same time allows the design of mobility scenarios by simulation of mobiles such UAVs, hence, it is a well-suited tool for simulating data collection form distributed gateways utilising the UAV. The simulation step process is composed of two modules (See Figure 5-1). The outputs of modules two and three of MATLAB (deploying the elected gateways on the ground and UAV smooth path) facilitate the air-to-Ground data gathering model from the elected gateways in CupCarbon. The UAV path is presumed to have the same path and length following with similar distribution of gateway nodes as in MATLAB. The first step process in air-to-Ground communication simulation is through the scheduling of the UAV-Ground communication based on the UAV presence on the top of each gateway. Once the UAV is on the communication range of each selected gateway, the buffered data on each gateway from the ground network is uploaded to the UAV dependent on the UAV velocity and size of buffered data on each gateway (two designed parameters that are adjusted in MATLAB and Contiki-Cooja network simulator). Following that at the air-to-Ground testing evaluation module, as shown in Figure 5-2, the air-to-Ground performance analysis including packet delivery rate and sensor networks consumed energy for enabling communication among the UAV and gateways have been analysed.

5.3.4 SITL Mission Planner

SITL Mission Planner is the simulation tool used in this study for the energy efficiency validation of the proposed UAV path based on a real-world scenario. The SITL flight simulator embedded in Mission Planner is in charge of analysing the instant behaviour of the UAV flight path performance, including instant power and velocity variants. As represented in Figure 5-1, the MATLAB output of the UAV smooth path design module is used for validation in SITL Mission Planner. Multiple waypoints on the UAV mission plan are required to define the simulation outlines on SITL. Hence, the UAV path (either Bezier curve or crisp path) with the same length and curves outputted from MATLAB is identified in SITL Mission Planner. The UAV path performance is validated based on definition of the volumes of real time consumed current and speed for the entire path which are logged in SITL Mission Planner. The outcomes of path design implementation in SITL Mission Planner and MATLAB can be compared to justify the UAV energy behaviour for the entire path as well as characterise the key parameters required for the smooth path design.

5.4 Development of UAV Path Design

In this section, the simulation works conducted on the MATLAB and SITL Mission Planner are presented. In the simulation, the fuzzy path and associated smooth paths modelled in such a way that the ground network models represent real infrastructures in real network settings. The similar path design, which have been tested in MATLAB, is implemented using the SITL Mission Planner. Herein, the implementation of fuzzy path and the simulation setup for smooth paths are discussed in this Section. The simulation cases and associated design parameters of the study are also presented.

5.4.1 UAV Path Analytical Design Model Development

To begin with, the first stage of the model evaluation is defined by deploying the SNs distribution which is thoroughly based on the real-world non-uniformed random distribution of SNs. Herein, the non-uniform random distribution of ground nodes for $N = 100$ is generated firstly in MATLAB. The distribution of gateway-capable nodes for the proposed fuzzy region design is assumed as threshold value of $D_{gc}(th) = 30\%$ to offer sufficient fluidity for route design. Also, the values of initial Gateway Election factor ξ and Spread Factor of gateway-capable nodes σ_gc are considered as $1/2$ and $250\ m$ respectively. Next, $K - means$ clustering method is used to group the SNs based on Euclidean distances between the SNs. This algorithm is utilised for max iterations of 1000. Figure 5-3 demonstrates the distribution of SNs in relation to the predefined gateway-capable nodes, with their transmission ranges demonstrated by various coloured-radiuses belonging to different groups. Then, the initial trajectory is modelled by classical TSP which can be simulated based on the CTEA paradigm, shown in dotted blue line in Figure 5-3, in which the main visit orders of UAV trajectory are identified based on grouping of the SNs (discussed in Chapter 4). The UAV fuzzy path is then aligned on CTEA initial path taking the communication ranges of entire gateway-capable nodes into account. Herein, the proposed smooth path must be bounded within the fuzzy region. This method can improve the resilience and efficiency of the system due to the UAV path relaxation arising from the wider fuzzy range. Herein, the proposed topology organization eases the UAV path design by offering the efficient path planning within the proposed UAV path fuzzy region and allowing for the flexibility of optimal CH election within the fuzzy region. As shown in Figure 5-3, an example benefit of the UAV path relaxation is illustrated by the possible uniform crisp path that might be fitted on the fuzzy path and help with smoothing the flight path.

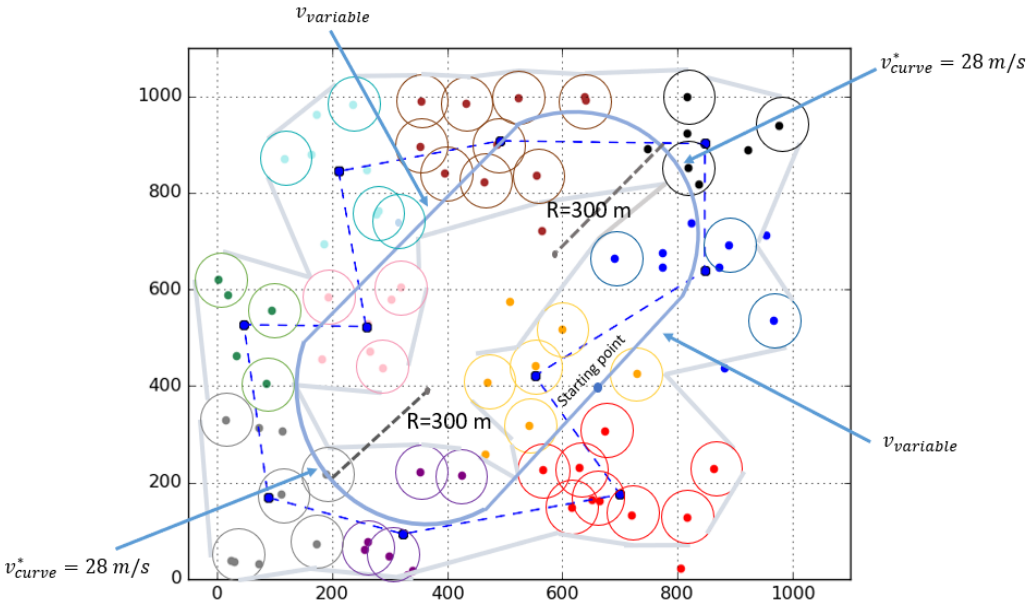


Figure 5-3 A design of UAV fuzzy range (the space in between the grey lines) and smooth path (dark blue path) within the fuzzy range using CTEA path (blue dotted path).

Following with identifying the fuzzy path based on the defined parameters including the distribution, spread factor and gateway election factor, based on the definition of the smooth path in Sub-section 4.4.4, a circular arcs/linear path called crisp path is structured aligned with the proposed fuzzy path with uniform circular arcs of $R = 300\ m$ (See Figure 5-3). The proposed circular arcs/linear relaxed path is divided into multiple stages as already shown in Figure 4-7. Throughout running of the path, the UAV has various traveling velocity and acceleration vectors and therefor different energy usage. As measured in Sub-section 4.4.4, the optimal speeds which minimize the UAV energy usage are calculated in arc and linear paths as $v^*_{curve} = 28\ m/s$, $v^*_{straight} = 30\ m/s$, respectively.

The simulation outcomes targeting enhance travelling flight speed, acceleration and power and energy spent by the UAV electrical system for the entire travel are represented and compared with BTSP path [39] in Figure 5-4 and Figure 5-5.

Based on the proposed relaxed route, initially once the UAV moves from starting point within its predefined path, its velocity raises from 0 to the given velocity for the curves as $28\ m/s$. Due to the variations of velocity throughout the linear route, UAV has the tangential acceleration-enabled movement equals to $a = 1.4\ m/s^2$. This figure clearly

shows that increasing the UAV speed to the given one for curves results in an increase in UAV energy consumption and accumulated energy usage.

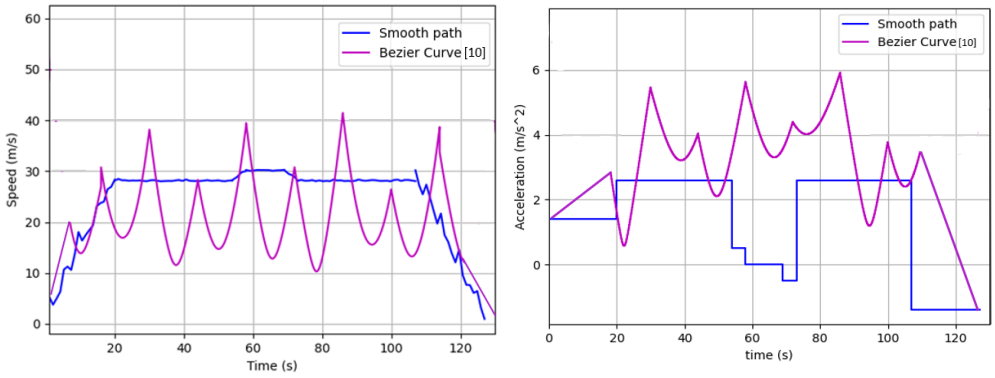


Figure 5-4 Comparison of the proposed relaxed path design with BTSP based on instantaneous speed and acceleration.

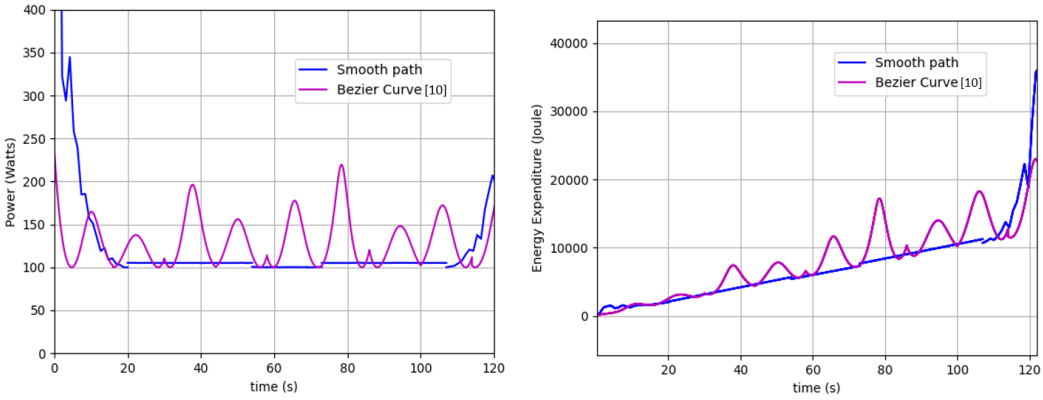


Figure 5-5 Comparison of the proposed relaxed path planning with BTSP based on instantaneous power and energy usage.

In the next phase, once the UAV traverses within the curve, the speed value is fixed on the optimal volume of $28\,m/s$. The centrifugal normal-to-the-path acceleration $a = 2.5\,m/s^2$ is responsible for UAV direction changing (shown in Figure 4-6). Once the UAV changes its direction from linear path to the curve one, the UAV power consumption drops and stabilizes around a constant value of $105\,watts$ and the UAV energy usage increases to the value of $5000\,joules$ with almost the same slope as last part.

Next, once the UAV returns to the linear path again, the power decreases to the minimal value of $100\,watts$ which justifies benefiting from movement without centrifugal acceleration and the sole tangential acceleration fluctuates over $a = \pm 0.5\,m/s^2$ due to UAV velocity variations between 28 to $30\,m/s$. The energy consumption of the UAV has also increased to 8000 joules.

In the following segment, the UAV represents the same behaviour as the first curve in terms of instantaneous velocity, acceleration, real-time power, and consumed energy once it is within the second curve.

Finally, once the UAV returns to the starting point, it brakes in order to land on the base station. Herein, the value of tangential acceleration due to decelerating is calculated as $a = -1.4\, m/s^2$. As shown in Figure 5-5, the portion of energy expenditure at this part, has dramatically raised to more than $30000\, joules$ due to the amount of instantaneous power that the UAV consumes to decelerate for landing. Hence, in the final phase, since the speed value is dropped to brake the UAV for landing, the instant energy usage increased exponentially.

For comparison between the proposed circular arcs/linear crisp path and the benchmark BTSP design, the BTSP path [39], as discussed in Sub-section 4.4.4, is designed within the fuzzy range. As shown in Figure 5-6, the BTSP is demonstrated for distribution of gateway-capable nodes equals with $D_{gc} = 20\%$ which is less than the threshold of gateway-capable nodes distribution $D_{gc}(th)$ (as discussed in Algorithm 1 in Sub-section 4.4.2). Hence, the Bezier curve is utilized to improve UAV energy efficiency on the entire travel by relaxing the sharp turning points of TSP route. Herein, the proposed UAV relaxed path is compared with the BTSP as benchmark in terms of energy efficiency.

It should be noted that the Bezier curves fitting on the edges are highly dependent on the neighbourhood of vising points [39], in which with increasing the radius of neighbourhood allowing the UAV to pass through the curves in a more relaxed manner. In this case, the radius of the visiting point neighbourhood is set to 100 metres for all visiting points, and the Bezier control points on each curve are structured on the edge of the neighbourhood circles based on the angle of turning points.

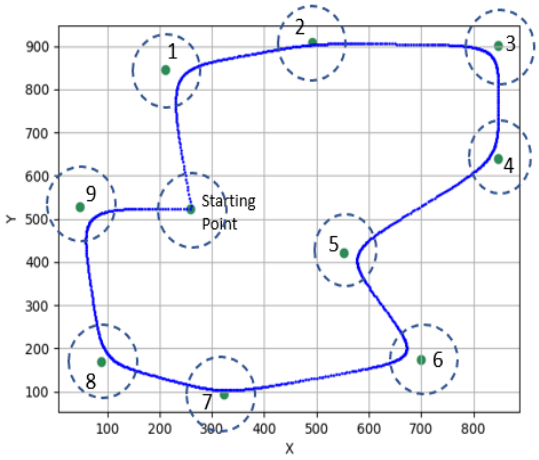


Figure 5-6 BTSP path designing considering multiple visit points.

As shown in Figure 5-6, first TSP is highlighted within the fuzzy region taking multiple gateways visiting points into account. Then the Bezier curves are aligned on the sharp turning points within the route accounting for four controlling points for each edge. The instantaneous speed, acceleration, power, and energy consumption of UAV within the entire route are obtained based on the normalized first and second derivatives of Bezier path vector (4-6) and the energy consumption of fixed-wing UAV (4-3), respectively.

According to the BTSP route design, when the UAV initiates its journey from starting point during the predefined linear route, its velocity raises to a maximal velocity. Next, once the UAV reaches at the curves relevant to visiting points, the instantaneous speed decreases into the range of 12-18 m/s depending on the roughness of the turning points. Hence, the UAV's velocity fluctuates around 10 to 40 m/s multiple times during the entire trip (Considering the outcomes demonstrated in Figure 5-4 pink graphs). The number of local minimal points in the real time speed plot reflects the number of curves that UAV brakes once it approaches them. The value of instantaneous acceleration due to UAV speed changes and turning points is expressed within Figure 5-4 (right side) and finally the value of instantaneous power and energy usage are shown within Figure 5-5. Since there are numerous curves in the route, UAV instantaneous power fluctuates around 100 to 220 $watts$ repeatedly. It is clear from Figure 5-5 power plot that once the UAV is within the curves, it spends more energy to prevent overshooting. It should be noted that the average value of power consumption for BTSP is around 140 $watts$ which compared to the proposed circular arcs/linear relaxed path (100-105 $watts$) has a worse performance. Figure 5-5 also shows the value of instantaneous UAV energy consumption during the entire travel. It is evident that the value of instantaneous energy consumption

is greater than $10000\ joules$ most of the times which is a meaningful value of energy consumption when compared to the circular arcs/linear path. This extra energy consumption is due to the energy needed for direction changes in BTSP versus relaxed path, which presents fewer direction changes within the fuzzy range and thus justifies an energy efficient route planning method compared to BTSP.

5.4.2 UAV Path Design Model Validation

For the simulation model validation, SITL Mission Planner is utilised to identify the proposed Circular arcs/linear and BTSP flight paths and analyse the real time behaviour of the UAV performance including instantaneous power and speed variants. As discussed in Chapter 3, there is a simulation embedded in Ardupilot Mission Planner named software-in-the-loop (SITL) that provides simulation prior to arming the vehicle and executing the mission in real time [102]. Numerous path designs can be generated in SITL Mission Planner to visualise the real-world behaviour of the UAV. Two examples of the proposed crisp and Bezier curve paths are validated here to evaluate the value of UAV energy consumption per mission. To begin with, the proposed model is tested in the real-time simulator using the VTOL quadplane category. The value of battery current consumed by the brushless motors is assumed as a criterion for analysing the instantaneous power and therefore energy consumption. The UAV voltage is always maintained fixed.

Firstly, to specify the simulation outlines on SITL, numerous waypoints are needed to be conducted on the UAV mission plan. Next, the circular arcs/linear relaxed path with similar length and curves as simulation is generated in SITL (See Figure 5-7). The UAV velocity is defined based on the circular arcs/linear relaxed path simulation parameters. Once the UAV arrives at the curves, it maintains a velocity of $28\ m/s$ and reaches to the linear route at a speed of $30\ m/s$. As shown in Figure 5-8, the values of instant consumed current (upper plot) and speed (lower plot) for the whole journey are logged in SITL. The orange circles of the upper figure show the current variable during VTOL-take-off and VTOL-landing of the plane. The UAV spends a significant amount of energy corresponding to 50 and $40\ amps$ for take-off and landing, respectively. The UAV traveling time, excluding UAV take-off and landing parts is approximately 120 seconds, as calculated by simulation (the red circle within Figure 5-8). The UAV spends around 12 to $17\ amps$ current moving in between visits points. Assuming the amount of UAV battery voltage is fixed on $12.9\ v$, the real time power is obtained around 150 to

180 $watts$. There is a slight difference between the amount of simulated power from analysis (100 to 110 $watts$) and real-world one here which could be associated with the amount of extra power spent by transmitting the telemetry messages. Additionally, Figure 5-8 represents the velocity variant throughout the entire mission (lower plot). The value of velocity fluctuates around 25 to 30 m/s until the UAV arrives to the ending point.

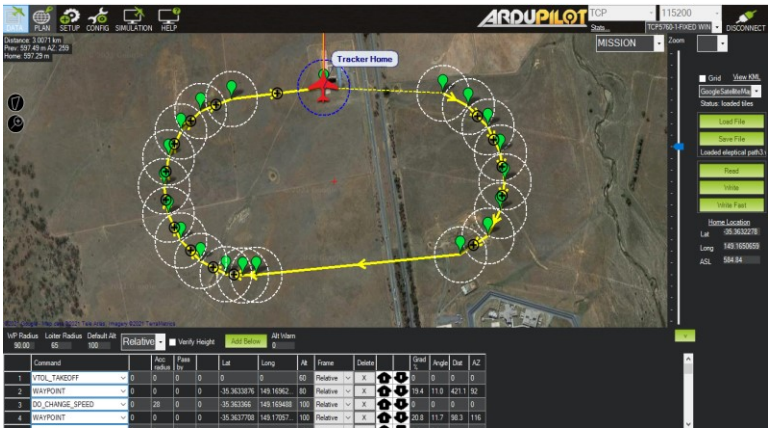
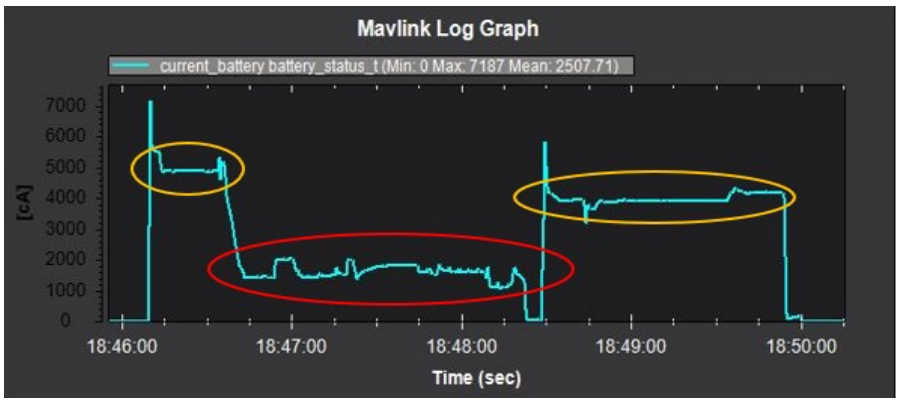
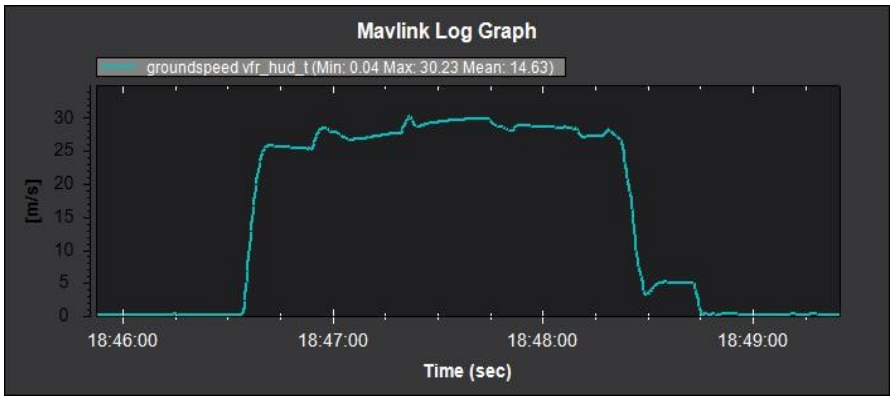


Figure 5-7 Waypoint planning for the circular arcs/linear relaxed path based on SITL.

It is also evident from Figure 5-8 that once the UAV raises its velocity instantly following take-off and decreases that prior to landing, the amplitude of current drops dramatically following take-off and surges before landing. This is reflected in analytical model development as shown in the blue plots of instantaneous simulated speed and power diagrams in Figures 5-4 and 5-5.



a).



b).

Figure 5-8 The variation of instant current usage (a) and speed (b) for the circular arcs/linear path.

As represented in Figure 5-9, to validate the BTSP flight route, it is also generated by SITL Mission planner with the same shape and length as the route already designed in Figure 5-6. The conducted BTSP is highly dependent on the neighbourhood radius (the radius which is defined in SITL Mission Planner indicating that once the UAV traverses over that circle, this visiting point is regarded the successfully passed point). By fixing the neighbourhood radius on $100\,meters$ in SITL Mission Planner (same as the neighbourhood radius in the simulation), the UAV route is generated by smooth turning points as shown in Figure 5-10. Hence, this path is highlighted as the same path obtained from the analytical simulation in the previous section.

Figure 5-9 Waypoint planning for the BTSP flight plan.

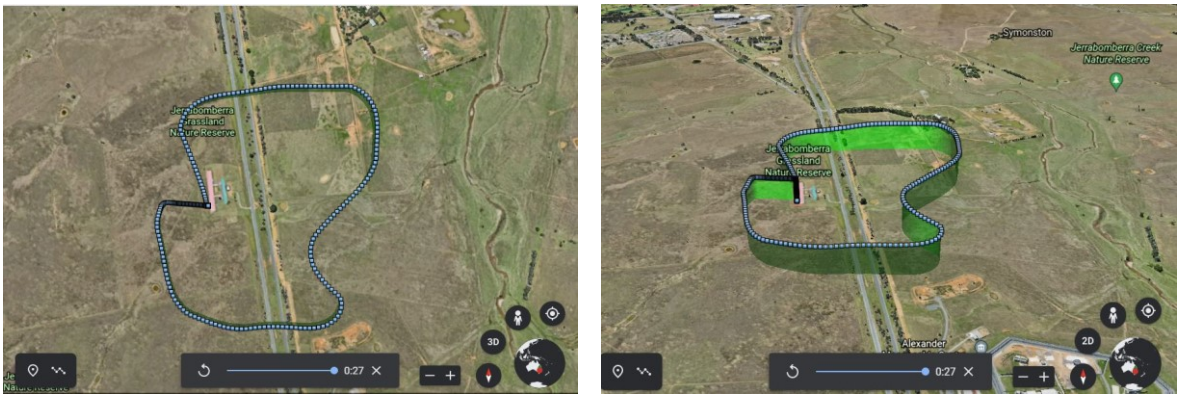


Figure 5-10 The logfile obtained from the BTSP flight plan testing.

Figure 5-11 depicts the value of energy consumption based on the UAV current expenditure once the UAV traverses on the BTSP route. It is evident from the figure that the value of the current consumption is higher with incremental fluctuations compared with the proposed circular arcs/linear path (Figure 5-8 upper plot) due to the higher number of UAV direction changings within the BTSP flying path. Moreover, when comparing the analytical simulation of the BTSP in the previous section with the BTSP actual validation design in SITL Mission Planner, it is clear that the value of instantaneous power fluctuates over 140 $watts$ (analytical simulation design) vs. 230 $watts$ (actual validation design), which justifies the impact of other factors on the energy usage of actual simulation, such as transmitting telemetry messages to the UAV which are already assumed within SITL Mission Planner.

