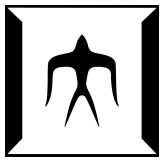
論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	A Study on Gateway Mobility Control for Heterogeneous Ad Hoc Networks
著者(和文)	宮太地
Author(English)	Taichi Miya
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第12391号, 授与年月日:2023年3月26日, 学位の種別:課程博士, 審査員:山岡 克式,植松 友彦,府川 和彦,北口 善明,西尾 理志
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12391号, Conferred date:2023/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis



Tokyo Institute of Technology

A Study on Gateway Mobility Control for Heterogeneous Ad Hoc Networks

---

by

Taichi MIYA

Supervisor:

Prof. Katsunori YAMAOKA

A Dissertation Presented to

Department of Information and Communications Engineering
School of Engineering

In Partial Fulfilment of
the Requirements for the Degree of

Doctor of Engineering

February 2023

Preface

In today's world, where everything is connected to the Internet, ad hoc networks are an indispensable technology for society – autonomous, distributed, and cooperative infrastructure-less communication networks. As the quality requirements of networks, such as bandwidth and latency, have become more refined and wireless communication protocols have diversified, heterogeneous ad hoc networks, in which multiple local ad hoc networks are interconnected by protocol translators called gateways, have attracted the attention of many researchers and engineers.

This paper is titled "A Study on Gateway Mobility Control for Heterogeneous Ad Hoc Networks" and comprises seven chapters. Chapter 1, "Introduction," introduces the development of ad hoc network-related technologies and the current social demands and describes their potential applications, especially in the field of swarm robots. Chapter 2, "Related Works," systematically summarizes the technical issues by surveying a wide range of existing studies on ***1)*** interoperability of ad hoc networks, ***2)*** shortest path searching algorithms on the graph theory, ***3)*** routing protocols for ad hoc networks, ***4)*** wireless communication network designs on swarm robotics to realize real-time video transmission, and ***5)*** formation control algorithm for swarm management. Chapter 3, "Application Modeling and Problem Statement," introduces examples of heterogeneous ad hoc networks composed of drone swarms and abstracts a drone-based video transmission network, the target of this study, into a general graph model. In Chapter 4, "Experimental Analysis of Communication Relaying Latency in Low-Energy Ad Hoc Networks," measurement experiments are planned and conducted in an ad hoc network

constructed using actual embedded devices. The numerical data obtained by the experiments clarify that an application processing delay generated by a low-power embedded CPU can be greater than the delay caused by the wireless link and accounts for a large proportion of the end-to-end communication delay. This fact indicates that in ad hoc networks, a simple reduction in the number of end-to-end path hops can effectively reduce end-to-end delay. Besides, based on the increasing trend of processing delay in relation to the amount of relay traffic, a mathematical model describing the end-to-end delay in a low-energy ad hoc network is proposed. Chapter 5, "Link Stabilizer and Path Optimizer," proposes two types of gateway mobility control algorithms: *Link Stabilizer* and *Path Optimizer*. The former is a microscopic mobility control algorithm that aims to maintain stable wireless links between gateways and neighboring nodes. The latter is a macroscopic mobility control algorithm involving the gateway's long-distance movement by dynamically reconfiguring the network topology to construct end-to-end shortest paths. This method is characterized by connecting the ends with the shortest physical path between them by relocating the gateways for each communication request. It can significantly reduce communication distance compared to the existing method that searches for the logical shortest path. Computer simulation results show that these algorithms improve gateway availability and connection stability and reduce the average end-to-end delay of the network. Chapter 6, "Path Coordinator," proposes *Path Coordinator* as a mobility control in high-traffic environments. While the algorithm in Chapter 5 is an independent mobility control for each gateway, it is unique in that the path coordinator operates all gateways in the network cooperatively and relocates them simultaneously. It allows for discovering a globally optimal gateway placement rather than a local optimum. Computer simulations assuming actual drone specs show that the algorithm smooths communication quality and improves the allowable delay satisfaction rate even under heavy loads. A conceptual communication protocol is also designed to implement these three algorithms in an autonomous decentralized environment. Chapter 7, "Conclusion and Future Works," summarizes the efforts of this study and outlines issues that need to be considered when expanding the scope to more

general situations in the future.