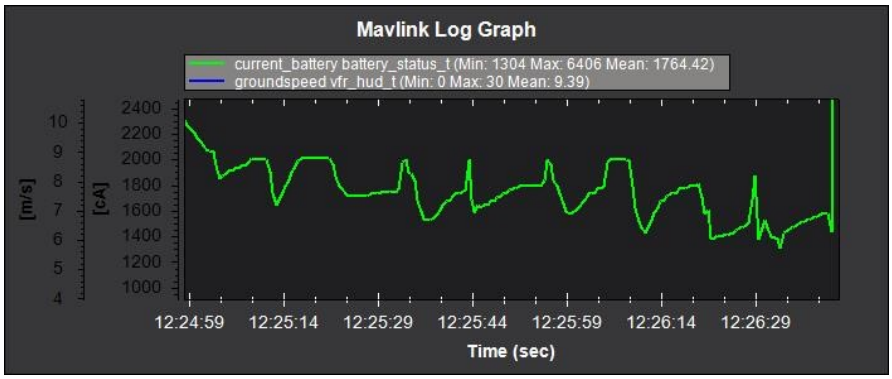


Figure 5-11 The instantaneous current usage based on BTSP flight path testing.

5.5 Development of Network Communication Generic Simulation Model

5.5.1 Simulation Model for Communication Window of Time

In this Section, the simulation works conducted on MATLAB and CupCarbon to support UAV-Ground connectivity are presented. Following the deployment of $N = 100$ SNs through the non-uniform random distribution in MATLAB, $K - means$ clustering approach is utilised to group the SNs based on Euclidean distances. An initial path is mapped using the CTEA method (as discussed in Sub-section 4.6.1). Then, a fuzzy region for UAV flying route considering the CTEA initial path and the communication range of gateway-capable nodes is defined and an appropriate UAV relaxed path, in this case BTSP, traveling the visit points accounting for minimal UAV energy usage within fuzzy range is aligned based on Algorithm.1. Finally, a proper UAV-Ground scheduling following the proposed communication window is aligned on the UAV relaxed flight path. The altitude of the UAV is assumed fixed as $H = 100m$ during the entire UAV journey with the UAV path aligned as a new dimension placement on the graph. The simulation parameters for this experiment are shown in Table 5-1.

Table 5-1 Model simulation parameters.

Simulation Parameters	Value
bit rate (bit/sec)	320
Packet size (bytes)	512
transmission range (m)	100
UAV velocity (m/s)	(5-40)
Number of nodes	100
Time step size δ	0.5
Altitude of the UAV (m)	100
The percentage of Gateway-capable nodes distribution D	20%
The minimal spread factor of gateway-capable nodes	250
The maximal spread factor of gateway-capable nodes	750
C1 Constant [42]	$9.26 \times 10{-}4$
C2 Constant [42]	2250

This section provides numerical results for evaluating the communication performance of the proposed models in Section 4.6. To this end, as the proposed window of connectivity is expressed on (4-16), the connectivity window of time is plotted over the UAV relaxed path within the fuzzy route depicted in Figure 5-12 (pink circle). The fuzzy route is identified (the hatched region in Figure 5-12) accounting for the average percentage of Gateway-capable nodes distribution and spread factor equal to 20% and between (700-850), respectively. BTSP is also used to relax the UAV route in the sharp edges (the pink UAV flying route within the fuzzy hatched range). The connectivity window is identified as the time once the UAV is within the entry and departed points calculated based on a range of UAV's speeds (see (4-17)). Herein, with raising the average UAV's speed, the average connectivity time is dropped which results in network performance degradation (see Figure 5-13).

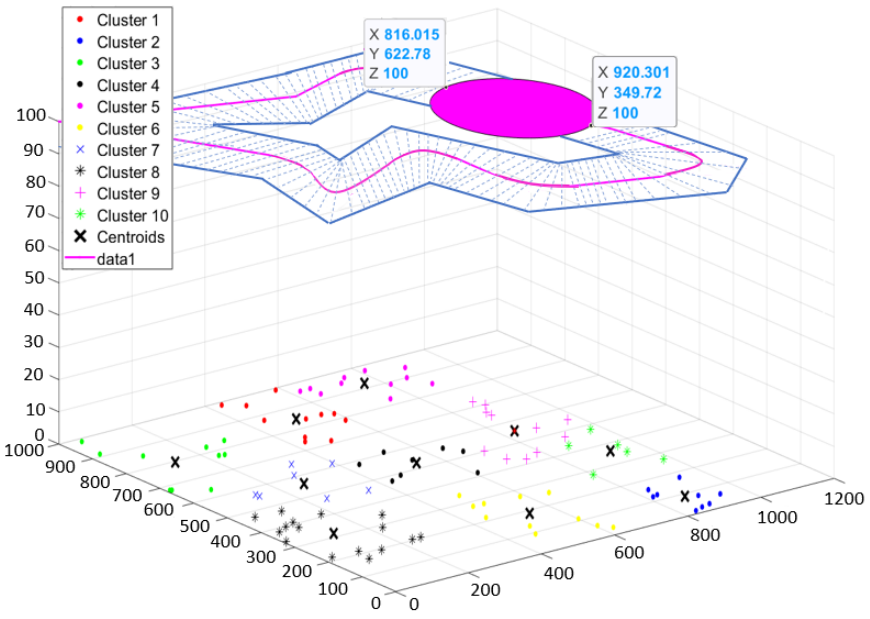


Figure 5-12 The proposed communication window of connectivity for one group of SNs.

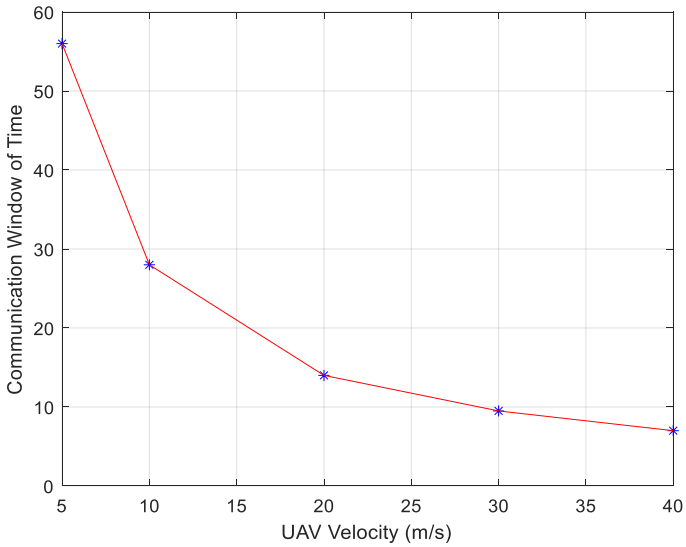


Figure 5-13 Effect of the UAV's speed on the average communication window of connectivity.

5.5.2 Simulation Model for UAV Communication with Individual SNs

The proposed priority-based generic communication scenario discussed in the section 4.6.2 to model the UAV interaction with individual SNs is initially simulated in Contiki-Cooja. The coordinates of the SNs are transferred from MATLAB to the Contiki-Cooja simulator point by point. A number of $N = 50$ SNs as well as one coordinator (UAV) are transferred from MATLAB to the virtualisation toolbox in Contiki-Cooja to simulate the UAV-Ground behaviour in a more detailed way. As shown in Figure 5-14, the trajectory of the coordinator (UAV) in Contiki-Cooja is exactly represented in accordance with the designed CTEA initial trajectory in MATLAB. The SNs could be divided into multiple group members belonging to multiple connectivity windows of time with $N = 50$ points. Hence, a Contiki-Cooja network scenario is designed to assess the communication performance in terms of packet delivery rate and average latency under different UAV velocities. It is presumed that the UAV begins its path outside of the communication window of time for each group, sequentially passes through each step of the path and finally departs from the associated window of connectivity. While the UAV is within each communication window of time, it communicates with individual SNs from the same group based on the allocation of distinct time slots to each SN to communicate with the UAV while the UAV is within the closest distance from the affected SN (as discussed in Section 4.6.2). The simulation outcomes including the maximum packet delivery rate and minimal average latencies for various UAV speeds under the UAV transmission range of $175m$ has been given in Table II. This table also indicates the effect of various ranges of UAV velocities on the packet delivery starting from $V = 5\,m/s$ to $V = 40\,m/s$. The

number of dropped packets increases as the UAV speed increases, because the UAV has less time to communicate with the SNs, and some packets belonging to the SNs particularly on the edge of the UAV coverage area, are dropped.

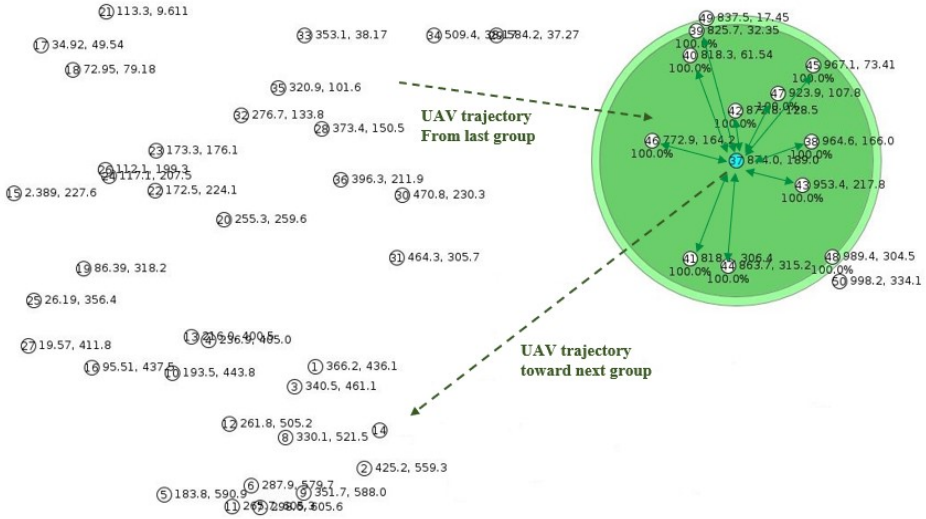


Figure 5-14 Data gathering of individual SNs designed in Contiki-Cooja.

Another performance measure evaluated here is the average latency once the UAV completes its travel on each connectivity window. As shown in Table 5-2, it is obvious that the UAV velocity of 5 m/s achieves better communication performance compared with other velocities with the packet delivery rate of 93 percent and latency of 90 ms.

Table 5-2 The simulation outcomes on PDR and average latency based on UAV velocity.

Number of SNs	50				
UAV velocity (m/s)	5	10	20	30	40
Packet Delivery Ratio	93%	90%	85%	74%	63%
Average Time Latency (ms)	90	48.87	63.25	87.5	80.81

5.5.3 Simulation Model for UAV Communication with Group Representatives

In this Section, the proposed generic communication network between the ground sensor nodes, group representatives and the UAV considering UAV CTEA initial path is simulated in CupCarbon network simulator. To this end, as discussed in Sub-section 4.6.3, the coordinates of the ground nodes as well as their assigned gateways are passed point by point from MATLAB to the CupCarbon. The UAV path in CupCarbon is exactly sketched based on the CTEA initial path. The UAV is expected to begin its trajectory outside of any gateway's communication window, travel sequentially through each gateway's connectivity window, and eventually depart from the associated connectivity window of the last gateway. While the spread factor is assumed to set to two various values of $\sigma_{gc} = 250$ for the dense network and $\sigma_{gc} = 750$ for the sparse one, the percentage of the gateway-capable nodes, D, is scaled up from 5% to 30% of the entire SNs to analyse the network communication performance for each network density. The network structure testbed for the proposed dense and sparse network is depicted in Figure 5-15. A detailed coding of transmitter (SN), router (gateway) and receiver (UAV) within CupCarbon is presented in Appendix A.6. Moreover, the effect of UAV speed on the network efficiency in the case of UAV communication with group representatives has been assessed. From the network simulation testing, it is evident that when the average UAV speed increases, the communication performance of packet delivery rate decreases within a given UAV communication range. Hence, the packet delivery rate based on different UAV speeds and percentages of the gateway-capable node distribution for two scenarios of the sparse and dense networks are shown in Figure 5-16. As represented in these figures, the best performance takes place when the percentage of the gateway-capable nodes equals $D = 30\%$, whereas by limiting the number of gateway-capable nodes to $D = 15\%$ or $D = 5\%$ of the entire network, the number of lost packets increases with the worse performance taking place when $D = 5\%$ gateway-capable nodes of distribution. The reason for this is that as the population of gateway-capable nodes grows under the fixed number of sensor node and density, the number of SNs connected to each gateway decreases, resulting in an increase in the number of received packets on each gateway’s buffer from connected SNs and following that increasing the number of uploaded packets on the UAV.

Figure 5-15 Data gathering of grouping sensor nodes designed in CubCarbon (upper design belongs to the sparse network with σ_{gc}= 750, while the lower one belongs to dense network with σ_{gc}= 250).

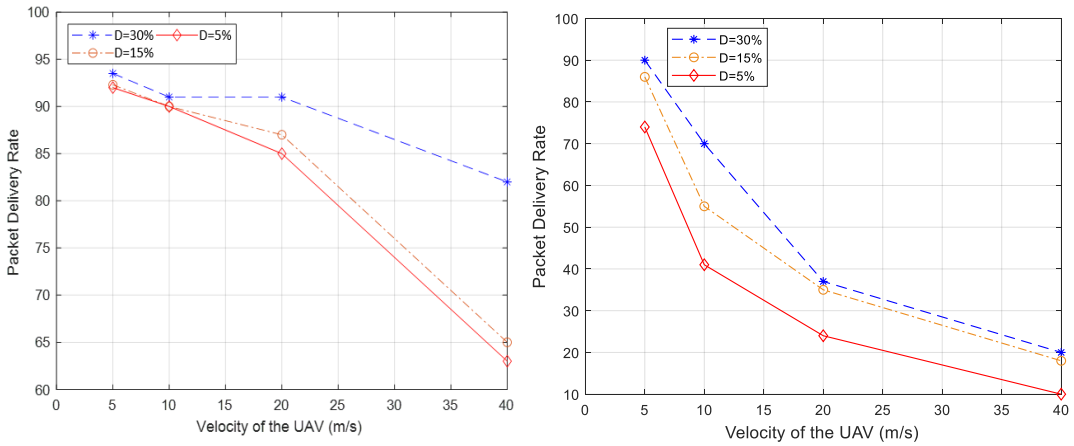


Figure 5-16 The impact of raising the velocity of the UAV on PDR for sparse & dense networks (the left figure shows the sparse network PDR with σ_{gc}=750, while the right one represents the dense network PDR with σ_{gc}=250).

These figures also reflect the effect of various ranges of UAV speed ranging from $V = 5\,m/s$ to $V = 40\,m/s$ on the packets received on the UAV receiver. With raising the UAV velocity, the number of dropped packets increased since the UAV has less time to communicate with the gateways and hence some transactions of gateways are dropped. By comparing the simulation results for sparse (left plot) and dense networks (right plot) in Figure 5-16, it is evident that the number of packets transmitted to the UAV in a dense network is lower than a sparse one due to the interference spots that UAV deals with while collecting the data from multiple gateways at the same time.

5.6 Network Communication based on SDWSN Strategy

As discussed in Section 4.5, the network communication model in SDWSN is divided into two main phases: topological scanning pre-orchestration phase via control information messages and data gathering post-orchestration phase via sensing data information messages. To begin with, a test is performed for a network with multiple functionalities (Leaf, router, and gateway) in which the topological structure mode can be dynamically configured based on star, single hop, multi-hop network structures. The ground network is represented in Contiki-Cooja as shown in Figure 5-17 with all gateway-capable nodes are involved in data gathering model (it can be represented as the time preceding the network topological initial orchestration). In this network structure, the ground network consists of tree network arrangements prior to orchestration phase.

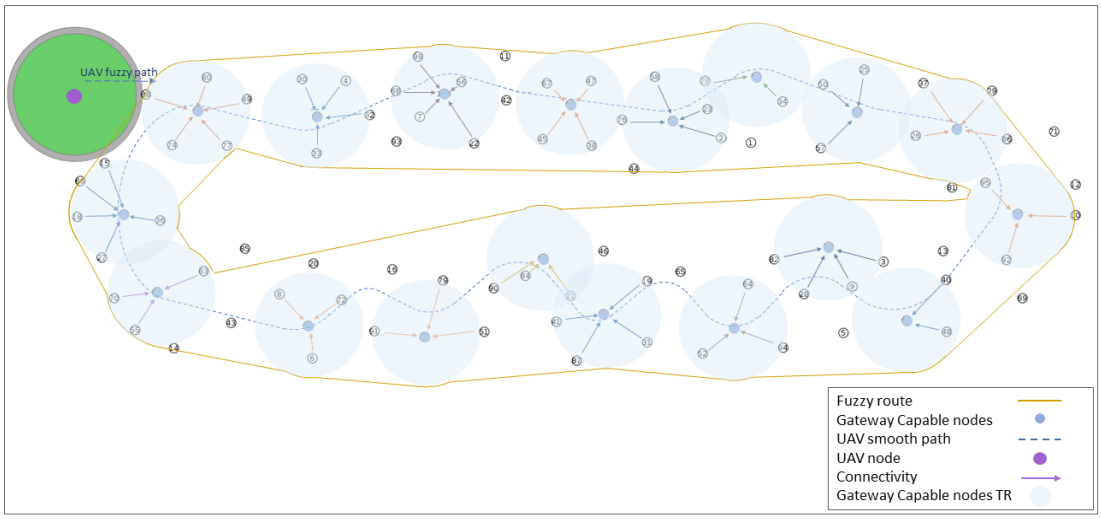


Figure 5-17 The distribution of network components within the UAV initial scanning pre-orchestration phase.

Following the orchestration phase and passing the control data to the drone, the most appropriate gateways for each group are elected based on fitness model computation and

various diverse network architectures such as tree, one/multi-hop data transmission groups are formed (See Figure 5-18). This phase can be represented as the phase after orchestration. The sensing data upstream flow takes place during this phase and the UAV passes over each elected gateway to gather the accumulated data. Herein, the UAV paths in both phases are considered as the proposed smooth path, as discussed in Section 4.4. The simulation goal is to evaluate the residual energy vs. packet delivery rate performances either during scanning pre-orchestration phase or data gathering post-orchestration phase. For energy evaluation per node, we have used the energytracker built-in tool in Contiki-Cooja, as discussed in Chapter 3. A testbed is designed for ground network for various density spread factors σ_{gc} with expanding the transmission message rate from 1 message per seconds to 100 messages per seconds. The ground network transmission ranges for all ground network components are set to 50 meters. The simulation time is set to 60 s for data transferring and the designed packets frames for the scanning and data gathering phases are presumed based on the packet frame designs suggested in Sub-section 4.5.4. The simulation parameters are shown in Table 5-3.

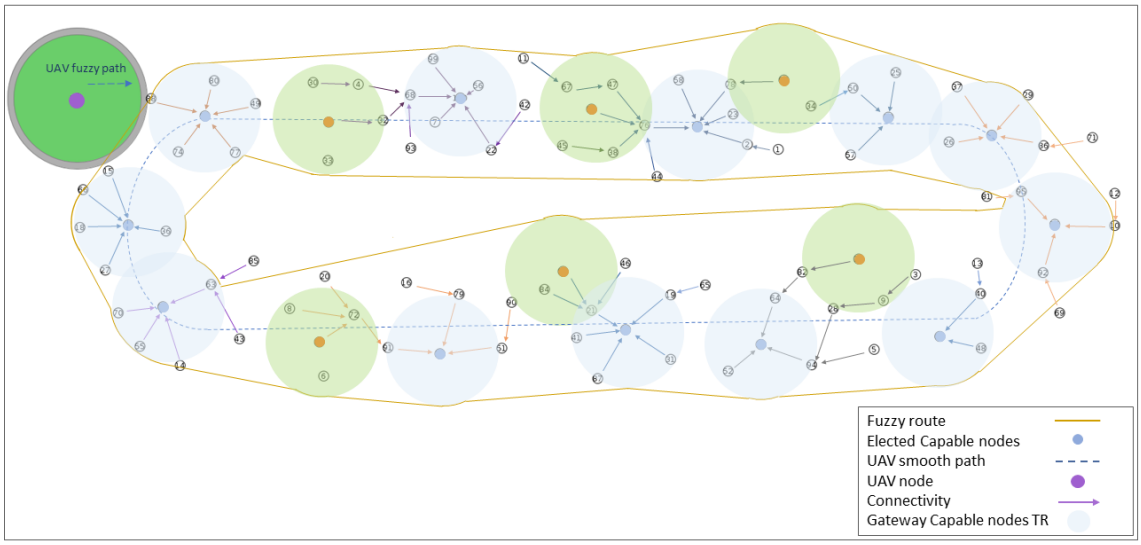


Figure 5-18 The distribution of network components in Contiki-Cooja network simulator for post-orchestration data gathering phase.

Table 5-3 Model simulation parameters.

Transmitting circuit loss or E_{elec} [106]	0.05 μJ
Free space power amplification ε_{fs} [106]	0.002 J
Air-to-Ground transmission range threshold (d_0)	550 m
Number of Ground Network entities	100
Distribution of Gateway-Capable nodes D	20%
gateway election factor ξ for initial scanning phase	1
gateway election factor ξ for data gathering phase	2/3
Ground Network Transmission Range	50 m
Average UAV Speed for both phases	20 m/s
Mission Completion time	1580 seconds
Transmitted Message rates	1-100 msg/sec

The ground network is structured in Contiki-Cooja considering various network formation scenarios depending on the election of gateway nodes out of the predefined gateway-capable nodes process. Herein, as shown in Figure 5-17 and 5-18, two various network formations for the same deployment of sensor nodes and gateway-capable nodes have been assumed to observe their impacts on the shape of the UAV path, ground SNs' energy consumption and the percentage of served SNs on the ground. Both architecture designs have two communication transaction message phases within them, which are the data flow notification phase and the scanning/data gathering phase. During the data flow notification phase, as discussed in Section 4.5.4, the updated functionality of each node emerging from fitness model decisions are returned to the elected routers and gateways as well as their specified leaf-nodes via the notification messages to orchestrate/re-orchestrate the ground network configuration. In the scanning phase data flow, the packet frames are relayed to the potential gateways and then uploaded to the UAV to provide inputs for

fitness model in order to redefine the functionality of the potential nodes, whereas, in the data gathering phase, the sensing packet frames are uploaded to the UAV.

In the initial scanning pre-orchestration process, as shown in Figure 5-17, all gateway-capable nodes are involved in data gathering model with the gateway election factor of ξ =1 which creates the star network structure. Figure 5-19 left depicts the data flow among the ground network components for a sample network group within this architecture. While based on Figure 5-18, only a percentage of gateway-capable nodes are elected as gateways and involved in data gathering model with $\xi = 2/3$; hence, the multi-hop tree-based communication is generated to transfer the data from leaf nodes to the elected gateways. The data flow amongst the ground network components for a sample network group in this network architecture is represented in Figure 5-19 right. A detailed coding of leaf, router, gateway nodes within Contiki-Cooja network simulator is presented in Appendix A.2 to A.5. The UAV path design for the scanning phase is drawn as Bezier curve path enabling traveling over many gateways while for the second network structure, the UAV crisp circular arcs/linear path has been shaped for the UAV to pass over a lower number of elected gateways to gather the data. As simulation outcomes, the values of received packets per gateway and ground energy consumption are recorded in Contiki-Cooja for each scenario.

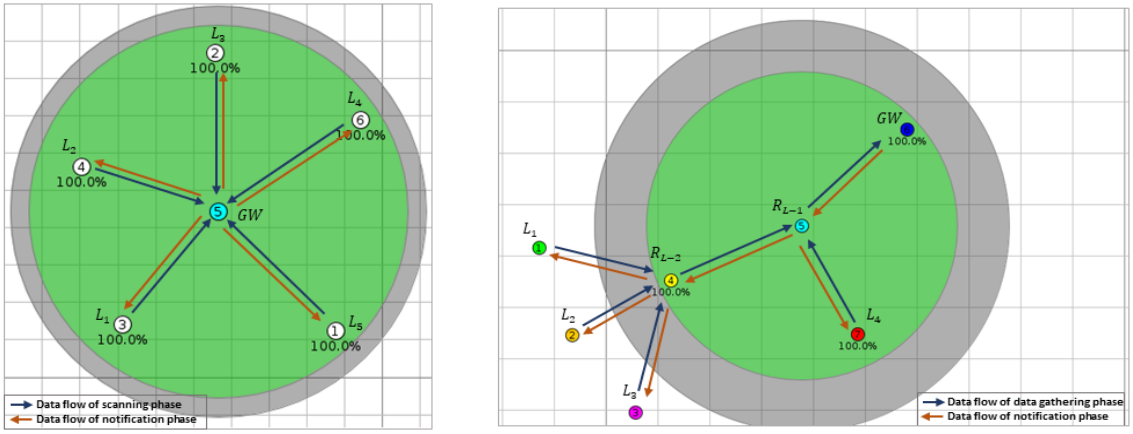


Figure 5-19 Data flow amongst the ground network components for initial scanning pre-orchestration phase (left plot) and subsequent post-orchestration data gathering phases (right plot).

5.6.1 Ground Network Performance

The ground network energy consumption model, as discussed in Section 4.5.3, is based on the definition of each SN's cycling time for each state, including TX, RX, Idle, and

Low Power. In an attempt to study the current consumption of the Contiki-Cooja motes in a distributed ground network, the current profiles of two hops communication among the leaf nodes, routers and gateways are represented in Figure 5-20 and 5-21. The time of active TX, RX, Idle and Low Power states are obtained per cycling time based on the energytracker tool in Contiki-Cooja network simulator. Also, the values of current consumption per each idle, active, and low power modes are explored from the datasheet of CC2538 ti and presumed based on the Table.1 in Chapter 4. The amount of voltage is considered as $V = 3\ v$ for the entire experiment.

Figure 5-20 depicts current profiles for the leaf and router-L1 nodes obtained from the Contiki-Cooja energytracker tool for a two-hop ground network communication structure. The plot clearly shows that the packet interval is $1\ s$ and that the device enters Power-mode in between packets. The leaf node first operates in low power mode during the specified time cycle, then switches to idle and transmission mode to initiate transmission (see Figure 5-20 left). The module then returns to low power mode to save power once the entire packet is disseminated during the dissemination time. It should be noted that the duration of the leaf node's dissemination period is highly dependent on the size of the designed packets, which equals $830\ s$ for the array of 8 packets frame. Figure 5-20 (right) depicts the current profile value for the router-L1 after allocating $16320\ s$ of its cycling time to receiving and $4530\ s$ to dissemination active modes, respectively. There is a meaningful time gap between receiving and transmitting modes to allow the transceiver to switch from receiving to transmitting mode. Due to the responsibility of receiving and relaying data from several leaf nodes to the router-L1, the receiving time for router L-1 is longer than the transmission time based on the Contiki-Cooja simulator. Figure 5-21 (left) depicts the current profile for the router-L2 after allocating $5360\ s$ of cycling time for receiving and 13840 s for dissemination active modes. The dissemination state time in router-L2 is longer than in router-L1 due to additional leaf node connections as well as other data relayed to router-L2 to be transferred to the coordinator. Finally, Figure 5-21 (right) depicts the current profile for the gateway node, in which the receiver mode is active for $13840\ s$ during each cycling time.

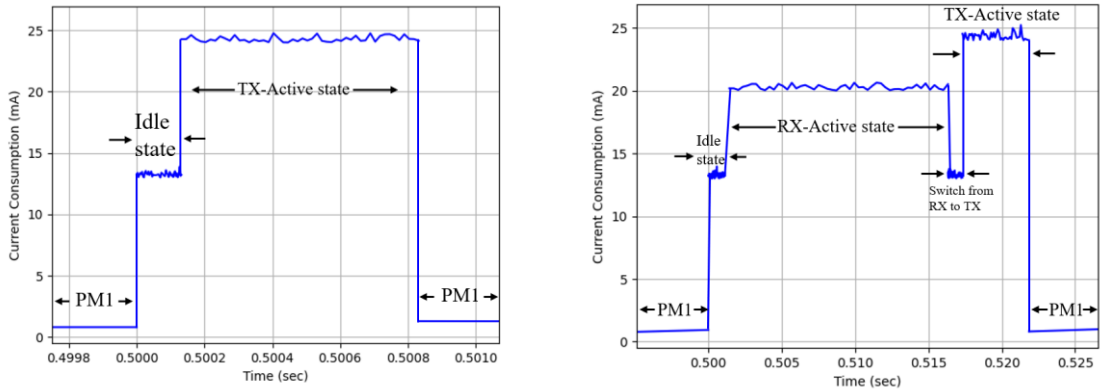


Figure 5-20 Current Consumption for packet dissemination of each Leaf Node (left figure) and packet dissemination and receiving of each Router-L1 Node (right figure) per time cycle.

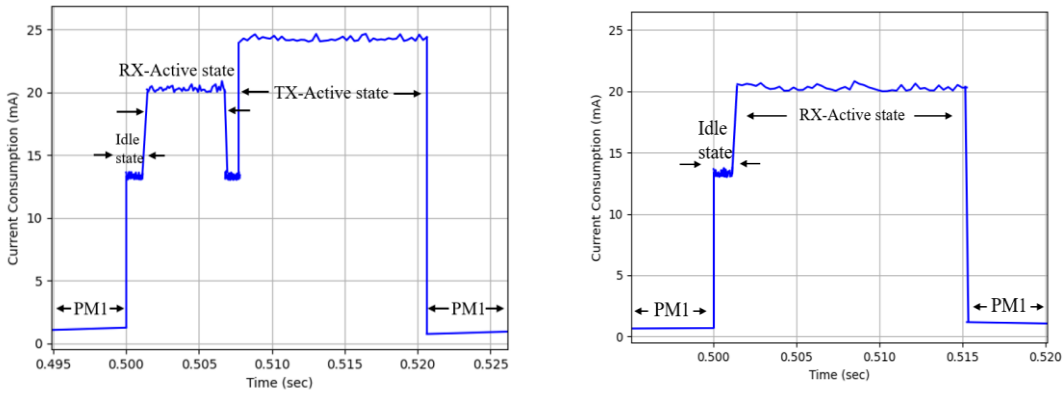


Figure 5-21 Current Consumption for Packet transmission and receiving of router-L2 node (left figure) and packet receiving of gateway (right figure) per time cycle.

Note that the aforementioned outcomes are part of an experiment to measure current profile for a specified two-hop network using the Contiki-Cooja simulator and CC2538 ti modules. The same experiment is carried out to examine the current consumption for alternative network structures such as star, 1-hop network etc. The aggregated power usage is measured for a network of multiple diverse structures based on various distribution of network components such as those in Figure 5-18. The simulation model is defined and analysed based on network architecture in Figure 5-18 following the updated network orchestration phase, and the simulation outcomes are provided in Figure 5-22 and 5-23. The simulation time is set to 60 s in Contiki-Cooja, since the average window of connectivity among the UAV and gateways in the air-to-Ground communication is 60 s based on the defined velocity of UAV, and the UAV-Gateways window of communication is presumed to be 60 s for all gateways. Hence, the simulation time for each group is considered 60 s in Contiki-Cooja, which is equivalent to the same period once the UAV is within the communication window of time of a specified gateway.

Herein, the ground network communication performances such as accumulated received packets in gateways and the energy consumption of the entire network based on a range of various network sparsity σ_{gc} and communication message rates are calculated and shown in Figures 5-22 and 5-23.

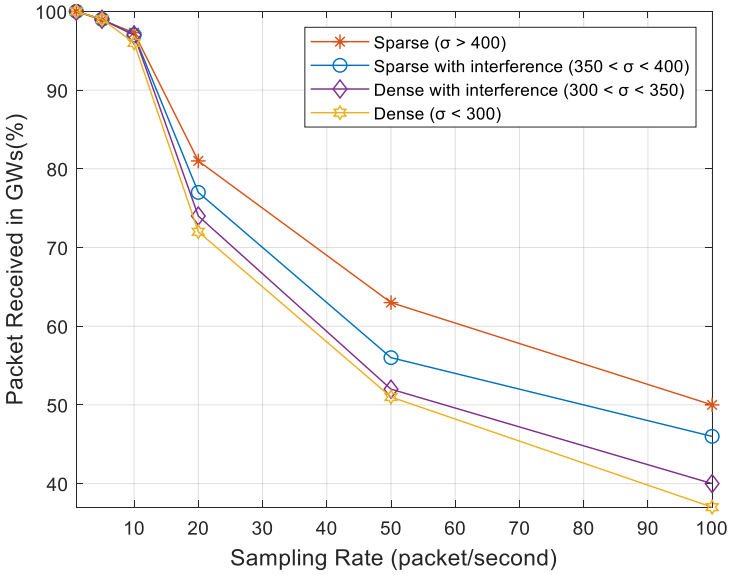


Figure 5-22 Packet received in Gateways from Leaf nodes in Ground network communication via Contiki-Cooja network simulator.

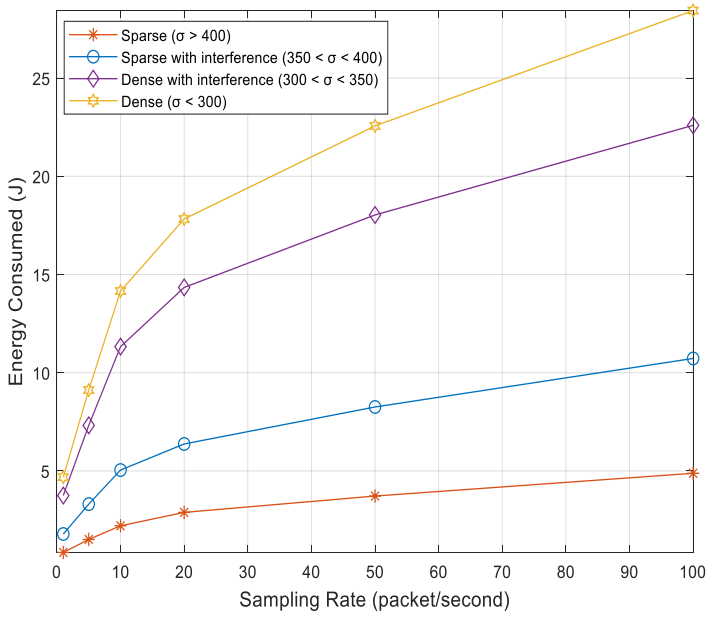


Figure 5-23 Energy consumption of ground network in terms of message rate and gateway nodes' spread factor via Contiki-Cooja network simulator.

As illustrated in these two figures, whenever network density increases, the ground network packet delivery and energy cost for message transmission from leaf to gateway

are degraded due to effect of interference from other group members. Furthermore, it is obvious from these figures that as the message rate increases, the number of received packets in the gateways' buffers decreases and the energy consumption of ground network increases. The reason for this is that as the message rate increases, so does the cycling time for the ground network components, causing the ground network components to consume more energy while delivering fewer packets to the gateways.

5.6.2 Air-to-Ground SDWSN Modelling Strategies and Performance

CupCarbon network simulator is used for the air-to-Ground communication between elected Gateways and the UAV for the proposed SDWSN communication model.

Numerous air-to-Ground data gathering scenarios can be developed in CupCarbon to assess the network performance of SDWSN communication model. The testbed design is based on defining multiple rounds for the UAV-Ground communication phases including initial scanning pre-orchestration or data gathering post-orchestration phases. Figures 5-24 and 5-25 depict these two simulation phases based on the distribution of selected gateways in CupCarbon.

In both scenarios, all parameters are assumed to be fixed. The location of gateway nodes in CupCarbon is assumed to be the same as the testbed in the Contiki-Cooja ground network model, facilitating air-to-Ground communication with the drone. Also, the size of data buffered in each elected gateway is determined by the computed communication overhead per gateway in Contiki-Cooja. UAV velocity, length and shape of the path are based on the proposed smooth path design within the fuzzy route, as discussed in Sub-ection 5.4.1. The air-to-Ground transmission range, as shown in Table II, is considered up to $550\ m$ in CupCarbon, once the average RSS is below $-80\ dBm$. The reason for this is that the communication rate for this transmission range is usually fair [107].

Figure 5-24 The air-to-Ground Communication among the elected gateways and UAV for initial scanning pre-orchestration phase in CupCarbon.

Figure 5-25 The air-to-Ground communication simulation for post-orchestration data gathering phase on CupCarbon.

As discussed in Sub-section 5.5.1, air-to-Ground connectivity is tested taking the movement of the UAV along the smooth path over the defined communication window of time for each elected gateway into account. Moreover, the average velocity of the UAV for this simulation is assumed $20\ m/s$ and the UAV has the responsibility to pick the accumulated data from each gateway while passing over the communication range of each gateway. The path length is equal to the length required by the UAV to pass through all gateways. The simulation model is defined and analysed based on network architecture in Figure 5-25 for post-orchestration data gathering phase with the same fixed network parameters such as the gateway election factor of $\xi = 2/3$ and the distribution of

gateway-capable nodes of $D = 20\%$ in order to observe the impact of varying the density spread factor parameter on the network performance. The simulation outcomes are provided in Figures 5-26, 5-27 and 5-28. The performances metrics including the percentage of overall packet delivery from leaf to the UAV, the energy consumption of UAV receiver, and the overall network cost for message transmission from leaf to the UAV are calculated in this section. Herein, the percentage of overall packets uploaded to the UAV buffer based on various communication message rates and network densities is represented in Figure 5-26. As shown in the figure, as the message rate increases from 1 to 5 $msg/sec,$ the packet delivery rate on the UAV buffer decreases slightly, whereas once the message rate reaches in between 5 and 10 msg/sec, the percentage of received packets on the UAV buffer stabilises around a fixed value for a specific network density. The reason for this is that at these message rates, the UAV is capable to gather the higher amount of data stored in the gateways' buffer during each gateway visit at its communication window of time, while with increasing the transferring message ratio from 20 msg/sec, the ground network performance degrades moderately resulting in a decrease in the overall packet delivery to the UAV. In other words, once the message rate is bounded to lower values, the ground network outperforms the air-to-Ground one in terms of the percentage of uploaded packets via gateways and the UAV, since the air-to-Ground communication has a lower packet delivery rate due to the mobility of the UAV. With increasing the message rate, the situation will be reverse, the percentage of forwarded packets from gateways to the UAV raises while the ground network performance gets degraded. Hence, one solution to improve the end-to-end packet delivery performance is to adjust the speed of the drone considering the message rate, while the reason why the packet delivery is not raising to the values greater than 62% in this case is highly dependent on this issue. For velocity choices slower than $20\ m/s$, the UAV is capable to receive the entire buffered data that disseminated by the gateways. Furthermore, as depicted in this figure, whenever network density increases, the ground network packet delivery for the end-to-end message transmission from leaf to the UAV degrades due to effect of interference from other gateways on the UAV receiver.

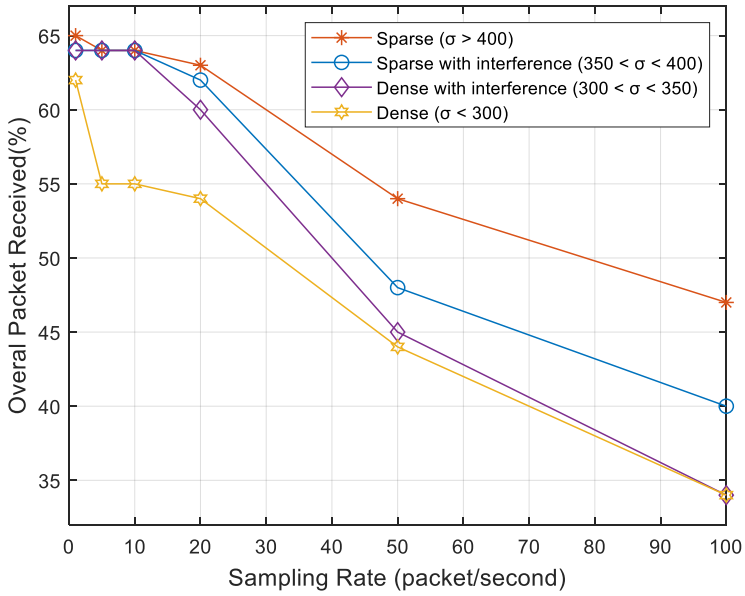


Figure 5-26 The overall packet delivery from leaf to the UAV based on message rate & gateway nodes' spread factor.

Other energy-dependent performances including the amount of energy consumed by the UAV receiver and the overall energy consumption for message transmission from the leaf to the UAV are depicted in Figures 5-27 and 5-28. As shown in these two figures, as network density and message rate increase, the UAV energy cost of receiving data packets from selected gateways and consequently the overall network cost of message delivery from leaf to drone increases.

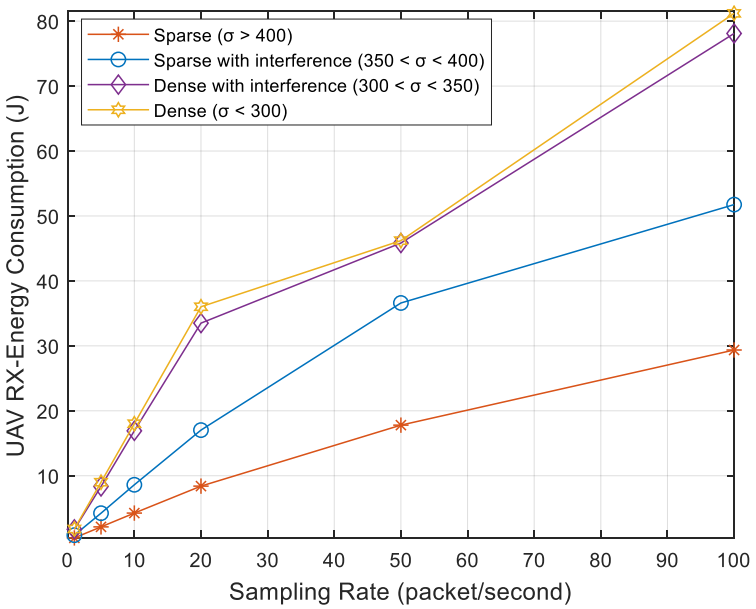


Figure 5-27 The energy consumption of UAV receiver in terms of message rate and gateway nodes' density based on CupCarbon network simulator.

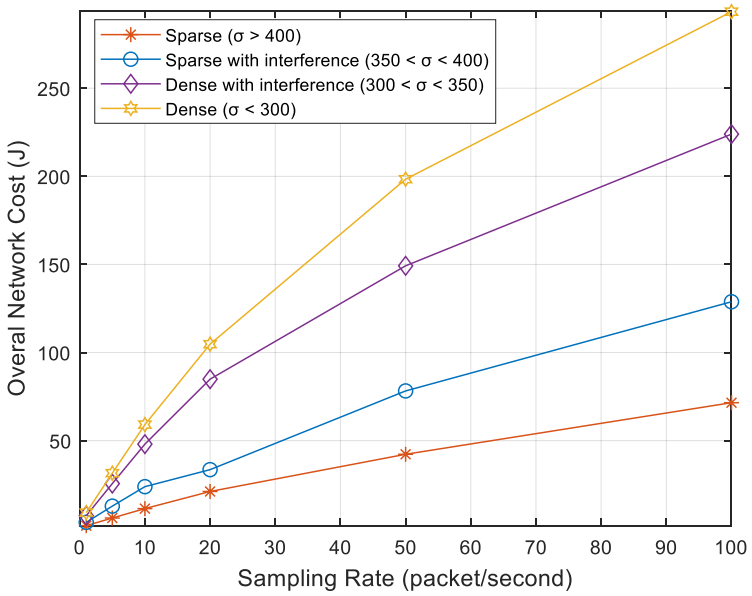


Figure 5-28 The overall network cost for message transmission from leaf to the UAV based on message rate & gateway nodes' density achieved from CupCarbon and Contiki-Cooja.

Also, the packet delivery for Ground network (from leaf nodes to gateways), the air-to-Ground network (from gateways to the UAV) and end-to-end network for two different phases (scanning pre-orchestration phase and data gathering post-orchestration phase) based on various network densities (from dense to sparse network) is represented in Figure 5-29. As shown in this figure, the packet delivery for initial scanning phase (either through ground, air-to-Ground and end-to-end communication) is higher than data gathering phase for various network densities, since the designed size of control packets, as discussed in Section 4.5.4, for configuring the recent roles of the network components is smaller than the data packets designed for the data gathering phase. Moreover, the gateway election factor is set to $\xi = 1$ for initial scanning phase implying that all gateway-capable nodes contribute to passing the control data to the UAV, whereas the gateway election factor of the data gathering phase is elected as $\xi = 2/3$, meaning that only a portion of gateway-capable nodes contribute to the data gathering process. As a result, packet delivery rate for initial scanning outperforms data gathering. Finally, the energy consumption for Ground network, the air-to-Ground network and end-to-end network for both the initial scanning phase prior to network orchestration and the data gathering phase following network orchestration based on various network densities are represented in Figure 5-30. It is obvious from this figure that data gathering phase consumes more energy than the initial scanning phase. Also, during both phases of network scanning and data gathering, the ground network consumes higher energy than

the air-to-Ground network communication. One reason for this is that the number of network entities involved in the communication process in the ground network is higher than in the air-to-Ground one. Finally, by looking into the Figures 5-29 and 5-30 at the same time, there is a meaningful energy gap between the overall energy consumption for data gathering and the network scanning phases, and the same gap is noticeable in overall packet delivery rate performance. This means that orchestration process through configuring the network roles is a more efficient way to improve the network functionality as this process costs less with lower energy consumption and higher packet delivery rates than the data gathering phase in total.

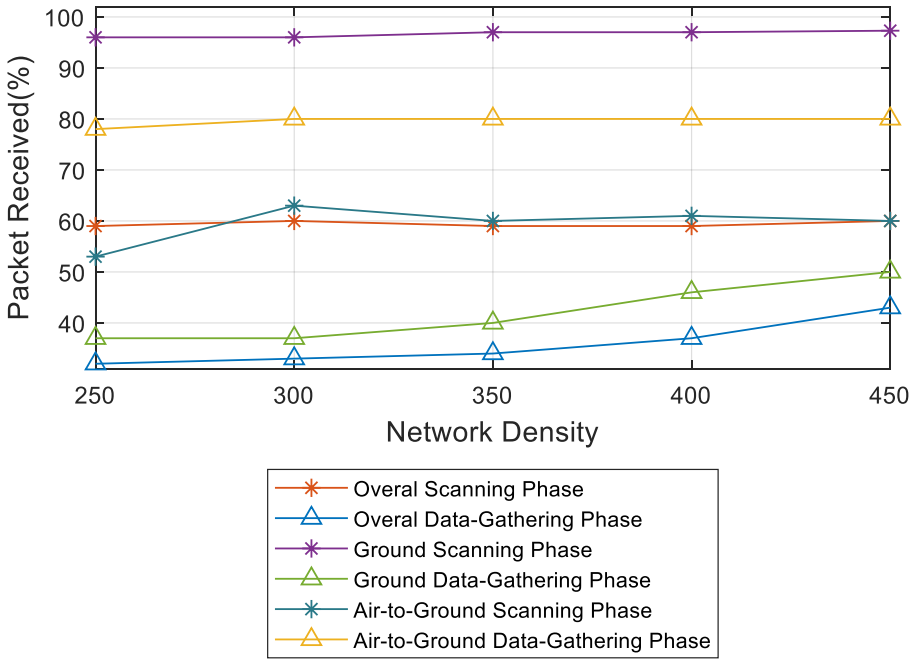


Figure 5-29 The packet delivery for Ground network (from leaf nodes to gateways), the air-to-Ground network (from gateways to the UAV) and end-to-end network for two different phases.

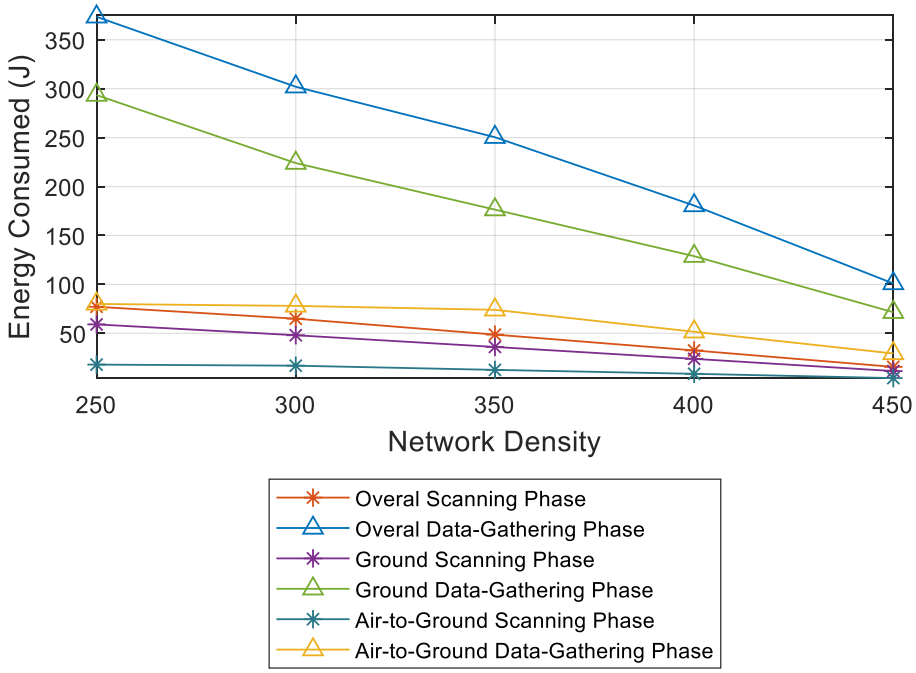


Figure 5-30 The energy consumption for Ground network (from leaf nodes to gateways), the air-to-Ground network (from gateways to the UAV) and end-to-end network for two different phases.

5.6.3 Comparison of SDWSN Modelling Strategies with Benchmark Model

To compare the suggested method with the method in [108], the proposed SDWSN network communication method shows better performance in terms of the rate of successful served sensor nodes following the configurability process. This comparison is based on various network scalability metric in which the proposed SDWSN network communication is compared to successive convex approximation (SCA) method in [108]. The network parameters are set to $D = 20\%$ for the distribution of gateway-capable nodes, $\xi = 2/3$ for the gateway election factor of the data gathering phase following the configurability process, and $\sigma_{gc} = 350$ for the density spread factor which is in between the dense and sparse networks. According to Figure 5-31, with increasing the number of deployed nodes on the ground, the number of served SNs in the proposed method gains a larger gap from SCA algorithm. Additionally, the complexity of the proposed communication network is exceedingly lower than the SCA algorithm [108] that employs complex non-convex/convex optimization algorithms to deal with the connection-based UAV path planning problems.