In summary, this paper shows that the three proposed gateway mobility control algorithms can be applied to a heterogeneous ad hoc network consisting mainly of drones to improve gateway availability and reduce end-to-end communication delay under a wide variety of traffic conditions and node mobility patterns. This study contributes to the sustainable development of technologies around ad hoc networks and thus provides a basis for new technologies in the future society.

Acknowledgement

Writing this dissertation, I have received a great deal of assistance, support, and guidance from people around me. I would like to express the deepest appreciation to my supervisor Prof. Katsunori Yamaoka. Moreover, I am deeply grateful to Prof. Kohta Ohshima and Prof. Yoshiaki Kitaguchi. Special thanks to Mrs. Matsuzaki, and all the members of Yamaoka-Kitaguchi Laboratory for all the fun, help and support. Finally, I am indebted to my family members for their full and unconditional support for all these years.

Contents

Preface . . . . . **i**

Acknowledgement . . . . . **iv**

Table of Contents . . . . . **v**

List of Figures . . . . . **ix**

List of Tables . . . . . **xiv**

Chapter 1: Introduction **1**

1.1 Social Background 1

1.1.1 Ad Hoc Network 1

1.1.2 Rise of HANETs 2

1.1.3 Future Potential of HANETs 4

1.2 Major Issues 4

1.2.1 Latency 4

1.2.2 Gateway Availability 5

1.3 Objectives and Motivation 6

1.4 Overview of Dissertation 7

Chapter 2: Related Work **8**

2.1 Interoperability of Ad Hoc Networks and HANETs 8

2.2 Drone Swarm 9

2.2.1 Rise of Protocol Diversity 9

2.2.2 Multi-hop Communication 12
2.2.3 Realtime Video Transmission and QoS 14
2.2.4 Formation Control 15
2.3 Generic Gateway Placement Optimization 16
2.4 Summary 17

Chapter 3: Application Modeling and Problem Statement 18
3.1 Protocol Stack 18
3.2 Application Scenario and Objective 19
3.3 Simplification 21

Chapter 4: Experimental Analysis of Communication Relaying Delay in Low-Energy Ad Hoc Networks 23
4.1 Introduction and Related Work 23
4.2 Network Stack of Linux Kernel 24
4.2.1 Generic Packet Receiving Sequence 25
4.2.2 Generic Packet Sending Sequence 26
4.3 Design of Experiments 27
4.3.1 Definitions of Delays in Network 27
4.3.2 Proxy: How to Relay Packets 28
4.3.3 Measurement Conditions 28
4.4 Measurement Methods 29
4.4.1 Building OLSR Ad Hoc Network 29
4.4.2 Preparation of Traffic Generator 30
4.4.3 Implementation of the Kernel Module for Measurement 31
4.4.4 Implementation of the Proxy 31
4.4.5 Additional CPU Load at Proxy 32
4.5 Results and Discussion 32
4.5.1 Time Variation of Received Signal Strength Indicator 33
4.5.2 Time Variation of Node Delay and Coffee-break Effect 33

4.5.3 Jitter and Packet Loss Rate 36
4.5.4 Node Delay 38
4.5.5 Link Delay vs. Node Delay 42
4.6 Conclusion 44

Chapter 5: Link Stabilizer and Path Optimizer 46
5.1 Introduction 46
5.2 Autonomous Gateway Mobility Control 48
5.2.1 Algorithm Overview 48
5.2.2 Link Stabilizer 49
5.2.3 Path Optimizer 56
5.3 Distributed Gateway Selection Mechanism 61
5.3.1 Protocol Overview 62
5.3.2 Extended Hybrid Wireless Mesh Protocol 62
5.4 Preparation for Performance Evaluation 66
5.4.1 Mobility Models 66
5.4.2 Random Geometric Graph 72
5.4.3 Probabilistic Movement using Graph Centralities 74
5.5 Performance Evaluation and Discussion 75
5.5.1 Gateway Availability Improvement by Link Stabilizer 76
5.5.2 Delay Reduction by Path Optimizer 85
5.5.3 Optimizer vs. Optimizer* 90
5.5.4 Optimizer vs. Brute-force 91
5.6 Conclusion 95