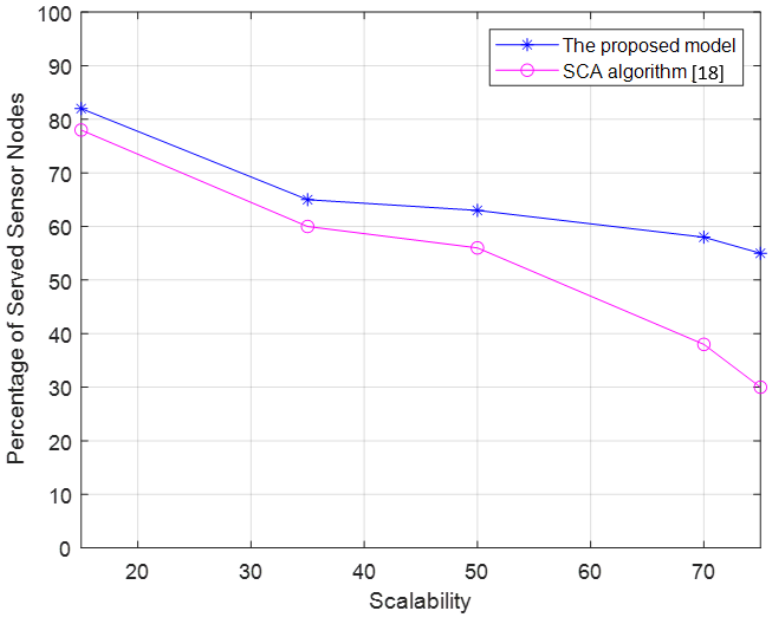


Figure 5-31 The comparison between the proposed SDWSN data gathering method and SCA algorithm [108] on the percentage of served sensor nodes versus scalability.

5.7 Overall Modelling Discussion

Following the testing of the proposed SDWSN data gathering model in multiple scenarios using the suggested simulation tools, the output performance of the simulation tools, including served sensor nodes, ground network energy consumption, air-to-Ground packet delivery, and average UAV energy consumption, is recorded to evaluate the proposed model. Figure 5-32 depicts the results of the evaluation.

There is a trade-off between UAV propulsion energy consumption and ground network energy cost, as shown in this figure. While the crisp path can offer a cost-efficient path for the UAV and serve a higher percentage of ground sensor nodes, it suggests an inefficient data gathering model for ground sensor nodes due to multi-hop communication protocols in the ground network structure, as exprseed in Figure 5-18. In contrast, while the Bezier curve path design reduces UAV energy efficiency per mission and lowers the percentage of served sensor nodes, it benefits ground network formation in terms of energy efficiency, since the ground network structure, as shown in Figure 5-17, is based on multiple plain architectures such as tree network structure.

To justify the existence of multiple gateway-capable nodes in the ground network, while deploying multiple gateway-capable nodes may deplete the sensor batteries and reduce the survivability of the ground network, this does not imply that a high percentage of node

population should act as gateways at the same time. Based on the proposed model, the distribution of the gateway-capable nodes is bounded to 30% out of the entire network. Moreover, during the softwarisation phase, the most effective gateway-capable nodes are elected as gateways which shows that the remaining gateways that are not nominated as gateways could be dropped to lower functionalities (such as leaf nodes) with lower energy expenditure. The re-orchestration functionality can also provide the advantage of gateway replacement in the event of a gateway failure or battery drop.

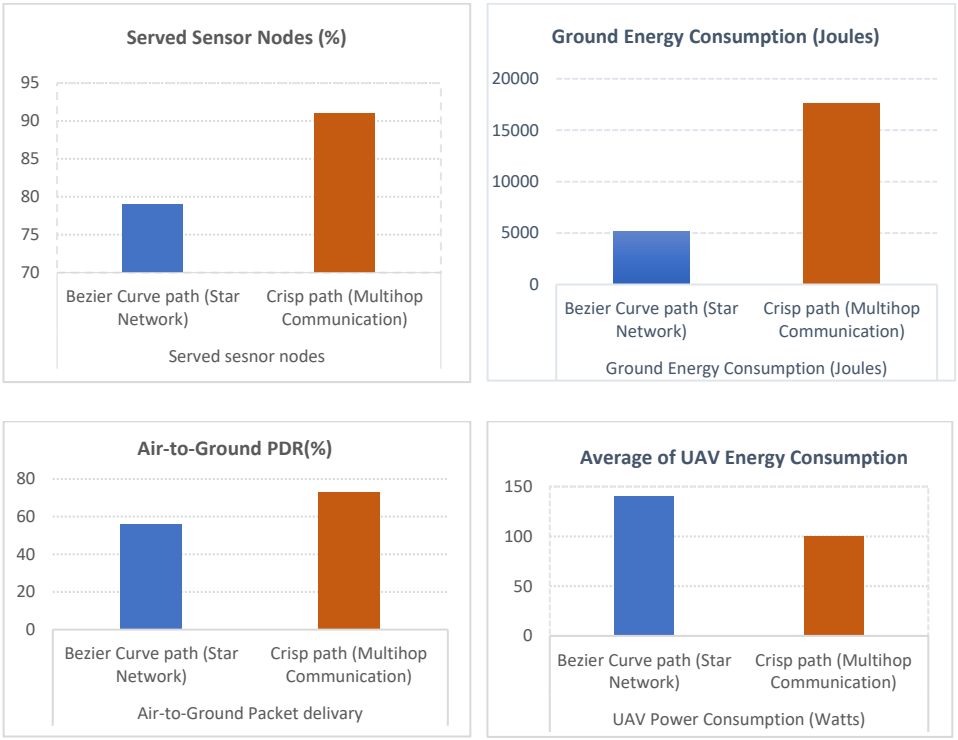


Figure 5-32 The outcomes of multiple scenarios of SDWSN data gathering model outputted from MATLAB, Contiki-Cooja, CupCarbon and Mission Planner.

5.8 Conclusion

This Chapter presented the modelling and simulation of the UAV-aided SDWSN data collection model. The simulation and experiment cases on MATLAB, SITL Mission Planner, Contiki-Cooja, and CupCarbon were described in detail. The UAV fuzzy path model and other components of the UAV energy efficiency were designed and analysed in MATLAB before being validated in SITL Mission Planner. The wireless network capability of dynamically assigning routeing nodes within the fuzzy route improved UAV path energy efficiency. Contiki-Cooja was also used to design and analyse ground network models and main ground network components. CupCarbon was used to outline and evaluate UAV-Ground communication models taking the core components of the

communication model into account. The proposed UAV-enabled SDWSN data gathering model provided a more flexible framework of ground network that could be restructured based on a fitness model to effectively enhance the UAV path design while taking software defined network concept into account.

Several design parameters were discussed and implemented in the experiment case studies namely, the distribution of gateway-capable nodes, the gateway election factor, and the density spread factor. Served sensor nodes, ground network energy consumption, air-to-Ground packet delivery, and average UAV energy consumption are the main performance parameters examined in this chapter. To begin, two samples of network communication without considering the softwarisation concept were considered, followed by multiple scenarios taking the softwarisation concept into account. Results showed varying network performances and UAV path designs can be met following the orchestration process with the SDWSN-enabled UAV data collection concept. Our findings also emphasized that there is a trade-off between the energy consumed by UAV propulsion and the cost of ground network energy. While the smooth path might provide a more energy-efficient path for the UAV, it reflects an inefficient data collection methodology for ground SNs. Alternatively, the Bezier curve path design drops UAV energy efficiency per mission, it highlights a more cost-effective ground network structure.

Furthermore, the proposed ground network modelling using the Contiki-Cooja and CupCarbon simulation tools aimed to indicate the overall system performance while also providing SDWSN design guidelines for a real-world WSN integration system. Also, the co-simulation concept of the SITL Mission Planner, MATLAB, Contiki-Cooja and CupCarbon was viable and could facilitate analysis of the network when complexity takes place as the network evolves.

Chapter 6 Ground physical Network Implementation

6.1 Introduction

The physical implementation of the proposed SDWSN necessitates the ability to dynamically reconfigure the modules, which the proposed module in this thesis can provide. The proposed network structure enables sensor node functionality to be modified over the air via cell phone configurability capability in order to modify some set-ups within module including bit rate, collection rate, packet size, polling intervals, power, etc or to alter the entire functionality of a sensor node from coordinator to leaf node or vice versa.

To begin with, in this section, a network architecture for the physical implementation of distributed SNs is proposed. Following that, a small scale SDWSN capable network is evaluated using the Launchpad SensorTag CC1352R and Development Boards and Raspberry Pi to highlight the reconfigurability concept of modules, followed by a large scale SDWSN actual testing to evaluate the packet delivery ratio and energy performance.

6.2 Proposed Network Architecture for Physical Network Implementation

The physical network system involving softwarisation capability, is implemented using Launchpad SensorTag CC1352R and Development Boards, as well as Raspberry Pi technologies such as the RPi 3B+ in this study. The Launchpad modules provide a dual-band low power radio kit which enables TI 15.4-Stack to be run concurrently with Bluetooth Low Energy (BLE) in a single chip solution. TI 15.4-Stack is an IEEE 802.15.4e/g RF communication stack that provides support for star-topology networks either a Sub-1 GHz (868 MHz/915 MHz) application or a 2.4GHz application. In this thesis, the Sub-1GHz TI 15.4-Stack is chosen for the ground network communication. The reason for this is that the Sub-1 GHz implementation offers several key benefits, including longer range and better protection against in-band interference, as well as the ability to send BLE beacon packets while operating on a Sub-1GHz TI 15.4-Stack network when using dual-band mode.

The Bluetooth Low Energy (BLE) communication protocol offers the client local access to each sensor node's data by utilising dual-band communication functionality on

CC1352R modules. The dual-band functionality also allows us to modify the functionality of a given SensorTag via over-the-air debugging (OAD) and configuration functionality using BLE. Hence, enabling remote configurability features may alleviate maintenance difficulties.

The actual ground network topology system considering the suggested dual-band communication functionality is depicted in Figure 6-1. As shown in this figure, the Launchpad SensorTag CC1352R are configured with the leaf node firmware allowing them to sense and forward the sensing data to the upper-level Coordinators (CC1352R Development Boards). The Gateways, in this architecture, include the Raspberry Pi 3B+ plugged in CC1352R Development Boards. The development boards are configured with the Collector firmware to collect the sensing data from the leaf nodes while the RPi is used for the Gateway data storage and buffering which is connected to development board through the USB cable.

The RPi could also facilitate transferring data (acquired by the Collector-firmware configured within the TI CC1352R development board) to the UAV once it is within communication range. This, however, is not taken into account in this thesis. One advantage of this physical network topology design is on providing the dual-band communication that enables the client to reconfigure each physical node using over-the-air debugging (OAD) functionality that is in-line with the softwarisation concept. As a result, this architecture not only provides a reconfigurability functionality for ground network entities, but it also allows for regular data collection from the ground network.

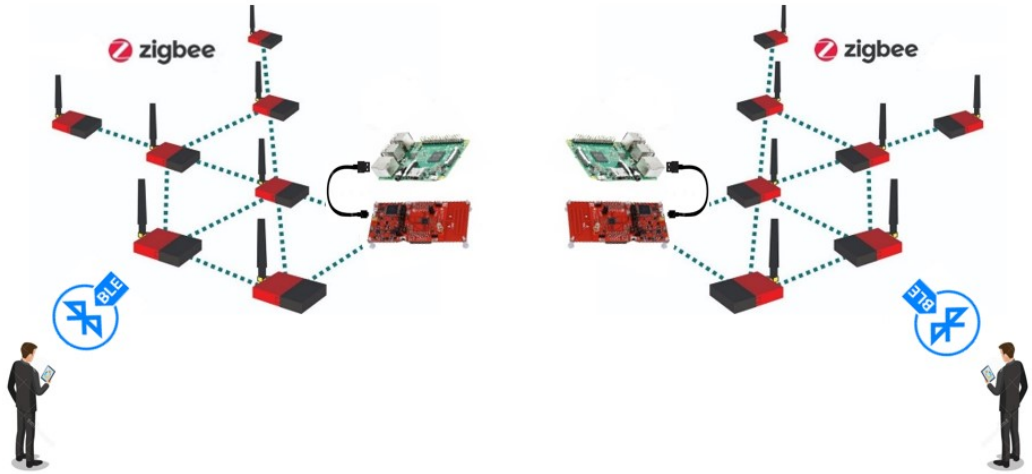


Figure 6-1 Proposed network architecture for physical network implementation.

6.3 Small-scaled Physical Network

In this Section, a pair of examples reflecting the reconfigurability concept based on the dual-band functionality of TI 15.4-Stack and BLE are highlighted. In the following section, the model will be scaled up to a large-scale ground network to test and evaluate network performances such as packet delivery.

6.3.1 The Network of SensorTags and Development Boards

This section focuses on a network of Launchpad modules that transfers sensing data such as temperature, RSSI, humidity, accelerometer, and so on to a development board as part of a data collection effort and modifies the functionality of network entities in real-time as part of a softwarisation effort.

In this testbed design, as depicted in Figure 6-2, two groups of modules including two development boards as well as two SensorTags are chosen as leaf nodes. One development board is also programmed as a coordinator. The SensorTag CC1352R modules are programmed using over-the-air debugging (OAD) functionality via Bluetooth Low Energy (BLE) through a smartphone. The leaf nodes could enable multi-protocol and dual-band communication features over the TI 15.4-Stack and BLE protocols simultaneously. TI 15.4-Stack is used for communication amongst leaf nodes and the coordinator and BLE is used for communication amongst each module and smartphone.

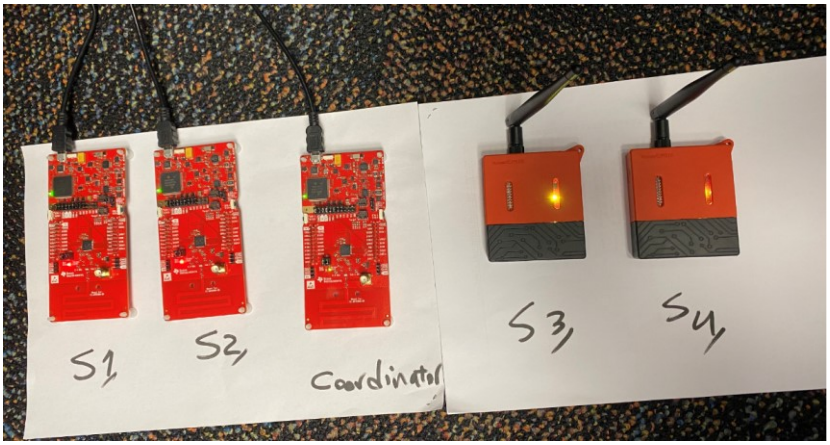


Figure 6-2 The network configuration for two development boards as sensor nodes and two SensorTags as Sensors.

The SimpleLink Starter app for iOS and Android provides extensive support for the CC1352R LaunchPad modules which are used for configurability capability in this thesis.

Hence, this App facilitates the over-the-air debugging (OAD) to modify the functionality of LaunchPad modules. This App uses BLE to connect the LaunchPad modules to a smartphone and supports reading of the LaunchPad button states, controlling LEDs, and receiving real-time sensing data such as RSSI, temperature, humidity, and accelerometer. It also supports cloud connectivity to the IBM Quickstart server or any cloud service via MQTT. This enables a cloud view in which the LaunchPad module can be controlled from any web browser. Figure 6-3 depicts an environment of the Starter application.

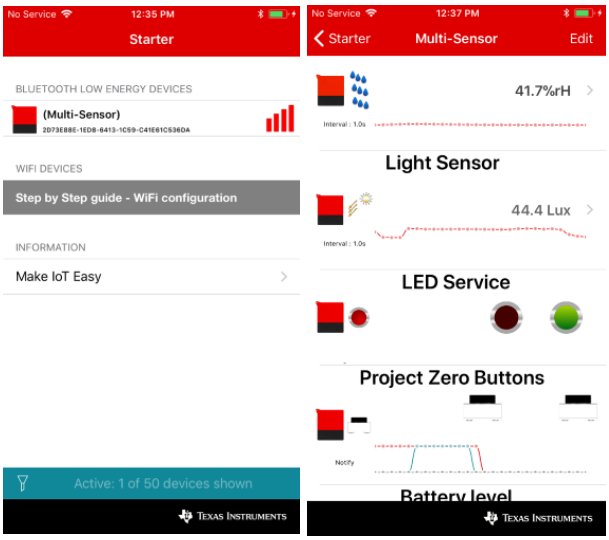


Figure 6-3 Over-the-air debugging (OAD) and sensing reading on the Starter App.

Following installing the SimpleLink Starter App on the smartphone, it will automatically scan for nearby devices via BLE. The LaunchPad SensorTag will advertise its initial configuration to the Starter App (in this case 'Multi Sensor' program is displayed first following running the Starter App). The device can be connected by clicking on the name and choosing 'Sensor View' to get to the GUI. The sensor values following with the button's status are displayed in the App (See Figure 6-2 right). For updating the nodes' functionality through softwarisation, the stack can be modified with OAD capability in Starter App. To this end, a .bin file must first be created in Code Composer Studio (CCS). Following that, the .bin file created in the workspace of CCS must be sent to the smartphone in order to perform OAD from the Starter app to the SensorTag. Following the OAD programming, the updated real-time sensor values and buttons status will be displayed in the mobile app based on the new program.

To program the LPSTK development board as a coordinator, it is required to be debugged via ‘Collector’ program in CCS. Figure 6-4 depicts a part of a main C file in CCS for debugging a collector to print the received sensing data.

```
CUI_statusLinePrintf(csfCuiHndl, deviceStatusLine, "Sensor - Addr=0x%04x,
Temp=%d, RSSI=%d", pSrcAddr->addr.shortAddr, pMsg->tempSensor.ambienceTemp,
rssi, pMsg->accelerometerSensor.xAxis, pMsg->accelerometerSensor.yAxis,

pMsg->accelerometerSensor.zAxis, pMsg->accelerometerSensor.xTiltDet,

pMsg->accelerometerSensor.yTiltDet);
```

Figure 6-4 Main C file in CCS for debugging a collector.

To receive the sensing values from various leaf nodes in a terminal window of coordinator via TI 15.4-Stack, ‘Tera Term’ is used to display the real-time received values. To this end, after connecting the terminal window to the coordinator’s COM port, the UART can be configured as Figure 6-5.

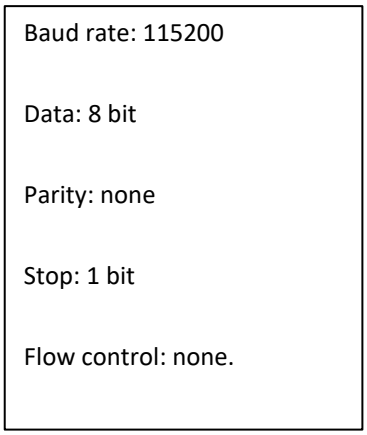


Figure 6-5 UART configuration to receive the sensing data in coordinator.

The received data from multiple leaf nodes in the coordinator’s console are shown in Figure 6-6. The RSSI and temperature readings regarding this experiment are displayed in the coordinator’s console, while the output values can be modified to include light, humidity, and accelerometer sensing data, as suggested in Figure 6-4.

As part of re-configurability process, the development board function can be totally modified from leaf node role to coordinator one or vice versa by OAD functionality in real-time. Other re-configuration process, such as modifications in bit rate, collection rate, packet size, polling intervals, power, and so on, can also be made as part of the over-the-

air real-time re-configurability. In the following section, the re-configurability via modifying the power level is discussed further.

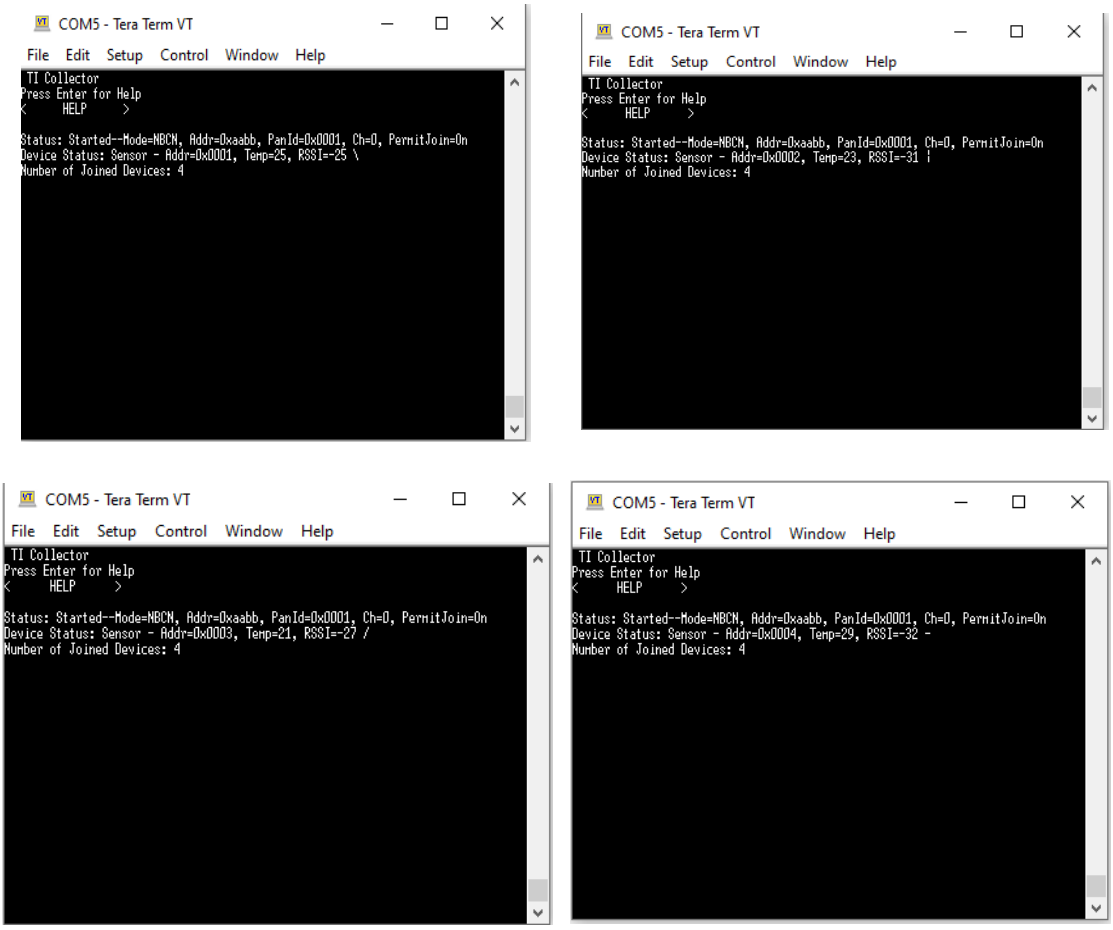


Figure 6-6 The amount of temperature and RSSI received in coordinator from four leaf nodes.

6.3.2 The Network of SensorTags, Development Board and RPi

The proposed testbed for the physical network topology of data collection using RPi 3B+ as a gateway is illustrated in Figure 6-7, in which two SensorTags are programmed as leaf nodes along with one development board as a coordinator. The development board is connected via USB cable to the RPi. A smartphone is also available to either reconfigure the SensorTags via OAD functionality or to monitor the real-time data of each SensorTag. BLE and TI 15.4-Stack communication protocols are used for interaction between smartphone-SensorTags and SensorTags-Development board, respectively.

To have access to the USB port of RPi and display the real-time outcomes on a terminal screen such as ‘Putty’, it is required to write a Python code following the Putty installation on ‘Raspbian’ as bellow:

```
Import serial
ser=serial.Serial('/dev/ttyACM0',115200)
readedText = ser.readline()
print(readedText)
ser.close()
```

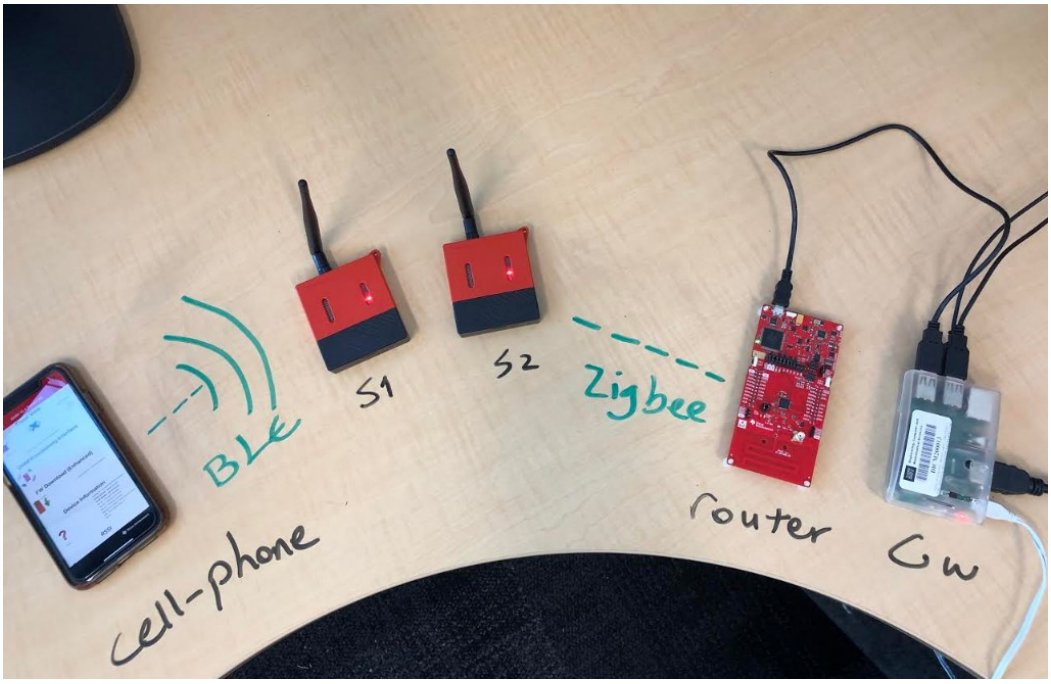


Figure 6-7 The physical network architecture for TI LaunchPad modules and RPi.

After running the code in the terminal window, the results, which include the received sensing data from SensorTags on the development board and then in the RPi, are displayed in the terminal. (See Figure 6-8).

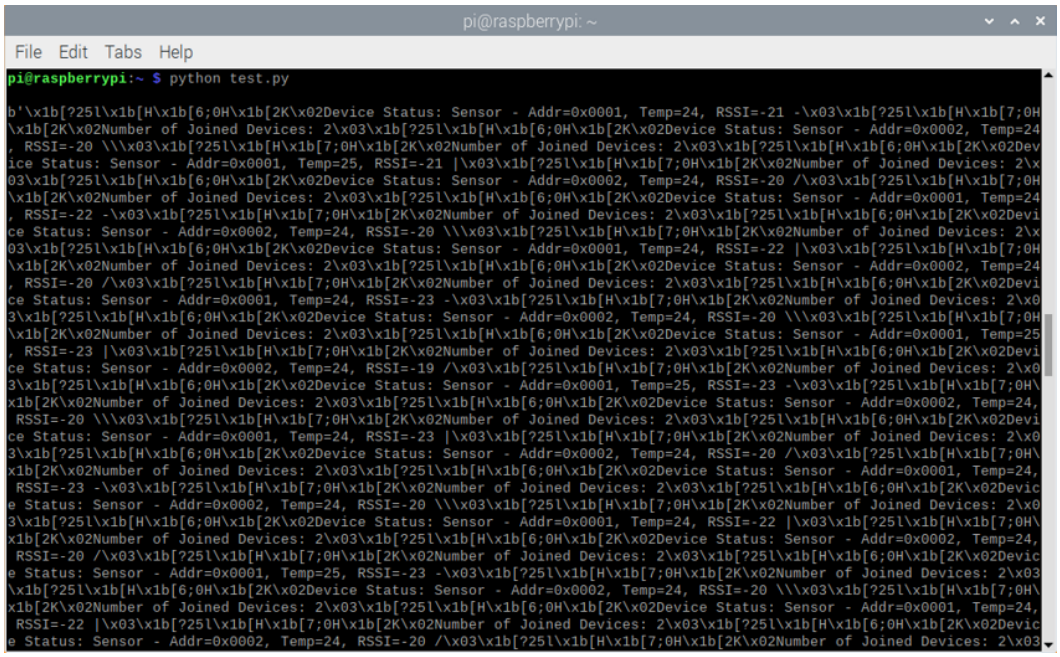


Figure 6-8 The received data from Temp and RSSI of each SensorTags on RPi terminal.

To reconfigure the functionality of the SensorTags, the ‘SysConfig’ GUI feature in CCS offers configurations ranging from Radio to Other Dependencies, as discussed in Section 3.4. Some Physical layer configurations, such as frequency band and data bit rate, are modifiable within the ‘Radio’ configuration. It should be noted that the node functionality can also be manually modified via source code. It is also applicable to perform re-configurability via SysConfig GUI MAC setting through the execution of a TDMA time-synchronized connection between the transmitters and receiver. TDMA MAC enables both communication partners to transfer data quickly at predictable time slots. Note that, in TDMA MAC, the transmitter is not required to send a very long preamble and the receiver is not required to wait and check for a signal on air. This leads to the lowest possible power consumption on both sides. When the broadcasting on the leaf board is started, it starts to send periodic beacon messages. Once the coordinator node receives the first beacon message, it switches the receiver off and goes into Sleep mode state. In this state, it wakes up the receiver just before the next packet from the leaf node is expected.

6.3.3 Re-configurability via Power Modification

Texas Instrument - Real Time Operating System (TI-RTOS) supports the following four power modes:

- Active
- Idle
- Low Power/Standby
- Low Power/Shutdown

In active and idle modes, all resources on the device are available. In active mode, CPU executes the specified codes or waits for interrupts. System resources are available in active mode. The CPU power domain, peripherals, and radio are active. As shown in Table 4-1 Chapter 4, the current consumption is highly dependent on whether the module is in TX or RX mode. The amount of current consumption is in between 20 to 24 mA.

In idle mode, the CPU is in deep sleep and waits for interrupts to wake up. The CPU power is off to reduce the current consumption. The CPU power adds about 25 seconds to the wake-up time. When waking up from idle mode, CPU resumes executing code at the point it left off before entering idle. The application program can use system resources including the high-speed crystal oscillator, the radio, and the peripherals while in idle mode. The amount of current consumption of idle mode is 13 mA. The difference between active and idle mode is on their wake-up time. The CPU power remains on in active mode ensuring a quick wakeup, whereas the wakeup time from idle mode is longer because the CPU must be turned on again.

Low power modes, which include standby and shutdown modes, are optimised for power consumption, and have limited resources available. In Low Power/standby mode, the CPU waits in deep sleep for interrupts to wake up. In Standby mode, system resources, including the high-frequency oscillator, radio, peripherals, and auxiliary domain are not available. Following the waking up from standby, the CPU resumes executing programs at the point before entering standby. Any interrupt-enabled and external input pin can cause the CPU to wake up. The current consumption of Low power/standby mode is 0.6 mA, as shown in Table 4-1 Chapter 4.

In shutdown mode, the CC1352 device shuts off except for the input and output pins. The device can wake up on any input pin. The system completely reboots when waking up from shutdown. In this thesis, it is presumed that the LaunchPad modules have not entered shutdown mode.

To profile the power consumption of the LaunchPad Modules, EnergyTrace is suggested in this thesis as a stand-alone power profiling tool within a debug session in CCS for state monitoring for low-power consumption scenarios. Following debugging of the state programs on leaf node and coordinator modules, it is required to perform provisioning for the calibration of EnergyTrace such as removing all jumpers on the LaunchPad between the XDS debugger and the device except for the $XDS110$ Power, the $3V3$ and RXD jumpers in order to observe the current usage of LaunchPad in EnergyTrace (see Figure 6-9).

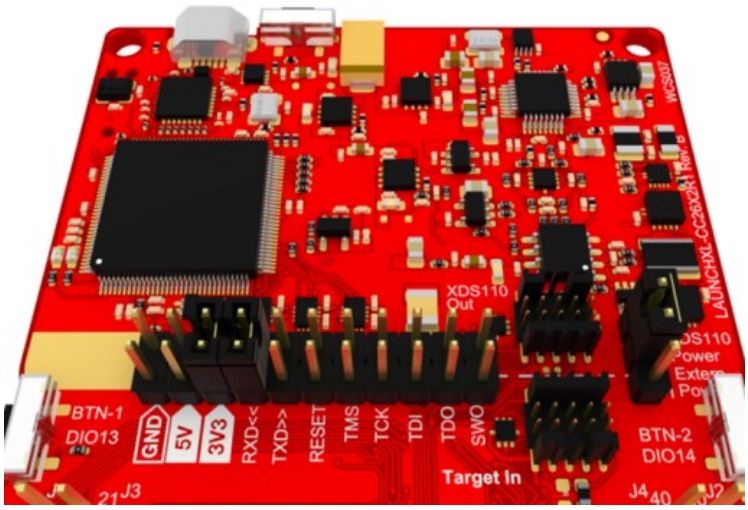


Figure 6-9 Jumper setting of LaunchPAd while using EnergyTrace tool.

Then by utilising the EnergyTrace feature built in CCS, the consumed current is obtained by specifying the packet intervals for data dissemination in each leaf node. Once the data dissemination finished, the current graph will be appeared in application's power profile. Figure 6-10 shows the current consumption profile for each transmitter of the proposed small-scaled network in Figure 6-2, taken over $1\ s$. From the plot, it is easy to identify the packet interval of $500\ ms$ and to verify that the device enters Power-Mode/Standby in-between packets.

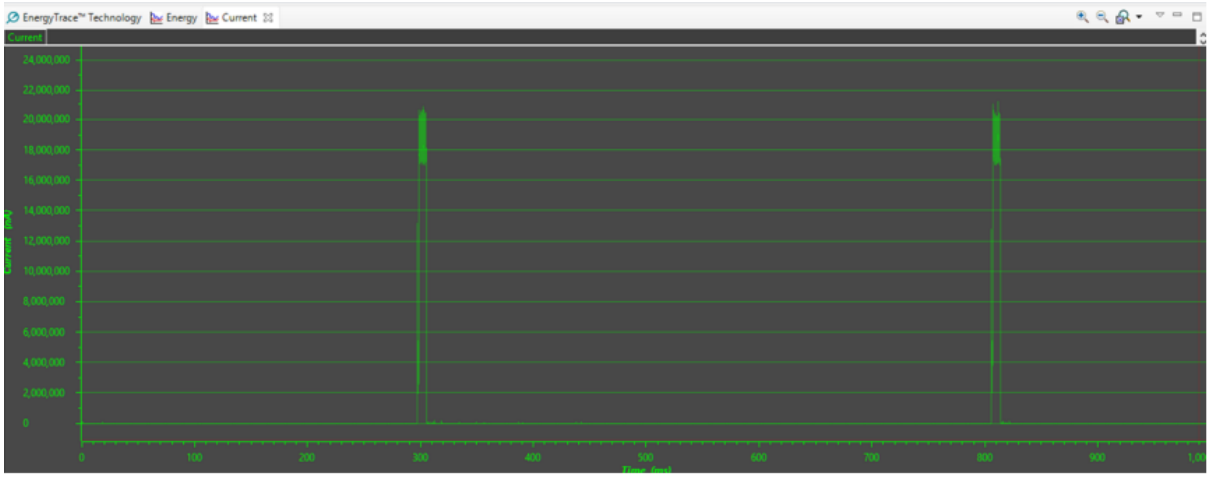


Figure 6-10 Current Consumption for Packet transmission.

The re-configurability process through power is activated by modifying the transmitted and received packet intervals applied on the same physical network structure in Figure 6-2 resulting in altering the current consumption of a RF link. The leaf nodes and coordinators, in the previous case, are set up to wake up two times per second, as shown in Figure 6-10, in every $500\,ms$. To modify the wakeup time of both transmitter and receiver, for instance 10 times per second wakeup time, it is required to set the wake-up interval using WOR_WAKEUPS_PER_SECOND define at the top of the transmitter and receiver source files, as shown in Figure 6-11. This is required to be set to the same values in both the RX and TX sections of source files. Once the wakeup times of both transmitter and receiver are modified, the EnergyTrace tool shows changes in energy usage of both modules. In Section 6.4, the suggested reconfigurability process via modifying the packet intervals is applied on a large physical network to validate the softwarisation concept on an actual network.

```
49 #include <ti/devices/DeviceFamily.h>
50 #include DeviceFamily_constructPath(driverlib/rf_prop_mailbox.h)
51
52 /***** Defines *****/
53 /* Wake-on-Radio wakeups per second */
54 #define WOR_WAKEUPS_PER_SECOND  10
55
56 /* Wake-on-Radio mode. Can be:
57  * - RSSI only
58  * - PQT, preamble detection
59  * - Both, first RSSI and then PQT if RSSI  */
60 #define WOR_MODE CarrierSenseMode_RSSIandPQT
61
62 /* Threshold for RSSI based Carrier Sense in dBm */
63 #define WOR_RSSI_THRESHOLD      ((int8_t)(-111))
64
65 /* Data Rate in use */
66 #define WOR_RF_PHY_DATARATE_50KBPS  0 // 2-GFSK 50Kbps
67 #define WOR_RF_PHY_DATARATE_100KBPS 1 // 2-GFSK 100Kbps
68 #define WOR_RF_PHY_DATARATE_200KBPS 2 // 2-GFSK 200Kbps
69 #define WOR_RF_PHY_DATARATE_300KBPS 3 // 2-GFSK 300Kbps
```

Figure 6-11 Modifying wakeup time in transmitter and receiver source files.

Other re-configuration options for enabling low-power settings in coordinator and leaf nodes to reduce radio propagation power consumption are available in the 'SysConfig' GUI (discussed further in Section 3.4). To this end, in the ‘power section’ of SysConfig GUI, as shown in Figure 6-12, multiple low-power configurations are offered for radio propagation including data and ack, polling only, poll and data, sleep mode radio configurations for both transmitter and receiver. Each of these states defines the type of data transmitted/received by modules during a power test.

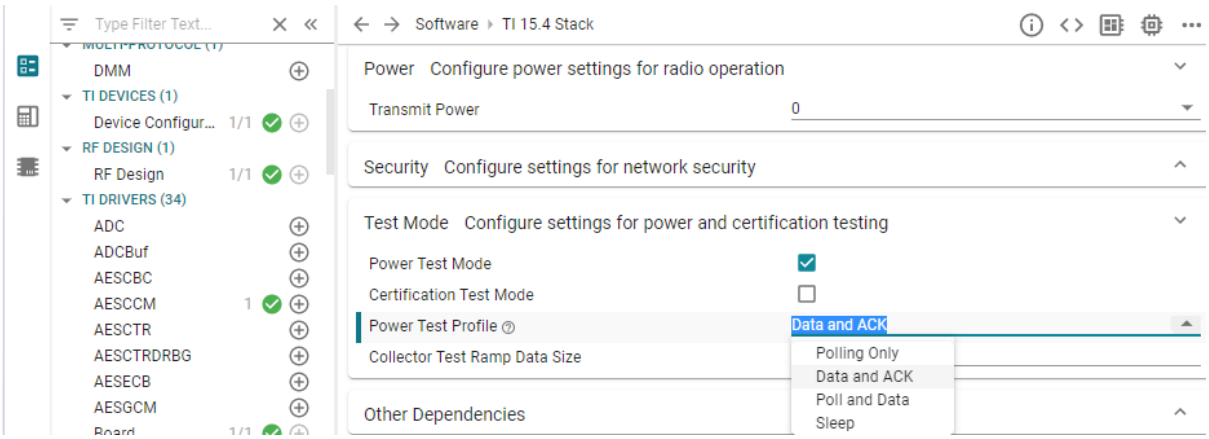


Figure 6-12 Setting multiple low-power configurations through SysConfig GUI.

6.4 Large-scaled Ground Network

In the scaled-up physical network scenario, the post-orchestration data gathering phase is validated through the real-world implementation that can be manipulated to be used in trap applications. As discussed in the Chapter 5, the simulated post-orchestration data gathering phase are structured with the fixed network parameters such as gateway election factor of $\xi = 2/3$ and the distribution of gateway-capable nodes of $D = 20\%$ in order to observe the impact of varying the density spread factor parameter on the network performance. In this section, the same network parameters are used to analyse the physical ground network performance. Hence, out of 20 nodes distributed in the ground, 17 SensorTags are programmed as leaf nodes, while 3 development boards are programmed as coordinators. Each development board is connected via USB cable to a RPi as a gateway. Herein, the physical ground nodes are deployed with the same distribution parameters as defined in the proposed simulation concept in Chapter 5 for post-orchestration phase. The network organisation is based on star network architecture. Also, TI 15.4-Stack communication protocol is used for interaction amongst SensorTags and development board. To validate the softwarisation concept on a real-world scenario, 3 coordinator nodes along with 17 leaf nodes are distributed with various density of

gateway nodes scenarios as shown in Figure 6-13 (σ = 1) and Figure 6-14 (σ = 2). The data gathering time for each density scenario is set to 60 s, which corresponds to the simulation period in Section 5. The message rate is also increased from 1 to 20 msg/sec. To evaluate the communication performance of the proposed physical network model, the packet delivery and power consumption are validated respectively.

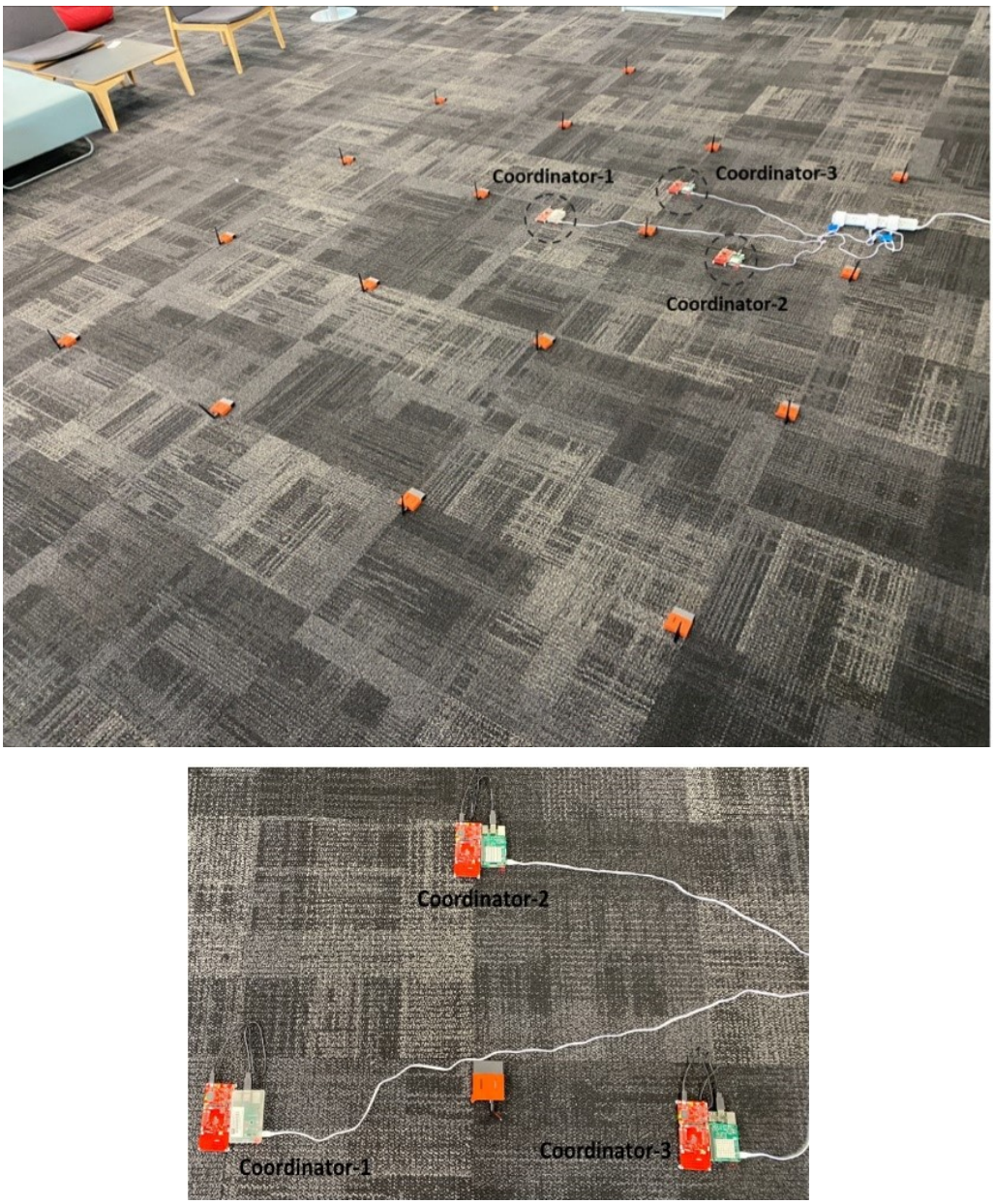


Figure 6-13 The physical ground network formation for a large-scaled network with σ = 1.

Figure 6-14 The physical ground network formation for a large-scaled network with $\sigma = 2$.

The ground physical network communication performances including the energy consumption of the entire network and the percentage of received packets in coordinators based on a various range of gateway sparsity σ and communication message rates are calculated and shown in Figure 6-15 and 6-16.

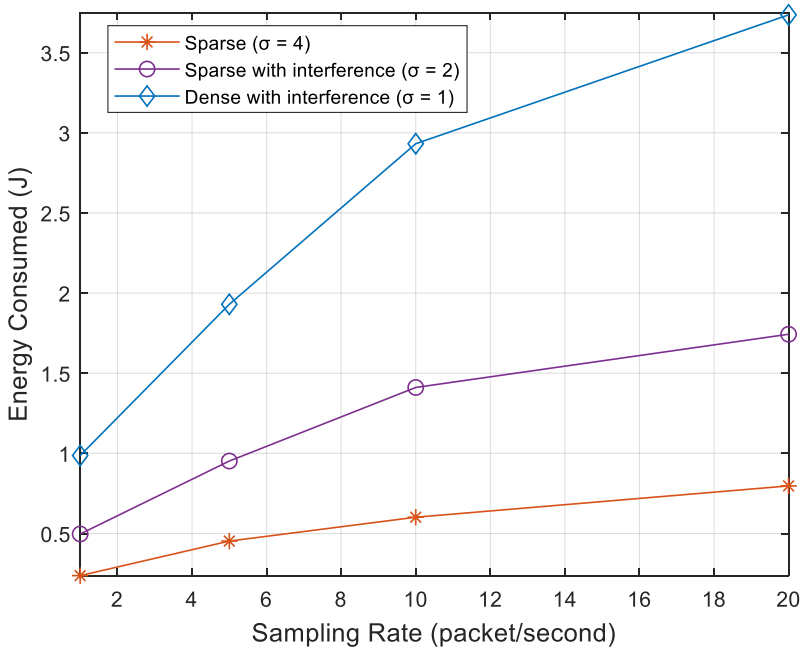


Figure 6-15 Energy consumption of physical ground network in terms of message rate and gateway nodes' spread factor.

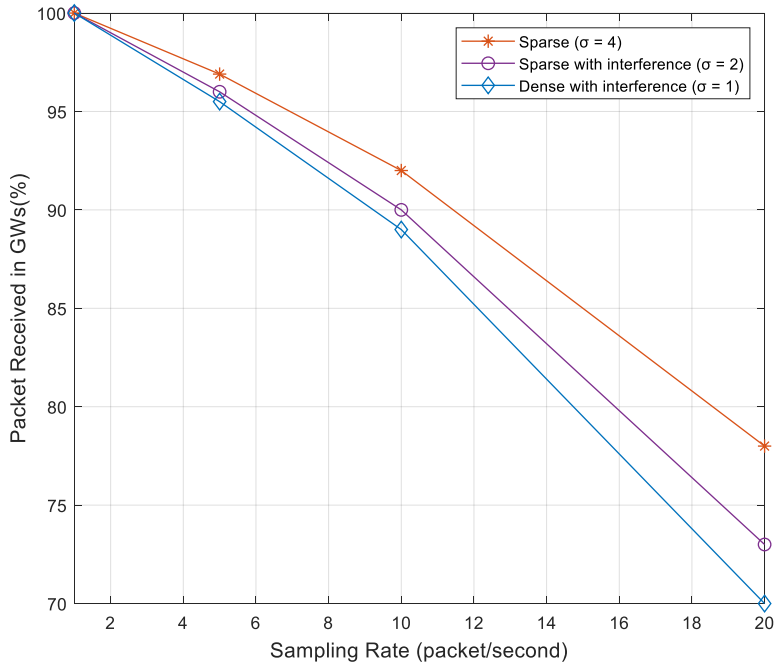


Figure 6-16 Packet received in coordinators (gateways) from distributed leaf nodes in a ground physical network communication.

As shown in these two figures, whenever network density increases, the ground network packet delivery and energy cost for message transmission from leaf nodes (here SensorTags) to coordinator nodes (here development boards) degrade due to increased

interferences from other group members. Furthermore, it is obvious from these figures that as the message rate increases, the percentage of packet delivery in the coordinators' buffers decreases and the energy consumption of ground network increases. The reason for this is that as the message rate increases, higher packet intervals of each network component cause the actual network components to consume more energy while delivering fewer packets to the coordinators.

Furthermore, when comparing the results of physical testbed design implementation in Figures 6-15 and 6-16 with the simulation results in Figures 5-22 and 5-23, it is clear that the physical testbed has lower performance, containing a lower packet delivery rate, and higher energy consumption when compared to the simulation network due to real-world implementation consideration. One reason behind that is the actual ground network testbed implementation suffers from more interference compare with the simulation environment. This includes the interference among the sensor nodes from one side and the interference among sensor nodes and gateways from the other.

6.5 Conclusion

The proposed physical network structure allows for ground node re-configurability by revisiting the related lines of source code to enable the given node to perform with different functionality. This can be achieved by using the over-the-air debugging (OAD) functionality associated with the proposed TI LaunchPad modules in this thesis.