Chapter 6: Path Coordinator 96
6.1 Introduction 96
6.2 Proposed Algorithms 99
6.2.1 Algorithm Overview 99
6.2.2 Gateway Relocation Algorithm 100

6.2.3 Flow Rerouting Algorithm 106
6.2.4 Computational Complexity 109
6.2.5 The Behavior of Algorithms 110
6.3 Cooperative Behavior of Distributed Gateways 111
6.3.1 Protocol Overview 111
6.3.2 Definitions of New Message Types 115
6.3.3 Expansion of the Protocol Sequence 116
6.4 Preparation for Performance Evaluation 119
6.4.1 Topology Models 119
6.4.2 Gateway Preplacement using Graph Centrality Measures 122
6.5 Performance Evaluation and Discussion 124
6.5.1 Relocation and Rerouting 125
6.5.2 Strategic Preplacement vs. Adaptive Relocation 138
6.5.3 Proposed vs. Brute-force 141
6.5.4 Timeline Analysis 145
6.6 Conclusion 149

Chapter 7: Conclusion and Future Works 151
7.1 Conclusion 151
7.2 Future Works and Suggestions 152
7.2.1 Building Enhanced Transmission Delay Model 153
7.2.2 Addressing Dynamic Link Quality 154
7.2.3 Radio Interference Consideration 154
7.2.4 Throughput-based QoS Evaluation 154

Bibliography 156

Achievements 168

List of Figures

Figure 1.1: Concept of HANETs (Heterogeneous Ad Hoc Networks). 3

Figure 3.1: Protocol stack of the target application: two clusters are interconnected by one gateway cluster. 19

Figure 3.2: Concept of heterogeneous drone swarms. 20

Figure 4.1: Packet queueing in the Linux kernel standard network stack. . . . 25

Figure 4.2: Three Raspberry Pis connected logically inline. 30

Figure 4.3: Format of UDP probe generated by iperf. 31

Figure 4.4: **Timelines:** Average time variation of RSSI and 100th percentile node delays during the measurement. 34

Figure 4.4: **Timelines:** Average time variation of RSSI and 100th percentile node delays during the measurement. 35

Figure 4.5: **Jitter and packet loss rates:** 100th percentile values under several conditions. 36

Figure 4.5: **Jitter and packet loss rates:** 100th percentile values under several conditions. 37

Figure 4.6: **Node and *processing delays*:** 95th percentile delays under the several conditions. 38

Figure 4.6: **Node and *processing delays*:** 95th percentile delays under the several conditions. 39

Figure 4.6: **Node and *processing delays*:** 95th percentile delays under the several conditions. 40

Figure 4.7: **ECDF:** Several percentile node delays under 1000 bytes, 200 pps and 0% *stress*. 40
Figure 4.7: **ECDF:** Several percentile node delays under 1000 bytes, 200 pps and 0% *stress*. 41
Figure 4.8: **End-to-end delay:** 95th percentile end-to-end delays including node and link delays. 43
Figure 4.8: **End-to-end delay:** 95th percentile end-to-end delays including node and link delays. 44