In this section, a network structure for a physical implementation of given nodes on the ground was proposed. A small-scale network testbed was then examined, followed by a large-scale network testbed, using the SensorTag CC1352R LaunchPad, development boards, and Raspberry Pi in order to evaluate network performance such as packet delivery rate and consumed energy. The results of the physical testing were then compared to the results of the simulation testing to validate the proposed concept.

Chapter 7 Conclusion and Future Work

7.1 Introduction

The chapter offers conclusion on the main components and findings of the thesis related to the UAV-aided WSN data collection. Herein, the integration of some of the most recent technologies such as UAV route planning, virtualization and softwarization is presented to provide a flexible data gathering model that can react dynamically to the updated scenarios in the ground network topology. The future work requirements in looking into the various aspects in the research of a WSN data collection based on the UAV such as the possibility of offering local computations through the edge or the fog without being dependent on the cloud are highlighted. Another area of future research is the possibility of proposing an optimal path based on network demands. This forms an important and challenging part of the future work, since it is required to define an optimisation problem to propose the optimal solution considering jointly minimising the UAV energy and SNs consumption taking the SDWSN parameters into account.

7.2 Summary of the Research Findings

This thesis provided the motivational background behind the concept of UAV-based WSN data gathering and UAV's feasibility to dynamically react to updated ground network. Given the problem domain, the core components involved in constructing a flexible, adaptive, and robust UAV-based data collection system that can comply with the network demand are critically discussed and focused on three key parts.

The first part relates to identifying a flexible span so called 'Fuzzy path' for the UAV path design instead of being fixed into a static route and assessing the impact of ground network organisation on the shape of the path. In this case, the choice of UAV flight's dynamic parameters such as path shape, flight speed and acceleration variations could allow wider options and be designed to maximize the travel efficiency while minimizing data packet loss. This organisation enables the UAV path to be dynamically adjusted in accordance with the updated ground network structure. The fuzzy path concept is also based on the capability of switching a gateway-capable node from a leaf node functionality to a gateway one or vice versa. Hence, the proposed topology organization eases the UAV path by involving the efficient and smooth path design within the proposed UAV path fuzzy region and offering the flexibility of adaptive gateway selection closer

to the UAV path. According to the designed system model, multiple operational parameters, such as distribution and spread factor of gateway capable nodes, and gateway election factors, influence the shape of the fuzzy path. Two types of smooth paths are considered as examples of smooth paths taking various formations of ground network into account including crisp circular/linear UAV path and Bezier Curve TSP path aligning on post-orchestration and pre-orchestration network formations, respectively. The UAV speed, acceleration, altitude, and the type of the UAV are some of the influencing parameters on the energy efficiency calculation of the UAV path design within the fuzzy path range. MATLAB and SITL Mission Planner were used as simulation tools, to visualise the analytical and real-world behaviour of designed paths, respectively. The energy model was analysed taking these two paths into account. The evaluation of the proposed UAV path designs revealed that the designed crisp circular/linear UAV path has improved energy efficiency for propulsion energy consumption when compared to the Bezeir Curve TSP with higher packet delivery while compromising ground network energy usage.

The second part relates to the ground network communication structure considering SDWSN concept and network topological orchestration strategies. The SDWSN-enabled ground network communication was split into two main phases: the scanning topological pre-orchestration and sensing data collection post-orchestration phase. During the scanning phase, control information messages were transferred while during the data collection phase, the sensing data information were transferred to the cloud server. As the primary goal of data collection effort is on passing the sensing data information messages, the control information messages have the responsibility to convey the ground network topological orchestration data such as election data from the cloud-level to the ground network and assigning the optimal functionalities to each component of the network. Furthermore, a generic model for ground network energy consumption has been proposed taking the static network topology on the ground into account based on the proposed modules, CC1352 LaunchPads, power parameters. Additionally, the air-to-Ground energy consumption model taking the mobile environment amongst the UAV and ground network representatives was presumed which is highly dependent on the real-time distance between them. To enable communication among the network components, the communication dialogue among the network components was suggested which was based on definition of control/sensing data information messages passed over the cloud. To enable the decision making for the gateway nodes election process, the fitness model

computation was presumed to be executed in the cloud environment facilitating the updated structure of a ground network organisation. In the fitness model election process for the involved potential gateways, the proposed parameters were link quality, energy consumption, and capacity factors. The SDWSN-enable ground communication network was modelled and tested on a virtual platform using the Contiki-Cooja network simulator and MATLAB, serving as a testing ground for various architectures preceding and succeeding the orchestration process was taken place. The analysis reflected the effect of network topology, density, and sampling rate on the energy consumed for the ground network to perform control/sensing data transmission from leaf to the gateway nodes. Hence, the main performance measures used for the analysis were packet delivery and energy consumed for the overall ground network based on various network densities and message rates in the system. The simulation outcomes of both scanning and data gathering phases mentioned above revealed that orchestration plays an important role in serving more SNs on the ground and improving UAV energy efficiency when compared to the situation prior to orchestration. This justified the need for the air-to-Ground communication analysis to be utilized in the proposed model.

The third part relates to the air-to-Ground communication structure and effective communication amongst the UAV and ground network via the proposed communication window of connectivity mechanism. Following the proposed air-to-ground connectivity design, the UAV could connect to ground SNs either indirectly or directly by passing over the given nodes within the fuzzy range. The proposed air-to-Ground communication design were implemented based on two approaches: air-to-Ground communication with each individual sensor nodes or air-to-Ground communication with each group representative nodes as cluster heads. For the first communication approach, following the UAV entrance to the communication window of time per group, the nodes of interest communicate with the UAV while traversing along the field. The minimum Euclidean distances between each step of the UAV path and the SNs belonging to each group were used to calculate the communication window for each group. While in the second communication approach, the UAV has communication with the most favourable gateway nodes out of the potential gateways. The prioritisation for nominating the most favourable gateway nodes was computed based on the proposed fitness model. Once the appropriate gateway for each group was elected and the updated ground structure was structured, the window of connectivity for communication among the UAV and the elected gateway was initiated. This was based on defining a threshold for the $RSSI$ signal

received from each group representative on the UAV receiver. The air-to-Ground communication network was modelled using the Cooja and CupCarbon network simulators for testing of the UAV communication with individual SNs and group representatives, respectively. Two performances here were the UAV's energy consumption and packet delivery to the UAV receiver under various network density scenarios. Also, the impact of UAV velocity and network distribution on the percentage of received packets on the UAV receiver was evaluated. Following the SDWSN-enabled data gathering modelling, the simulation outputs of both scanning and data gathering phases confirmed that orchestration plays a crucial role in improving packet delivery rate on the UAV-Ground connectivity when compared to the situation prior to orchestration.

7.3 Future Work

The contribution accomplished in this thesis requires some future work to be identified for enhancing the overall concept. From the UAV path design point of view, in this thesis, the UAV key parameters such as UAV velocity, acceleration etc. are utilized for modelling and testing of the given ideology. This approach needs to be thoroughly tested considering other environmental scenarios such as the presence of the wind parameter or obstacles as it presents different scenarios for the UAV energy usage that can impact the UAV movement pattern in real-world testing. While the UAV fuzzy path attempted to offer a basis for a flexible and efficient UAV path design, using UAV path minimisation algorithms taking the re-orchestration strategy into account is another future work. It would also be useful to explore the Artificial Intelligence (AI) methods such as utilising Reinforcement Learning (RL) models for the future prediction of the UAV route to enhance the communication interaction with the ground SNs, in parallel with other analytical optimisation solutions that require higher computation capabilities.

From ground network point of view, this thesis focused on the UAV-based data collection over the static WSN on the ground, while considering the mobility impact of ground sensor nodes utilises further analysis of hand-off communication analysis among the mobile sink node and ground sensor nodes from one side and sink nodes and the UAV from the other. In this regard, utilising multiple UAVs wherein assigning each UAV to a specified ground region can be a potential solution for this challenge. In this case, network latency is of significance in evaluating the network performance. Hence, formulation of the end-to-end latency model in data flow from the mobile SN to the UAV station and its incorporation as an additional parameter in optimising ground network routing and UAV

path planning is another viewpoint on the network performance. In Addition, using various ground network organisations following re-orchestration of the ground nodes once one or a group of nodes needs to be replaced due to any faulty situation such as battery issue would be useful to investigate as future work.

Given that the proposed data gathering model is a cloud-based solution, it is worth investigating whether some of the ground re-configurability process could be done locally at the edge of the ground network or through utilising multiple UAVs as edge computing stations rather than sending the information to the cloud servers. The ground group representatives or the UAVs could handle some of the required computation to create the election process. Utilising the decentralised computation approaches not only improves communication latency, but it may also mitigate the risk of central cloud computation failure due to the inaccessibility.

Appendix

A.1 Proof of equation (4-1)

Considering the given specific geographic dimension and gateway capable nodes distribution, $\sqrt{\frac{\sum_{i=1}^{N}(q_0^i-\mu)^2}{N}}$ expresses the standard deviation of gateway-capable nodes (which represents the density of gateway-capable nodes in the network) per cluster. There may be two possibilities for the proposed network structure here:

1. Once the spread factor is less than a specified threshold $\sigma_{gc}(th)$ (Which is highly dependent on the defined distribution of gateway-capable nodes and field dimension), the network considered as condensed.
2. Once the spread factor is more than the threshold then another condition on identifying the density of network should be verified as below:

$$\{\forall i,j \mid D_{i,j} \geq 2 * TR\} \quad (1)$$

This represents the condition in which the Euclidean distance between each two gateway-capable nodes $D_{i,j}$ needs to be higher than twice as defined transmission range TR to avoid any interference between each gateway pair. If there is a pair of gateway-capable nodes found that not following (1), the network is considered as intense one with at least one interference among the gateway-capable nodes. Otherwise, the network is without interference among the gateway-capable nodes and while the spread factor is over the specified threshold, it is considered as a sparse network.

A.2 Ground Network Latency Analytical Model

As the ground network can be flexibly structured based on distributed groups/clusters, with each gateway communicating with the higher level of the system, i.e., UAV, latency analytical modelling can provide better analysis for network simulation. Herein, the topological pre-orchestration scanning phase entails latency within the pre-configuration round, whereas the orchestration phase entails latency within the election outcome notification round.

The main parameters along with their respective notations which influence the latency associated with the re-orchestration process are listed as below:

Table I. The key parameters with their respective notations for the proposed latency model calculation.

Notation	Parameter represented
S_{rate}	Communication message transmission rate (bit/sec) (a designed parameter input in Contiki Cooja)
P_{length}	Size of the packet being transmitted by a node (a designed parameter input in Contiki Cooja)
L_{Prop}	Propagation Latency
L_{TR}	Transmission Latency
$L_{Scanning}$	Latency experienced during the scanning phase
$L_{Notification}$	Latency experienced during notification phase

The exchange of data between the leaf nodes and the gateway nodes can entail propagation and transmission latencies that needs to be taken into consideration [79]. When the gateway node forwards the data to the UAV, the latency occurred by the data transmission from the gateway and the propagation latency between the gateway and UAV are part of the latency of the pre-orchestration phase. Finally, the data forwarded from the UAV to the cloud may also entail latency in data transmission of the UAV and propagation latency between the cloud and the UAV. The resulting transmission latency L_{TR} and propagation latency L_{Prop} are each expressed as:

$$L_{TR} = \frac{P_{length}}{S_{rate}} \quad (1)$$

Wherein P_{length} is the length of a packet/message transmitted by a node which is a designed parameter input for Contiki Cooja. Herein, P_{length} of a router or gateway node can vary depending on the number of connections. S_{rate} is the communication message transmission rate and can be modified within the defined codes of each network component within Contiki Cooja.

$$L_{Prop} = T_{received} - T_{transmission} \quad (2)$$

Wherein L_{Prop} is the difference between time stamps of the message receipt ($T_{received}$) of the destination and transmission ($T_{transmission}$) of the source node in Contiki Cooja once the communication between two network components has taken place.

The latency experienced during the pre-orchestration phase is expressed as:

$$L_{Scanning} = \sum_{n=1}^{N} L_{TR(n)} + L_{Prop(Gateway-Leaf)} + \sum_{m=1}^{M} L_{TR(m)} + L_{Prop(UAV-Gateway)} + L_{TR(UAV)} + L_{Prop(Cloud-UAV)} \quad (3)$$

Wherein N is the total number of leaf nodes, whereas M is the total number of gateway nodes.

The latency experienced during the notification round for the orchestration phase is expressed as:

$$L_{Notification} = L_{TR(Cloud)} + T_{Process(Fitness)} + L_{Prop(UAV-Cloud)} + L_{TR(UAV)} + L_{Prop(Gateway-UAV)} + \sum_{m=1}^{M} L_{TR(m)} + L_{Prop(Leaf-Gateway)} \quad (4)$$

Wherein $T_{Process(Fitness)}$ is considered as a configuration parameter in this work which defines the processing latency caused by running the fitness model for orchestrating the ground network. As the notification of the ground network orchestration takes place, the system can enable the data gathering post-orchestration phase.

It is worth mentioning that the communication latency is supported by the designed packet structure for sensing/control data packets based on Chapter 4. The latency for data gathering post-orchestration phase is calculated the same as (3) taking the packet length and the number of hops into consideration. Herein, as the structure of the ground network

post-orchestration can be based on multi-hop approach, L_{Prop} may vary depending on the number of hops that the packet travels from source to destination.

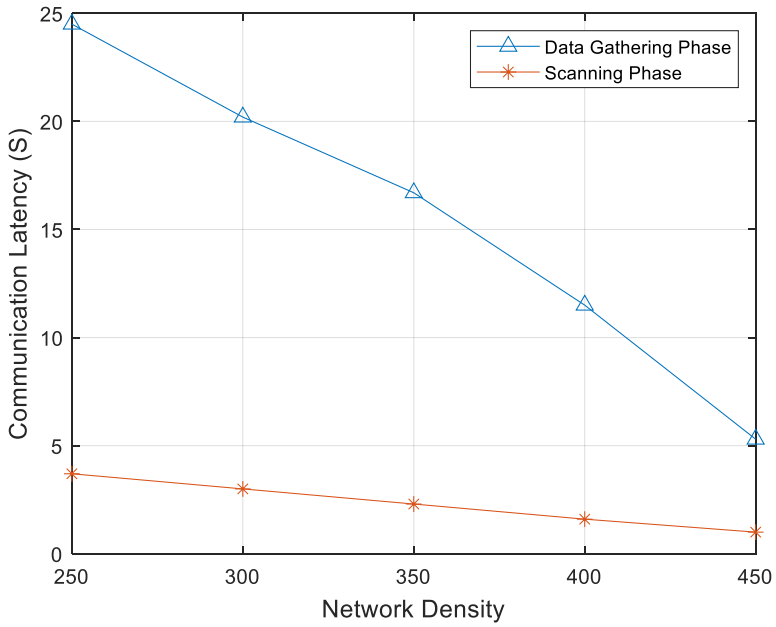


Figure 1 The communication latency for message transmission from leaf to the cloud via the UAV based on various network densities.

The communication latency associated with the phases of pre-orchestration and post-orchestration based on various network densities is represented in Figure 1. Herein, the output of Contiki Cooja scenarios represented by the time stamps is supported by the proposed latency analytical model wherein the pre-orchestration phase experiences less communication latency due to the proposed packet structure design as well as the type of ground network topology. In summary, the orchestration process through configuring the network roles is an efficient way to improve the network functionality as this process costs less with lower communication latency and at the same time higher packet delivery rates.

A.3 a sample code for Leaf node considering control packet frame design in Contiki-Cooja

```
#include "contiki.h"
//#include "cpu.h"
#include "sys/etimer.h"
#include "sys/rtimer.h"
#include "dev/leds.h"
//#include "dev/uart.h"
#include "dev/button-sensor.h"
//#include "dev/adc-sensor.h"
#include "dev/watchdog.h"
#include "dev/serial-line.h"
//#include "dev/sys-ctrl.h"
#include "net/rime/broadcast.h"
#include <stdio.h>
#include <stdint.h>
#include <math.h>
/*---------------------------------------------------------------------------*/
#define LOOP_INTERVAL       CLOCK_SECOND
#define LEDS_OFF_HYSTERISIS (RTIMER_SECOND >> 1)
#define LEDS_PERIODIC       LEDS_YELLOW
#define LEDS_BUTTON         LEDS_RED
#define LEDS_SERIAL_IN      LEDS_ORANGE
#define LEDS_REBOOT         LEDS_ALL
#define LEDS_RF_RX          (LEDS_YELLOW | LEDS_ORANGE)
#define BROADCAST_CHANNEL   129
#define N_DECIMAL_POINTS_PRECISION (100)
/*---------------------------------------------------------------------------*/
```

```
static struct etimer et;
static struct rtimer rt;
static uint16_t counter;
static uint16_t counter1;
static uint16_t node_ID = 1;
int Transmit_Flag = 0;
int i = 0;
static uint16_t a[3];
float battery_dbl;
float latitude_dbl;
float longitude_dbl;
float rssi_dbl;
int batteryintpart;
int batteryintpartbase=97;
int batterydecpart;
int latitudeintpart=-6;
int latitudedecpart;
int longitudeintpart=55;
int longitudedecpart;
int rssiintpart;
int rssidecpart;
static uint16_t c[9];
/*---------------------------------------------------------------------------*/
PROCESS(cc2538_demo_process, "cc2538 demo process");
AUTOSTART_PROCESSES(&cc2538_demo_process);
/*---------------------------------------------------------------------------*/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
```

```
{
  uint16_t *dataptr_temp;
   dataptr_temp= (uint16_t *)packetbuf_dataptr();
/*---------------------------------------------------------------------------*/
static const struct broadcast_callbacks bc_rx = { broadcast_recv };
static struct broadcast_conn bc;
/*---------------------------------------------------------------------------*/
void
rt_callback(struct rtimer *t, void *ptr)
{
  leds_off(LEDS_PERIODIC);
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(cc2538_demo_process, ev, data)
{
  PROCESS_EXITHANDLER(broadcast_close(&bc))
  PROCESS_BEGIN();
  counter = 0;
counter1 = 0;
  broadcast_open(&bc, BROADCAST_CHANNEL, &bc_rx);
    while(1) {
      etimer_set(&et, CLOCK_SECOND*.2);
       PROCESS_YIELD();
                        /**************************************Battery-
Consumption level **********************/
                 battery_dbl=rand();
                batteryintpart = (int)battery_dbl;
                batteryintpart = abs(batteryintpart);
```

```
batteryintpart=(batteryintpart % 2);

batteryintpart=batteryintpart+batteryintpartbase;

batterydecpart=
((int)(battery_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_P
RECISION);

batterydecpart = abs(batterydecpart);

/*************************************************************/

/********************************************************** FOR
Latitude*****************/

latitude_dbl = rand();

latitudedecpart=
((int)(latitude_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_P
RECISION);

latitudedecpart=abs(latitudedecpart);

/**********************************************************
For Longitude*****************/

longitude_dbl = rand();

longitudedecpart=
((int)(longitude_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_
PRECISION);

longitudedecpart=abs(longitudedecpart);

/*************************************************************/

/************************************************FOR
RSSI*****************************/

rssi_dbl=rand();

rssiintpart = (int)rssi_dbl;

rssiintpart=abs(rssiintpart);

rssiintpart= (rssiintpart % 98);

rssiintpart=rssiintpart*(-1);

rssidecpart=
((int)(rssi_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRE
CISION);

rssidecpart=abs(rssidecpart);
```

```
            rssidecpart=0;
            /*************************************************************/
             c[0]=node_ID;
            c[1]=batteryintpart;
            c[2]=latitudeintpart;
            c[3]=rssiintpart;
            c[4]=batterydecpart;
            c[5]=latitudedecpart;
            c[6]=rssidecpart;
            c[7]=longitudeintpart;
            c[8]=longitudedecpart;
    {
            printf("This is end device with node_ID=%d transmitting the following values to the Coordinator.\n", node_ID);
            printf("1. Remaining Battery = '%d.%d' percentage.\n", batteryintpart, batterydecpart);
             printf("2. Latitude value = '%d.%d' degree.\n",latitudeintpart ,latitudedecpart);
            printf("2. Longitude value = '%d.%d' degree.\n",longitudeintpart ,longitudedecpart);
            printf("3. rssi          = '%d.%d' dBm.\n",rssiintpart, rssidecpart);
          packetbuf_copyfrom(&c, sizeof(c));
            broadcast_send(&bc);
counter1++;
printf("The total packets transmitted:%d \n",counter1);
}
}
PROCESS_END();
}
```

A.4 a sample code for Router Level-1 considering control packet frame design in Contiki-Cooja

```
#include "contiki.h"
//#include "cpu.h"
#include "sys/etimer.h"
#include "sys/rtimer.h"
#include "dev/leds.h"
//#include "dev/uart.h"
#include "dev/button-sensor.h"
#include "dev/watchdog.h"
#include "dev/serial-line.h"
//#include "dev/sys-ctrl.h"
#include "net/rime/broadcast.h"
#include <stdio.h>
#include <stdint.h>
#include <math.h>
/*---------------------------------------------------------------------------*/
#define LOOP_INTERVAL       CLOCK_SECOND
#define LEDS_OFF_HYSTERISIS (RTIMER_SECOND >> 1)
#define LEDS_PERIODIC       LEDS_YELLOW
#define LEDS_BUTTON         LEDS_RED
#define LEDS_SERIAL_IN      LEDS_ORANGE
#define LEDS_REBOOT         LEDS_ALL
#define LEDS_RF_RX          (LEDS_YELLOW | LEDS_ORANGE)
#define BROADCAST_CHANNEL   129
#define MAX_NODES             3
#define TIME_IN_SEC           47
#define N_DECIMAL_POINTS_PRECISION (100)
```

```
/*---------------------------------------------------------------------------*/
/*Coordinator address 0xCD, 0xCC*/
/*---------------------------------------------------------------------------*/
static struct etimer et;
static struct rtimer rt;
static uint16_t counter;
static uint16_t count_flag;
static uint16_t a[3];
float batteryrouter_dbl;
float latitude_dbl;
float longitude_dbl;
//signed short rssi;
float rssi_dbl;
int batteryrouterintpart;
int batteryrouterintpartbase=30;
int batteryrouterdecpart;
int latitudeintpart=12;
int latitudedecpart;
int longitudeintpart=44;
int longitudedecpart;
//int tempdecpart_int;
int rssiintpart;
int rssidecpart;
int maxcapacity=4;
int current_number_of_connected_nodes=3;
int current_level=1;
short signed battery[16]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

```
short signed rssi[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed batterydec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed rssidec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed values_to_be_transmitted[20];
int x;
int y;
static uint16_t counter1=0;
/*---------------------------------------------------------------------------*/
PROCESS(cc2538_demo_process, "cc2538 demo process");
AUTOSTART_PROCESSES(&cc2538_demo_process);
/*---------------------------------------------------------------------------*/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
 int16_t *dataptr_temp1;
 dataptr_temp1= (int16_t *)packetbuf_dataptr();
   x =dataptr_temp1[0];
  battery[x] =dataptr_temp1[1];
  latitude[x] =dataptr_temp1[2];
  rssi[x] =dataptr_temp1[3];
  batterydec[x] =dataptr_temp1[4];
  latitudedec[x] =dataptr_temp1[5];
  rssidec[x] =dataptr_temp1[6];
  longitude[x] =dataptr_temp1[7];
longitudedec[x] =dataptr_temp1[8];
```

```
counter1++;
}
/*---------------------------------------------------------------------------*/
static const struct broadcast_callbacks bc_rx = { broadcast_recv };
static struct broadcast_conn bc;
/*---------------------------------------------------------------------------*/
void
rt_callback(struct rtimer *t, void *ptr)
{
  leds_off(LEDS_PERIODIC);
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(cc2538_demo_process, ev, data)
{
  PROCESS_EXITHANDLER(broadcast_close(&bc))
  PROCESS_BEGIN();
  counter = 0;
  count_flag=0;
  broadcast_open(&bc, BROADCAST_CHANNEL, &bc_rx);
  while(1) {
   if(count_flag==0)
{
etimer_set(&et, CLOCK_SECOND);
count_flag=1;
}
    PROCESS_YIELD();
/*****************************************************FOR
battery*************************************************************/
```

```
batteryrouter_dbl=rand();

batteryrouterintpart = (int)batteryrouter_dbl;

batteryrouterintpart = abs(batteryrouterintpart);

batteryrouterintpart=(batteryrouterintpart % 2);

//batteryintpart=<batteryintpart;

batteryrouterintpart=batteryrouterintpart+batteryrouterintpartbase;

batteryrouterdecpart=
((int)(batteryrouter_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRECISION);

batteryrouterdecpart = abs(batteryrouterdecpart);


/**********************************************************************
**********************************************************************/

/*********************************Latitude*****************************
**********************/

 latitude_dbl = rand();

// latitudeintpart = (int)latitude_dbl;

 //latitudeintpart=abs(latitudeintpart);

// latitudeintpart = (latitudeintpart % );

latitudedecpart=
((int)(latitude_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRECISION);

latitudedecpart=abs(latitudedecpart);

/**********************************************************************
*****************************For longitude*****************************/

longitude_dbl = rand();

// longitudeintpart = (int)longitude_dbl;

 //longitudeintpart=abs(longitudeintpart);

 //longitudeintpart = (longitudeintpart % 40);

longitudedecpart=
((int)(longitude_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRECISION);
```

```
            longitudedecpart=abs(longitudedecpart);

/**********************************************************************
**********************************************************************
***/

            /********************************************************** FOR
RSSI**********************************************************************/

            rssi_dbl=rand();

            rssiintpart = (int)rssi_dbl;

            rssiintpart=abs(rssiintpart);

            rssiintpart= (rssiintpart % 98);

            rssiintpart=rssiintpart*(-1);          .

            rssidecpart=
((int)(rssi_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRE
CISION);

            rssidecpart=abs(rssidecpart);

            rssidecpart=0;

/**********************************************************************
**********************************************************************
*****/

   if(ev == PROCESS_EVENT_TIMER) {

    leds_on(LEDS_PERIODIC);

        counter++;

          //somewhere is here do the serial output

           etimer_set(&et, CLOCK_SECOND*.0125);

            if(counter==4)

                {

           counter=(counter % 4);

           printf("Values reported by Router-L1 are as follows:\n");

//printf("This is end device with node_ID=%d transmitting the following values to the
Coordinator.\n", node_ID);
```

```
printf("1. Remaining Battery = '%d.%d' percentage.\n", batteryrouterintpart, batteryrouterdecpart);

printf("2. Latitude value = '%d.%d' degree.\n",latitudeintpart ,latitudedecpart);

printf("3. Longitude value = '%d.%d' degree.\n",longitudeintpart ,longitudedecpart);

printf("4. rssi = '%d.%d' dBm.\n",rssiintpart, rssidecpart);

printf("5. Maximum capacity = '%d' nodes.\n", maxcapacity);

printf("6. Current number of connected nodes = '%d' nodes.\n",current_number_of_connected_nodes);

printf("7. Current level = Level '%d'.\n", current_level);

for (x=1;x<=4;x++)

{

printf("Node %d: %d.%d,%d.%d,%d.%d,%d|", x, battery[x],batterydec[x], latitude[x], latitudedec[x],longitude[x],longitudedec[x], rssi[x]);

values_to_be_transmitted[0]=x;

values_to_be_transmitted[1]=battery[x];

values_to_be_transmitted[2]=latitude[x];

values_to_be_transmitted[3]=rssi[x];

values_to_be_transmitted[4]=batterydec[x];

values_to_be_transmitted[5]=latitudedec[x];

values_to_be_transmitted[6]=rssidec[x];

values_to_be_transmitted[7]=longitude[x];

values_to_be_transmitted[8]=longitudedec[x];

values_to_be_transmitted[9]=batteryrouterintpart;

values_to_be_transmitted[10]=batteryrouterdecpart;

values_to_be_transmitted[11]=latitudeintpart;

values_to_be_transmitted[12]=latitudedecpart;

values_to_be_transmitted[13]=longitudeintpart;

values_to_be_transmitted[14]=longitudedecpart;
```

```
values_to_be_transmitted[15]=rssiintpart;
values_to_be_transmitted[16]=rssidecpart;
values_to_be_transmitted[17]=maxcapacity;
values_to_be_transmitted[18]=current_number_of_connected_nodes;
values_to_be_transmitted[19]=current_level;
packetbuf_copyfrom(&values_to_be_transmitted, sizeof(values_to_be_transmitted));
broadcast_send(&bc);
}
printf("\n");
printf("The total packets received:%d \n",counter1);
            etimer_set(&et, CLOCK_SECOND*.0125);
                }
              }
  }
  PROCESS_END();
}
```

A.5 a sample code for Router Level-2 considering control packet frame design in Contiki-Cooja

```
#include "contiki.h"
#include "sys/etimer.h"
#include "sys/rtimer.h"
#include "dev/leds.h"
#include "dev/button-sensor.h"
#include "dev/watchdog.h"
#include "dev/serial-line.h"
#include "net/rime/broadcast.h"
#include <stdio.h>
#include <stdint.h>
#include <math.h>
/*---------------------------------------------------------------------------*/
#define LOOP_INTERVAL       CLOCK_SECOND
#define LEDS_OFF_HYSTERISIS (RTIMER_SECOND >> 1)
#define LEDS_PERIODIC       LEDS_YELLOW
#define LEDS_BUTTON         LEDS_RED
#define LEDS_SERIAL_IN      LEDS_ORANGE
#define LEDS_REBOOT         LEDS_ALL
#define LEDS_RF_RX          (LEDS_YELLOW | LEDS_ORANGE)
#define BROADCAST_CHANNEL   129
#define MAX_NODES             3
#define TIME_IN_SEC           47
#define N_DECIMAL_POINTS_PRECISION (100)
/*---------------------------------------------------------------------------*/
/*Coordinator address 0xCD, 0xCC*/
/*---------------------------------------------------------------------------*/
```

```
static struct etimer et;
static struct rtimer rt;
static uint16_t counter;
static uint16_t count_flag;
static uint16_t a[3];
short signed battery[16]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed rssi[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed batterydec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed rssidec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed values_to_be_transmitted[51];
int x;
int y;
int batteryrouterintpart;
int batteryrouterintpartbase;
int batteryrouterdecpart;
int latitudeintpart;
int latitudedecpart;
int longitudeintpart;
int longitudedecpart;
//int tempdecpart_int;
int rssiintpart;
int rssidecpart;
int maxcapacity;
int current_number_of_connected_nodes;
```

```
int current_level;
////Router-2 sensed variants//////////
float batteryrouter2_dbl;
float latitude2_dbl;
float longitude2_dbl;
float rssi2_dbl;
int batteryrouter2intpart;
int batteryrouter2intpartbase=40;
int batteryrouter2decpart;
int latitude2intpart=37;
int latitude2decpart;
int longitude2intpart=33;
int longitude2decpart;
int rssi2intpart;
int rssi2decpart;
int maxcapacity2=4;
int current_number_of_connected_nodes2=2;
int current_level2=2;
/*---------------------------------------------------------------------------*/
PROCESS(cc2538_demo_process, "cc2538 demo process");
AUTOSTART_PROCESSES(&cc2538_demo_process);
/*---------------------------------------------------------------------------*/
/*************************************************************************
*************************************************************************
*************************************/
/*Reception function starts here*/
/*************************************************************************
*************************************************************************
*************************************/
static void
```

```
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)

{

 int16_t *dataptr_temp1;

   //leds_toggle(LEDS_RF_RX);          //Commenting this particular line would disable the LEDs. This can  minimize the current consumption of the CC2538 chip.

   dataptr_temp1= (int16_t *)packetbuf_dataptr();

     x =dataptr_temp1[0];

  battery[x] =dataptr_temp1[1];

  latitude[x] =dataptr_temp1[2];

  rssi[x] =dataptr_temp1[3];

  batterydec[x] =dataptr_temp1[4];

  latitudedec[x] =dataptr_temp1[5];

  rssidec[x] =dataptr_temp1[6];

  longitude[x] =dataptr_temp1[7];

  longitudedec[x] =dataptr_temp1[8];

//////////////////receiving the sensed amount from router-1 ////////////

  batteryrouterintpart =dataptr_temp1[9];

  batteryrouterdecpart =dataptr_temp1[10];

  latitudeintpart =dataptr_temp1[11];

  latitudedecpart =dataptr_temp1[12];

  longitudeintpart =dataptr_temp1[13];

  longitudedecpart =dataptr_temp1[14];

  rssiintpart =dataptr_temp1[15];

  rssidecpart =dataptr_temp1[16];

  maxcapacity =dataptr_temp1[17];

  current_number_of_connected_nodes =dataptr_temp1[18];

  current_level =dataptr_temp1[19];

}
```

```
/*************************************************************************
*************************************************************************
****************************************/

/*Reception function ends here*/

/*************************************************************************
*************************************************************************
****************************************/

/*---------------------------------------------------------------------*/

static const struct broadcast_callbacks bc_rx = { broadcast_recv };

static struct broadcast_conn bc;

/*---------------------------------------------------------------------*/

void

rt_callback(struct rtimer *t, void *ptr)

{

 leds_off(LEDS_PERIODIC);

}

/*---------------------------------------------------------------------*/

PROCESS_THREAD(cc2538_demo_process, ev, data)

{

 PROCESS_EXITHANDLER(broadcast_close(&bc))


 PROCESS_BEGIN();

 counter = 0;   //Counter initialization

 count_flag=0;

 broadcast_open(&bc, BROADCAST_CHANNEL, &bc_rx);

 while(1) {

  if(count_flag==0)

{

etimer_set(&et, CLOCK_SECOND);

count_flag=1;
```

```
}

   PROCESS_YIELD();

/*********************************************************** FOR
battery*****************************************************************/

              batteryrouter2_dbl=rand();

               batteryrouter2intpart = (int)batteryrouter2_dbl;

               batteryrouter2intpart = abs(batteryrouter2intpart);

               batteryrouter2intpart=(batteryrouter2intpart % 2);

//batteryintpart=<batteryintpart;

               batteryrouter2intpart=batteryrouter2intpart+batteryrouter2intpartbase;

               batteryrouter2decpart=
((int)(batteryrouter2_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POI
NTS_PRECISION);

               batteryrouter2decpart = abs(batteryrouter2decpart);

/*********************************************************************
*********************************************************************
*******/

/*********************************************************** FOR
Latitude*****************************************************************
***/

                latitude2_dbl = rand();

              // latitudeintpart = (int)latitude_dbl;

               //latitudeintpart=abs(latitudeintpart);

              // latitudeintpart = (latitudeintpart % );

               latitude2decpart=
((int)(latitude2_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_
PRECISION);

               latitude2decpart=abs(latitude2decpart);

/*********************************************************************
**************For Longitude*****************************************/

longitude2_dbl = rand();

              // longitudeintpart = (int)longitude_dbl;
```

```
            //longitudeintpart=abs(longitudeintpart);

            //longitudeintpart = (longitudeintpart % 40);

            longitude2decpart=
((int)(longitude2_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS
_PRECISION);

            longitude2decpart=abs(longitude2decpart);


/*****************************************************************
*************************************************************/

            /************************************************** FOR
RSSI*************************************************************/

            rssi2_dbl=rand();

            rssi2intpart = (int)rssi2_dbl;

            rssi2intpart=abs(rssi2intpart);

            rssi2intpart= (rssi2intpart % 98;

            rssi2intpart=rssi2intpart*(-1);        // Since RSSI values are always negative.

            rssi2decpart=
((int)(rssi2_dbl*N_DECIMAL_POINTS_PRECISION)%N_DECIMAL_POINTS_PRE
CISION);

            rssi2decpart=abs(rssi2decpart);

            rssi2decpart=0;

/*****************************************************************
*************************************************************
*****/

   if(ev == PROCESS_EVENT_TIMER) {

    leds_on(LEDS_PERIODIC);

        counter++;

//somewhere is here do the serial output

/*********************************************************Time interval for
polling                    can                    be                    changed
here*************************************************************
************************/

            etimer_set(&et, CLOCK_SECOND*.25);
```

```
/**************************************************Packets are transmitted (broadcasted) using this function**************************************************************************/

        if(counter==4) //put any number . //How many end devices are present in the star network?

            {

        counter=(counter % 4);  //Which 'operator' should be used here so as to set the value of the counter to 0?. //it has to be same number as above within the counter

        printf("Values reported by Router-L2 are as follows:\n");

printf("Values traversed from Router-L1 are as follows:\n");

printf("1. Remaining Battery = '%d.%d' percentage.\n", batteryrouterintpart, batteryrouterdecpart);

         printf("2. Latitude value = '%d.%d' degree.\n",latitudeintpart ,latitudedecpart);

        printf("3. Longitude value = '%d.%d' degree.\n",longitudeintpart ,longitudedecpart);

        printf("4. rssi          = '%d.%d' dBm.\n",rssiintpart, rssidecpart);

printf("5. Maximum capacity = '%d' nodes.\n", maxcapacity);

printf("6. Current number of connected nodes = '%d' nodes.\n",current_number_of_connected_nodes);

printf("7. Current level = Level '%d'.\n", current_level);

printf("Values specified to Router-L2 are as follows:\n");

printf("1. Remaining Battery = '%d.%d' percentage.\n", batteryrouter2intpart, batteryrouter2decpart);

         printf("2. Latitude value = '%d.%d' degree.\n",latitude2intpart ,latitude2decpart);

        printf("3. Longitude value = '%d.%d' degree.\n",longitude2intpart ,longitude2decpart);

        printf("4. rssi          = '%d.%d' dBm.\n",rssi2intpart, rssi2decpart);

printf("5. Maximum capacity = '%d' nodes.\n", maxcapacity2);

printf("6. Current number of connected nodes = '%d' nodes.\n",current_number_of_connected_nodes2);

printf("7. Current level = Level '%d'.\n", current_level2);
```

```
for (x=1;x<=4;x++)   //x<=the number that is mentioned within the counter

{

printf("Node %d: %d.%d,%d.%d,%d.%d,%d|", x, battery[x],batterydec[x], latitude[x], latitudedec[x],longitude[x],longitudedec[x], rssi[x]);

}

for (x=1;x<=4;x++)   //x<=the number that is mentioned within the counter

{

values_to_be_transmitted[20]=x;

values_to_be_transmitted[21]=battery[x];

values_to_be_transmitted[22]=latitude[x];

values_to_be_transmitted[23]=rssi[x];

values_to_be_transmitted[24]=batterydec[x];

values_to_be_transmitted[25]=latitudedec[x];

values_to_be_transmitted[26]=rssidec[x];

values_to_be_transmitted[27]=longitude[x];

values_to_be_transmitted[28]=longitudedec[x];

////////////////transmitting the router-1 sensing data/////

values_to_be_transmitted[29]=batteryrouterintpart;

values_to_be_transmitted[30]=batteryrouterdecpart;

values_to_be_transmitted[31]=latitudeintpart;

values_to_be_transmitted[32]=latitudedecpart;

values_to_be_transmitted[33]=longitudeintpart;

values_to_be_transmitted[34]=longitudedecpart;

values_to_be_transmitted[35]=rssiintpart;

values_to_be_transmitted[36]=rssidecpart;

values_to_be_transmitted[37]=maxcapacity;

values_to_be_transmitted[38]=current_number_of_connected_nodes;

values_to_be_transmitted[39]=current_level;
```

```
//////////////transmitting the router-2 sensing data//////
values_to_be_transmitted[40]=batteryrouter2intpart;
values_to_be_transmitted[41]=batteryrouter2decpart;
values_to_be_transmitted[42]=latitude2intpart;
values_to_be_transmitted[43]=latitude2decpart;
values_to_be_transmitted[44]=longitude2intpart;
values_to_be_transmitted[45]=longitude2decpart;
values_to_be_transmitted[46]=rssi2intpart;
values_to_be_transmitted[47]=rssi2decpart;
values_to_be_transmitted[48]=maxcapacity2;
values_to_be_transmitted[49]=current_number_of_connected_nodes2;
values_to_be_transmitted[50]=current_level2;
////////////////////////////////////////////////
packetbuf_copyfrom(&values_to_be_transmitted, sizeof(values_to_be_transmitted));
broadcast_send(&bc);
}
printf("\n");
            etimer_set(&et, CLOCK_SECOND*.25);
                  }
      }
  }

  PROCESS_END();
}
```

A.6 a sample code for Coordinator considering control packet frame design in Contiki-Cooja

```
#include "contiki.h"
#include "sys/etimer.h"
#include "sys/rtimer.h"
#include "dev/leds.h"
#include "dev/button-sensor.h"
#include "dev/watchdog.h"
#include "dev/serial-line.h"
#include "net/rime/broadcast.h"
#include <stdio.h>
#include <stdint.h>
#include <math.h>
/*---------------------------------------------------------------------------*/
#define LOOP_INTERVAL       CLOCK_SECOND
#define LEDS_OFF_HYSTERISIS (RTIMER_SECOND >> 1)
#define LEDS_PERIODIC       LEDS_YELLOW
#define LEDS_BUTTON         LEDS_RED
#define LEDS_SERIAL_IN      LEDS_ORANGE
#define LEDS_REBOOT         LEDS_ALL
#define LEDS_RF_RX          (LEDS_YELLOW | LEDS_ORANGE)
#define BROADCAST_CHANNEL   129
#define MAX_NODES             3
#define TIME_IN_SEC           47
/*---------------------------------------------------------------------------*/
/*Coordinator address 0xCD, 0xCC*/
/*---------------------------------------------------------------------------*/
static struct etimer et;
```

```
static struct rtimer rt;
static uint16_t counter;
static uint16_t count_flag;
static uint16_t a[3];
short signed battery[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed rssi[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed batterydec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed latitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed rssidec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitude[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
short signed longitudedec[16]= [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
int x;
int y;
int batteryrouterintpart;
int batteryrouterintpartbase;
int batteryrouterdecpart;
int latitudeintpart;
int latitudedecpart;
int longitudeintpart;
int longitudedecpart;
int rssiintpart;
int rssidecpart;
int maxcapacity;
int current_number_of_connected_nodes;
int current_level;
////Router-2 sensed variants//////////
float batteryrouter2_dbl;
```

```
float latitude2_dbl;
float longitude2_dbl;
float rssi2_dbl;
int batteryrouter2intpart;
int batteryrouter2intpartbase;
int batteryrouter2decpart;
int latitude2intpart;
int latitude2decpart;
int longitude2intpart;
int longitude2decpart;
int rssi2intpart;
int rssi2decpart;
int maxcapacity2;
int current_number_of_connected_nodes2;
int current_level2;
/*---------------------------------------------------------------------------*/
PROCESS(cc2538_demo_process, "cc2538 demo process");
AUTOSTART_PROCESSES(&cc2538_demo_process);
/*---------------------------------------------------------------------------*/
/*************************************************************************
*************************************************************************
*****************************************/
/*Reception function starts here*/
/*************************************************************************
*************************************************************************
*****************************************/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
 int16_t *dataptr_temp1;
```

```
   //leds_toggle(LEDS_RF_RX);          //Commenting this particular line would disable the LEDs. This can  minimize the current consumption of the CC2538 chip.

   dataptr_temp1= (int16_t *)packetbuf_dataptr();

      x =dataptr_temp1[0];

     battery[x] =dataptr_temp1[1];

  latitude[x] =dataptr_temp1[2];

  rssi[x] =dataptr_temp1[3];

  batterydec[x] =dataptr_temp1[4];

  latitudedec[x] =dataptr_temp1[5];

  rssidec[x] =dataptr_temp1[6];

  longitude[x] =dataptr_temp1[7];

longitudedec[x] =dataptr_temp1[8];

////////////////////////////////////

batteryrouterintpart=dataptr_temp1[9];

batteryrouterdecpart=dataptr_temp1[10];

latitudeintpart=dataptr_temp1[11];

latitudedecpart=dataptr_temp1[12];

longitudeintpart=dataptr_temp1[13];

longitudedecpart=dataptr_temp1[14];

rssiintpart=dataptr_temp1[15];

rssidecpart=dataptr_temp1[16];

maxcapacity=dataptr_temp1[17];

current_number_of_connected_nodes=dataptr_temp1[18];

current_level=dataptr_temp1[19];

//////////////transmitting the router-2 sensing data//////

}

/*************************************************************************
*************************************************************************
*****************************************/
```

```
/*Reception function ends here*/
/*************************************************************************
*************************************************************************
*****************************************/
/*---------------------------------------------------------------------*/
static const struct broadcast_callbacks bc_rx = { broadcast_recv };
static struct broadcast_conn bc;
/*---------------------------------------------------------------------*/
void
rt_callback(struct rtimer *t, void *ptr)
{
 leds_off(LEDS_PERIODIC);
}
/*---------------------------------------------------------------------*/
PROCESS_THREAD(cc2538_demo_process, ev, data)
{
 PROCESS_EXITHANDLER(broadcast_close(&bc))
 PROCESS_BEGIN();
 counter = 0;   //Counter initialization
 count_flag=0;
 broadcast_open(&bc, BROADCAST_CHANNEL, &bc_rx);
 while(1) {
  if(count_flag==0)
{
etimer_set(&et, CLOCK_SECOND);
count_flag=1;
}
   PROCESS_YIELD();
   if(ev == PROCESS_EVENT_TIMER) {
```

```
    leds_on(LEDS_PERIODIC);

        counter++;

//somewhere is here do the serial output

/*************************************************************Time interval for polling can be changed here*****************************************************************************/

            etimer_set(&et, CLOCK_SECOND*.0125);

/*********************************************************Packets are transmitted (broadcasted) using this function*****************************************************************/

              if(counter==4) //put any number . //How many end devices are present in the star network?

                  {

            counter=(counter % 4);  //Which 'operator' should be used here so as to set the value of the counter to 0?. //it has to be same number as above within the counter

            printf("Values received at GW are as follows:\n");

printf("Values reported by Router-L1 are as follows:\n");

printf("1. Remaining Battery = '%d.%d' percentage.\n", batteryrouterintpart, batteryrouterdecpart);

              printf("2. Latitude value = '%d.%d' degree.\n",latitudeintpart ,latitudedecpart);

            printf("3. Longitude value = '%d.%d' degree.\n",longitudeintpart ,longitudedecpart);

            printf("4. rssi          = '%d.%d' dBm.\n",rssiintpart, rssidecpart);

printf("5. Maximum capacity = '%d' nodes.\n", maxcapacity);

printf("6. Current number of connected nodes = '%d' nodes.\n",current_number_of_connected_nodes);

printf("7. Current level = Level '%d'.\n", current_level);

printf("Values reported by Leaf Nodes are as follows:\n");

for (x=1;x<=4;x++)   //x<=the number that is mentioned within the counter

{
```

```
printf("Node %d: %d.%d,%d.%d,%d.%d,%d|", x, battery[x],batterydec[x], latitude[x], latitudedec[x],longitude[x],longitudedec[x], rssi[x]);

}

printf("\n");

            etimer_set(&et, CLOCK_SECOND*.0125);

                  }

      }

 }

 PROCESS_END();

}
```

A.7 a sample code for air-to-Ground communication from transmitters to the UAV in CupCarbon

```
/*--------------------------Transmitter Code-----------------------------------*/
radio radio2       // Defining the WiFi Communication protocol
areadsensor x    // Fetching the data from the random generator
rdata x a b c
print c
//wait 25000
//send c 96
loop
areadsensor x
rdata x a b c
radio radio2
print c
send c 2               //transmit to sensor ID 2
delay 100             //Defining the sampling rate
/*-------------------------------------------------------------------------*/
/*--------------------------Router Code-----------------------------------*/
//Router
loop
wait
read v
print v
delay 1
send v 3        //relay to sensor ID 3
/*-------------------------------------------------------------------------*/
/*--------------------------Receiver Code-----------------------------------*/
//Receiver
```

```
loop
wait
read x
print x
mark x
```

References

[1] M.A. Matin and M. Islam, "Overview of wireless sensor network," *Wireless sensor network*s-*technology and protocols*, vol. 1, no. 3, Sep. 2012, doi:10.4236/wsn.2009.14032.

[2] M.T. Nguyen, C.V. Nguyen, H.T. Do, H.T. Hua, T.A. Tran, A.D. Nguyen, G. Ala and F. Viola, "Uav-assisted data collection in wireless sensor networks: A comprehensive survey," *Electronics*, vol. 10, no. 21, pp. 2603, 2021, doi:10.3390/electronics10212603.

[3] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," *IEEE Access*, vol. 7, pp. 87658–87680, 2019, doi:10.1109/access.2019.292410.

[4] A. Balmori, "Electromagnetic radiation as an emerging driver factor for the decline of insects," *Science of The Total Environment,* vol. 767, pp. 144913, May 2021, doi:10.1016/j.scitotenv.2020.144913.

[5] B. B. Levitt, H. C. Lai, and A. M. Manville, "Effects of non-ionizing electromagnetic fields on flora and fauna, part 1. Rising ambient EMF levels in the environment," *Reviews on Environmental Health*, vol. 37, no. 1, pp. 81–122, May 2021, doi:10.1515/reveh-2021-0026.

[6] Z. Jia, M. Sheng, J. Li, D. Niyato, and Z. Han, "LEO Satellite-Assisted UAV: Joint Trajectory and Data Collection for Internet of Remote Things in 6G Aerial Access Networks," *IEEE Internet of Things*, pp. 1–1, 2020, doi: 10.1109/jiot.2020.3021255

[7] R. Giuliano, F. Mazzenga, and A. Vizzarri, "Satellite-Based Capillary 5G-mMTC Networks for Environmental Applications," *IEEE Aerospace and Electronic Systems Mag.*, vol. 34, no. 10, pp. 40–48, Oct. 2019, doi: 10.1109/maes.2019.2923295.

[8] X. Jiang, M. Sheng, N. Zhao, C. Xing, W. Lu, and X. Wang, "Green UAV communications for 6G: A survey," *Chinese J of Aeronautics*, May 2021, doi:10.1016/j.cja.2021.04.025.

[9] C. Zhan, Y. Zeng, and R. Zhang, "Energy-Efficient Data Collection in UAV Enabled Wireless Sensor Network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, Jun. 2018, doi:10.1109/lwc.2017.2776922.

[10] Beakal Gizachew Assefa and Oznur Ozkasap, "A survey of energy efficiency in SDN: Software-based methods and optimization models," *J of Network and Computer Applications*, vol. 137, pp. 127–143, Jul. 2019, doi:10.1016/j.jnca.2019.04.001.

[11] I. Acharyya, "IoT-Based Sensor Networks: Architectural Organization, Virtualization and Network Re-orchestration," PhD dissertation, Auckland University of Technology, Auckland, New Zealand, 2021.

[12] P. Sweetapple and G. Nugent, "Estimating disease survey intensity and wildlife population size from the density of survey devices: Leg-hold traps and the brushtail possum," *Preventive Veterinary Medicine*, vol. 159, pp. 220–226, Nov. 2018, doi:10.1016/j.prevetmed.2018.09.019.

[13] D.o. Conservation, "Where to put trap and bait lines, " 2020.

[14] D. Forsyth et al., "Calibrating brushtail possum (Trichosurus vulpecula) occupancy and abundance index estimates from leg-hold traps, wax tags and chew cards in the Department of Conservation's Biodiversity and Monitoring Reporting System," *New Zealand J of Ecology*, vol. 42, no. 2, 2018, doi: 10.20417/nzjecol.42.20.

[15] B. S. Faiçal et al., "An adaptive approach for UAV-based pesticide spraying in dynamic environments," *Computers and Electronics in Agriculture*, vol. 138, pp. 210–223, Jun. 2017, doi:10.1016/j.compag.2017.04.011..

[16] J. Martínez-de Dios, A. de San Bernabé, A. Viguria, A. Torres-González, and A. Ollero, "Combining Unmanned Aerial Systems and Sensor Networks for Earth Observation," *Remote Sensing*, vol. 9, no. 4, p. 336, Apr. 2017, doi: 10.3390/rs9040336.

[17] N. Mohamed, Jameela Al-Jaroodi, Imad Jawhar, H. N. Noura, and S. H. Mahmoud, "UAVFog: A UAV-based fog computing for Internet of Things," in *Proc. IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation Conf.* pp. 1-8, 2017.

[18] D. Vasisht, et al. "Farmbeats: An iot platform for data-driven agriculture," in *Proc. Networked Systems Design and Implementation Conf.,* pp. 515-528, Mar 2017.

[19] J. Valente, D. Sanz, A. Barrientos, J. del Cerro, Á. Ribeiro, and C. Rossi, "An Air-Ground Wireless Sensor Network for Crop Monitoring," *Sensors*, vol. 11, no. 6, pp. 6088–6108, Jun. 2011, doi:10.3390/s110606088.

[20] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, and M. Pan, "IoT Enabled UAV: Network Architecture and Routing Algorithm," *IEEE Internet of Things J*, vol. 6, no. 2, pp. 3727–3742, Jan. 2019, doi:10.1109/jiot.2018.2890428.

[21] F.G. Costa, et al. "The use of unmanned aerial vehicles and wireless sensor network in agricultural applications," in *Proc. IEEE Int. Conf. Geoscience and Remote Sensing Symposium*. pp. 5045-5048, 2012.

[22] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Communications Mag.* , vol. 54, no. 5, pp. 36–42, May 2016, doi:10.1109/mcom.2016.7470933.

[23] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput Maximization for UAV-Enabled Mobile Relaying Systems," *IEEE Trans. on Communications*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016, doi:10.1109/tcomm.2016.2611512.

[24] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned Aerial Vehicle With Underlaid Device-to-Device Communications: Performance and Tradeoffs," *IEEE Trans. on Wireless Communications*, vol. 15, no. 6, pp. 3949–3963, Jun. 2016, doi:10.1109/twc.2016.2531652.

[25] Y. Zeng, X. Xu, and R. Zhang, "Trajectory Design for Completion Time Minimization in UAV-Enabled Multicasting," I*EEE Trans. on Wireless Communications*, vol. 17, no. 4, pp. 2233–2246, Apr. 2018, doi:10.1109/twc.2018.2790401.

[26] K. F. Hayajneh, K. Bani-Hani, H. Shakhatreh, M. Anan, and A. Sawalmeh, "3D Deployment of Unmanned Aerial Vehicle-Base Station Assisting Ground-Base Station," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–11, Aug. 2021, doi:10.1155/2021/2937224.

[27] Q. Wu, Y. Zeng, and R. Zhang, "Joint Trajectory and Communication Design for Multi-UAV Enabled Wireless Networks," *IEEE Trans. on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018, doi: 10.1109/twc.2017.2789293.

[28] C. Wang, F. Ma, J. Yan, D. De, and S. K. Das, "Efficient Aerial Data Collection with UAV in Large-Scale Wireless Sensor Networks," *Int. J of Distributed Sensor Networks,* vol. 11, no. 11, p. 286080, Jan. 2015, doi:10.1155/2015/286080

[29] W. Yue and Z. Jiang, “Path Planning for UAV to Collect Sensors Data Based on Spiral Decomposition,” *Procedia Computer Science*, vol. 131, pp. 873–879, 2018, doi:10.1016/j.procs.2018.04.291.

[30] C. Zhan, Y. Zeng, and R. Zhang, “Trajectory Design for Distributed Estimation in UAV-Enabled Wireless Sensor Network,” *IEEE Trans. on Vehicular Technology*, vol. 67, no. 10, pp. 10155–10159, Oct. 2018, doi: 10.1109/tvt.2018.2859450.

[31] S. Rashed and M. Soyturk, “Analyzing the Effects of UAV Mobility Patterns on Data Collection in Wireless Sensor Networks,” *Sensors*, vol. 17, no. 2, p. 413, Feb. 2017, doi:10.3390/s17020413.

[32] Y. Qu, Q. Pan, and J. Yan, “Flight path planning of UAV based on heuristically search and genetic algorithms,” in *Proc. 31st Annu. Conf. of IEEE Industrial Electronics Society,* pp. 45-50, 2005, doi:10.1109/iecon.2005.1568876.

[33] C. Zhang, Z. Zhen, D. Wang, and L. Liu, “UAV path planning method based on ant colony optimization,” in *Proc. Chinese Control and Decision Conf*. pp. 3790-3792, 2010.

[34] J. Zhang, Y. Zeng, and R. Zhang, “UAV-Enabled Radio Access Network: Multi-Mode Communication and Trajectory Design,” *IEEE Trans. on Signal Processing*, vol. 66, no. 20, pp. 5269–5284, Oct. 2018, doi: 10.1109/tsp.2018.2866384.

[35] J. Li et al., “Joint Optimization on Trajectory, Altitude, Velocity, and Link Scheduling for Minimum Mission Time in UAV-Aided Data Collection,” *IEEE Internet of Things*, vol. 7, no. 2, pp. 1464–1475, Feb. 2020, doi: 10.1109/jiot.2019.2955732

[36] Goudarzi, N. Kama, M. H. Anisi, S. Zeadally, and S. Mumtaz, “Data collection using unmanned aerial vehicles for Internet of Things platforms,” *Computers & Electrical Engineering*, vol. 75, pp. 1–15, May 2019, doi: 10.1016/j.compeleceng.2019.01.028.

[37] D. Popescu, F. Stoican, G. Stamatescu, L. Ichim, and C. Dragana, “Advanced UAV–WSN System for Intelligent Monitoring in Precision Agriculture,” *Sensors*, vol. 20, no. 3, p. 817, Feb. 2020, doi:10.3390/s20030817.

[38] F. Chao et al., “Predictive Trajectory-Based Mobile Data Gathering Scheme for Wireless Sensor Networks,” *Complexity*, vol. 2021, pp. 1–17, Jan. 2021, doi: 10.1155/2021/3941074.

[39] J. Faigl and P. Vana, “Surveillance Planning With Bézier Curves,” *IEEE Robotics and Automation Letters,* vol. 3, no. 2, pp. 750–757, Apr. 2018, doi: 10.1109/lra.2018.2789844

[40] J. Faigl, P. Vana, and R. Penicka, “Multi-vehicle close enough orienteering problem with Bézier curves for multi-rotor aerial vehicles,” in *Proc. Int. Conf. on Robotics and Automation (ICRA)*. pp. 3039-3044, May 2019.

[41] Leopoldo Puente Rodriguez, J. A. Cobano, and A. Ollero. “Smooth trajectory generation for wind field exploitation with a small UAS,” in *Proc. 2017 Int. Conf. on Unmanned Aircraft Systems (ICUAS)*. pp. 1241-1249, Jun 2017.

[42] Y. Zeng and R. Zhang, “Energy-Efficient UAV Communication With Trajectory Optimization,” *IEEE Trans. on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017, doi:10.1109/twc.2017.2688328.

[43] Y. Chen, S. Zhang, S. Xu, and G. Li, “Fundamental trade-offs on green wireless networks,” *IEEE Communication Mag.*, vol. 49, no. 6, pp. 30–37, Jun. 2011, doi: 10.1109/mcom.2011.5783982.

[44] Mahdi Ben Ghorbel, D. Rodriguez-Duarte, Hakim Ghazzai, Md. Jahangir Hossain, and Hamid Menouar, “Joint Position and Travel Path Optimization for

Energy Efficient Wireless Data Gathering Using Unmanned Aerial Vehicles," *IEEE Trans. on Vehicular Technology*, vol. 68, no. 3, pp. 2165–2175, Jan. 2019, doi:10.1109/tvt.2019.2893374.

[45] Y. Xu, L. Xiao, D. Yang, L. Cuthbert, and Y. Wang, "Energy-Efficient UAV Communication with Multiple GTs Based on Trajectory Optimization," *Mobile Information Systems*, vol. 2018, pp. 1–10, 2018, doi:10.1155/2018/5629573

[46] E. I. Grøtli and T. A. Johansen, "Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP," *J of Intelligent & Robotic Systems*, vol. 65, no. 1–4, pp. 265–282, Aug. 2011, doi:10.1007/s10846-011-9619-8.

[47] Di Franco, C. and G. Buttazzo, "Energy-aware coverage path planning of UAVs," in *Proc. IEEE Int. Conf. on autonomous robot systems and competitions*. pp. 111-117, 2015.

[48] D. Ebrahimi, S. Sharafeddine, P.-H. Ho, and C. Assi, "UAV-Aided Projection-Based Compressive Data Gathering in Wireless Sensor Networks," *IEEE Internet of Things*, vol. 6, no. 2, pp. 1893–1905, Apr. 2019, doi: 10.1109/jiot.2018.287884.

[49] J. Gong, T.-H. Chang, C. Shen, and X. Chen, "Flight Time Minimization of UAV for Data Collection Over Wireless Sensor Networks," *IEEE J on Selected Areas in Communications*, vol. 36, no. 9, pp. 1942–1954, Sep. 2018, doi: 10.1109/JSAC.2018.2864420.

[50] C. You and R. Zhang, "3D Trajectory Optimization in Rician Fading for UAV-Enabled Data Harvesting," *IEEE Trans. on Wireless Communications*, vol. 18, no. 6, pp. 3192–3207, Jun. 2019, doi:10.1109/twc.2019.2911939.

[51] C. You and R. Zhang, "Hybrid Offline-Online Design for UAV-Enabled Data Harvesting in Probabilistic LoS Channels," *IEEE Trans. on Wireless Communicarions,* vol. 19, no. 6, pp. 3753–3768, Jun. 2020, doi: 10.1109/twc.2020.2978073.

[52] Pejman Abdollahzadeh Karegar and Adnan Al-Anbuky, "Travel Path Planning for UAV as a Data Collector for a Sparse WSN," in *Proc. 17th Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*. pp. 359-366, 2021.

[53] Pejman Abdollahzadeh Karegar and Adnan Al-Anbuky, "UAV-assisted data gathering from a sparse wireless sensor adaptive networks," *Wireless Networks*, vol. 29, no. 3, pp. 1367–1384, Dec. 2022, doi:10.1007/s11276-022-03194-4.

[54] Pejman Abdollahzadeh Karegar and Adnan Al-Anbuky, "UAV as a Data Ferry for a Sparse Adaptive WSN," in *Proc. 27th Asia Pacific Conf. on Communications (APCC)*. pp. 284-289, 2022.

[55] J. Valente, D. Sanz, A. Barrientos, J. del Cerro, Á. Ribeiro, and C. Rossi, "An Air-Ground Wireless Sensor Network for Crop Monitoring," *Sensors*, vol. 11, no. 6, pp. 6088–6108, Jun. 2011, doi:10.3390/s110606088.

[56] T.F. Khan, and D.S. Kumar. "Mobile collector aided energy reduced (MCER) data collection in agricultural wireless sensor networks," in *Proc. IEEE 6th Int. Conf. on Advanced Computing (IACC)*. pp. 629-633, 2016.

[57] B. S. Faiçal et al., "The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides," *J of Systems Architecture*, vol. 60, no. 4, pp. 393–404, Apr. 2014, doi:10.1016/j.sysarc.2014.01.004.

[58] H. Jawad, R. Nordin, S. Gharghan, A. Jawad, and M. Ismail, "Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review," *Sensors,* vol. 17, no. 8, p. 1781, Aug. 2017, doi:10.3390/s17081781

[59] P. Mathur, R. H. Nielsen, N. R. Prasad, and R. Prasad, "Data collection using miniature aerial vehicles in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 6, no. 1, pp. 17–25, Feb. 2016, doi:10.1049/iet-wss.2014.0120.

[60] H. Cao, Y. Liu, X. Yue, and W. Zhu, "Cloud-Assisted UAV Data Collection for Multiple Emerging Events in Distributed WSNs," *Sensors*, vol. 17, no. 8, p. 1818, Aug. 2017, doi:10.3390/s17081818.

[61] Tifenn Rault, Abdelmadjid Bouabdallah, and Yacine Challal, "WSN lifetime optimization through controlled sink mobility and packet buffering," In *Global Information Infrutructure Symp. GIIS* Oct. pp. 1-6, 2013.

[62] Cristian Dragana, Grigore Stamatescu, V. Mihai, and D. Popescu, "An approach for weighted average consensus in event detection," in *Proc. 25th Mediterranean Conf. on Control and Automation (MED)*.- pp. 1100-1105, 2017.

[63] D. Vasisht et al. "{FarmBeats}: An {IoT} Platform for {Data-Driven} Agriculture," in *Proc. 14th USENIX Symp. on Networked Systems Design and Implementation (NSDI 17)*. pp. 515-529, 2017.

[64] G. Raja, V. S. Saran, Sudha Anbalagan, Ali Kashif Bashir, M. Imran, and N. Nasser, "Collisionless fast pattern formation mechanism for dynamic number of uavs," in *Proc. GLOBECOM IEEE Global Communications Conf*. pp. 1-6, 2020.

[65] G.P. Kumar, and B. Sridevi, "Simulation of Efficient Cooperative UAVs using Modified PSO Algorithm," *WSEAS Trans. on Information Science and Applications*, pp. 94-99, 2019.

[66] M. Alagirisamy and C.-O. Chow, "An energy based cluster head selection unequal clustering algorithm with dual sink (ECH-DUAL) for continuous monitoring applications in wireless sensor networks," *Cluster Computing*, vol. 21, no. 1, pp. 91–103, Jul. 2017, doi:10.1007/s10586-017-0943-z.

[67] K. Kalaivanan and V. Bhanumathi, "Reliable location aware and Cluster-Tap Root based data collection protocol for large scale wireless sensor networks," *J of Network and Computer Applications*, vol. 118, pp. 83–101, Sep. 2018, doi: 10.1016/j.jnca.2018.06.005.

[68] R. Velmani and B. Kaarthick, "An Efficient Cluster-Tree Based Data Collection Scheme for Large Mobile Wireless Sensor Networks," *IEEE Sensors*, vol. 15, no. 4, pp. 2377–2390, Apr. 2015, doi:10.1109/jsen.2014.2377200.

[69] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Ring Routing: An Energy-Efficient Routing Protocol for Wireless Sensor Networks with a Mobile Sink," *IEEE Trans. on Mobile Computing*, vol. 14, no. 9, pp. 1947–1960, Sep. 2015, doi:10.1109/tmc.2014.2366776.

[70] Tazibt, C. Y., Bekhti, M., Djamah, T., Achir, N., & Boussetta, K. "Wireless sensor network clustering for UAV-based data gathering," in *Proc. Wireless Days*. pp. 245- 247, 2017.

[71] R. Tarighi, K. Farajzadeh, and H. Hematkhah, "Prolong network lifetime and improve efficiency in WSN-UAV systems using new clustering parameters and CSMA modification," *International J of Communication Systems*, vol. 33, no. 7, p. 4324, Jan. 2020, doi: 10.1002/dac.4324.

[72] Kanungo, T., et al., "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. on pattern analysis and machine intelligence*, pp. 881-892, 2002.

[73] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," *J of Network and Computer Applications*, vol. 97, pp. 23–34, Nov. 2017, doi:10.1016/j.jnca.2017.08.006.

[74] Cristian Dragana, Grigore Stamatescu, L. Ichim, and D. Popescu, "Interlinking unmanned aerial vehicles with wireless sensor networks for improved large area monitoring," in *Proc. 4th International Conf. on Control, Decision and Information Technologies (CoDIT),* Apr. 2017, pp. 359-364, doi:10.1109/c02618.

[75] I. Khan, Fatna Belqasmi, Roch Glitho, N. Crespi, M. Morrow, and Paul Anthony Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 553–576, Jan. 2016, doi:10.1109/comst.2015.2412971.

[76] D.Z. Al-Hamid, and A. Al-Anbuky. "Vehicular Grouping Protocol: Towards Cyber Physical Network Intelligence," in *Proc. 2021 IEEE Int. Conf. on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics).* pp. 553-576, 2021.

[77] I.S. Acharyya, A. Al-Anbuky, and S. Sivaramakrishnan. "Software-defined sensor networks: towards flexible architecture supported by virtualization," in *Proc. Global IoT Sum. (GIoTS)*. pp. 1-4, 2019.

[78] D.Z. Al-Hamid and A. Al-Anbuky. "Vehicular grouping and network formation: virtualization of network self-healing," in *Proc. Int. Conf. on Internet of Vehicles, Paris, France*. pp. 106-121, Nov 2018.

[79] D.Z. Al-Hamid and A. Al-Anbuky. "Vehicular Network Dynamic Grouping Scheme," in *2021 IEEE Int. Conf. on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. pp. 316-318, 2021.

[80] D.Z. Al-Hamid and A. Al-Anbuky. "Vehicular Intelligence: Towards Vehicular Network Digital-Twin," in *Proc. 27th Asia Pacific Conf. on Communications (APCC)*. pp. 427-432, 2022.

[81] D. Z. Al-Hamid and A. Al-Anbuky, "Vehicular Networks Dynamic Grouping and Re-Orchestration Scenarios," *Information,* vol. 14, no. 1, p. 32, Jan. 2023, doi: 10.3390/info14010032.

[82] I.S. Acharyya and A. Al-Anbuky. "Towards wireless sensor network softwarization," in *Proc. IEEE NetSoft Conf. and Workshops (NetSoft)*. pp. 378-383, 2016.

[83] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghrayeb, "Age of Information Aware Trajectory Planning of UAVs in Intelligent Transportation Systems: A Deep Learning Approach," *IEEE Trans. on Vehicular Technology*, vol. 69, no. 11, pp. 12382–12395, Nov. 2020, doi:10.1109/tvt.2020.3023861.

[84] K. Karunanithy and B. Velusamy, "Energy efficient cluster and travelling salesman problem based data collection using WSNs for Intelligent water irrigation and fertigation," *Measurement,* vol. 161, p. 107835, Sep. 2020, doi: 10.1016/j.measurement.2020.107835.

[85] S. Park, S. Yun, H. Kim, R. Kwon, J. Ganser, and S. Anthony, "Forestry monitoring system using lora and drone," in *Proc. of the 8th Int. Conf. on Web Intelligence, Mining and Semantics*. pp. 1-8, Jun 2018.

[86] A. Rao, H. Shao, and X. Yang. "The design and implementation of smart agricultural management platform based on UAV and wireless sensor network," in *Proc. IEEE 2nd Int. Conf. on Electronics Technology (ICET)*. pp. 248-252, 2019.

[87] S. Say, H. Inata, J. Liu, and S. Shimamoto, "Priority-Based Data Gathering Framework in UAV-Assisted Wireless Sensor Networks," *IEEE Sensors J*, vol. 16, no. 14, pp. 5785–5794, Jul. 2016, doi:10.1109/jsen.2016.2568260.

[88] Say Sotheara, K. Aso, Naoto Aomi, and S. Shimamoto, "Effective data gathering and energy efficient communication protocol in wireless sensor networks employing UAV," in *Proc. IEEE wireless communications and networking Conf. (WCNC)*. pp. 2342-2347, 2014.

[89] D.-T Ho and S. Shimamoto. “Highly reliable communication protocol for WSN-UAV system employing TDMA and PFS scheme,” in *Proc. IEEE Globecom Workshops (Gc Wkshps)*. pp.1320-1324, 2011.

[90] Khan, A.R., S.M. Bilal, and M. Othman. “A performance comparison of open source network simulators for wireless networks,” in *Proc. IEEE Int. Conf. on control system, computing and engineering*. pp. 34-38, 2012.

[91] Thiyagarajan, V. “Energy Efficient Wireless Data Gathering using Unmanned Aerial Vehicle by adopting Crow Search Optimization Algorithm,” *J of Soft Computing and Engineering Applications*, vol. 2, no. 1, 2021.

[92] Alexandru-Valentin Vladuta, Constantin Grumazescu, I. Bica, and Victor-Valeriu Patriciu, “Energy considerations for data gathering protocol in wireless sensor networks using unmanned aerial vehicles,” in *Proc. Int. Conf. on Communications (COMM)*. pp. 237-240, 2016.

[93] A.-V. . Vladuta, M. L. Pura, and I. Bica, “MAC Protocol for Data Gathering in Wireless Sensor Networks with the Aid of Unmanned Aerial Vehicles,” *Advances in Electrical and Computer Engineering*, vol. 16, no. 2, pp. 51–56, 2016, doi: 10.4316/aece.2016.02007.

[94] I. Cardei, M. Cardei, and R. Papa. “UAV-enabled data gathering in wireless sensor networks,” in *Proc. IEEE 37th Int. Performance Computing and Communications Conf. (IPCCC)*. pp. 1-8, 2018.

[95] M. Garraffa, Mustapha Bekhti, L. Létocart, Nadjib Achir, and Khaled Boussetta, “Drones path planning for WSN data gathering: A column generation heuristic approach,” in *Proc. IEEE Wireless Communications and Networking Conf. (WCNC)*. pp. 1-6, 2018.

[96] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, and Y. Ren, “Stability of Cloud-Based UAV Systems Supporting Big Data Acquisition and Processing,” I*EEE Trans. on Cloud Computing*, vol. 7, no. 3, pp. 866–877, Jul. 2019, doi: 10.1109/tcc.2017.2696529.

[97] A. Bounceur, et al. “Cupcarbon-lab: An iot emulator,” in *Proc. 15th IEEE annual consumer communications & networking Conf. (CCNC)*. 2018.

[98] A. Dunkels, B. Gronvall, and T. Voigt. “Contiki-a lightweight and flexible operating system for tiny networked sensors,” in *Proc. 29th annual IEEE Int. Conf. on local computer networks*. pp. 455-462, 2004.

[99] A. Barbato, M. Barrano, A. Capone, and Nicolo Figiani, “Resource oriented and energy efficient routing protocol for IPv6 wireless sensor networks,” in *Proc. IEEE Online Conf. on Green Communications (OnlineGreenComm)*. 2013.

[100] https://ardupilot.org/planner/docs/mission-planner-overview.html, *ArduPilot.* 2023.

[101] https://docs.qgroundcontrol.com/master/en/getting_started/quick_start.html, *QGroundControl.* 2023.

[102] https://ardupilot.org/plane/docs/quadplane-support.html, *QuadPlane.* 2023.

[103] P. Bézier, “Matematical and Practical Possibilities of Unisurf,” *Computer Aided Geometric Design*, pp. 127–152, Jan. 1974, doi:10.1016/b978-0-12-079050-0.50012-6.

[104] M. Amirinasab Nasab, S. Shamshirband, A. Chronopoulos, A. Mosavi, and N. Nabipour, “Energy-Efficient Method for Wireless Sensor Networks Low-Power Radio Operation in Internet of Things,” *Electronics*, vol. 9, no. 2, p. 320, Feb. 2020, doi:10.3390/electronics9020320

[105] https://www.ti.com/lit/ds/symlink/cc2538.pdf?ts=1663294495656&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC2538%253FkeyMatc

h%253DCC2538%2526tisearch%253Dsearch-everything%2526usecase%253DGPN, *CC2538em datasheet available at:*. 2023.

[106] J. Ren, Y. Zhang, K. Zhang, A. Liu, J. Chen, and X. Shen, "Lifetime and Energy Hole Evolution Analysis in Data-Gathering Wireless Sensor Networks," *IEEE Trans. on Industrial Informatics*, vol. 12, no. 2, pp. 788–800, Apr. 2016, doi:10.1109/tii.2015.2411231.

[107] E. Yanmaz, S. Hayat, J. Scherer, & C. Bettstetter. "Experimental performance analysis of two-hop aerial 802.11 networks," *In Proc. IEEE Wireless Communications and Networking Conf. (WCNC)*, pp. 3118-3123, Apr 2014.

[108] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV Trajectory Planning for Data Collection from Time-Constrained IoT Devices," *IEEE Trans. on Wireless Communications,* vol. 19, no. 1, pp. 34–46, Jan. 2020, doi:10.1109/twc.2019.2940447