Figure 5.1: A state diagram of gateway function 49
Figure 5.2: **Micro-connectivity control:** A gateway surrounded by two clusters, cluster red and blue, determines the links to maintain and discard. 51
Figure 5.2: **Micro-connectivity control:** A gateway surrounded by two clusters, cluster red and blue, determines the links to maintain and discard. 52
Figure 5.3: The logical neighbors exert a force on gateway. 53
Figure 5.4: Types of forces acting between gateway and other nodes: G means gateway and N means non-gateway. 54
Figure 5.5: An actual behavior of gateway movement under the control of *Link Stabilizer*: cluster red and blue are interconnected by the green gateways. 57
Figure 5.6: An actual behavior of gateway movement under the control of *Path Optimizer*: one of the gateways has been selected and relocated for shortcut formation. 61
Figure 5.7: Data transmission sequence in extended HWMP. 64
Figure 5.8: Three stages of node movement in the random waypoint model. 67
Figure 5.9: Three stages of node movement in the SRCM [93]. 69
Figure 5.10: Image of node movement in the hovering model: the white arrow represents an external force, such as wind. 71

Figure 5.11: Sample topology of a random geometric graph [95]. 73
Figure 5.12: Standard sigmoid function ($a = 1, x_0 = 0$) 75
Figure 5.13: **ECDF:** Performance of the *Link Stabilizer* under the SRCM with 5–10 m/s of node speed and 1–10 meters of mobility radius. 78
Figure 5.13: **ECDF:** Performance of the *Link Stabilizer* under the SRCM with 5–10 m/s of node speed and 1–10 meters of mobility radius. 79
Figure 5.14: **ECDF:** Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability. 79
Figure 5.14: **ECDF:** Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability. 80
Figure 5.15: **ECDF:** Performance of *Link Stabilizer* under the random waypoint model with 5–10 m/s of node speed and 1–5 seconds of pausing time. 82
Figure 5.15: **ECDF:** Performance of *Link Stabilizer* under the random waypoint model with 5–10 m/s of node speed and 1–5 seconds of pausing time. 83
Figure 5.16: **ECDF:** Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability. 83
Figure 5.16: **ECDF:** Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability. 84
Figure 5.17: **ECDF:** Number of reduced hops by *Path Optimizer* under several conditions. 86
Figure 5.18: **ECDF:** Gateway traveling distance for every shortcut formation under several conditions. 87
Figure 5.19: **Cost and availability:** Total number of gateway relocations and average number of idle gateways. 88
Figure 5.20: **Satisfaction rate and mobility cost:** Performance of the Optimzier* compared with the Stationary and *Path Optimizer*. . 92

Figure 5.20: **Satisfaction rate and mobility cost:** Performance of the Optimzier* compared with the Stationary and *Path Optimizer*. . 93
Figure 5.21: **Performance upper limit:** Performance of *Path Optimizer* compared with the Stationary and Brute-force. 94

Figure 6.1: Target situation of this paper: dynamically relocate gateways according to the situation to keep the locations optimal for the current flows. 98
Figure 6.2: Gateway as a 3-states machine. 100
Figure 6.3: A 3-hop flow and its hop budget area surrounded by an ellipse curve. 102
Figure 6.4: Flow rerouting strategies: first minimize the path hops, then aggregate flows among gateways. 106
Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur. 112
Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur. 113
Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur. 114
Figure 6.6: Control and data transmission sequence in the proposed protocol. 117
Figure 6.7: Example of Manhattan Grid Topology. 121
Figure 6.8: Flow generation pattern for test cases. 126
Figure 6.9: **Relocation and rerouting:** Comparison of the total amount of reduced path hops under various conditions. 128
Figure 6.10: **ECDF:** Amount of reduced path hops by *Path Coordinator* under Connected RGG network. 130

Figure 6.10: **ECDF:** Amount of reduced path hops by *Path Coordinator* under Connected RGG network. 131
Figure 6.11: **ECDF:** Amount of reduced path hops by *Path Coordinator* under Manhattan Grid Topology network. 131
Figure 6.11: **ECDF:** Amount of reduced path hops by *Path Coordinator* under Manhattan Grid Topology network. 132
Figure 6.12: **ECDF:** Amount of gateway movement occurred by *Path Coordinator* under Connected RGG network. 133
Figure 6.13: **ECDF:** Amount of gateway movement occurred by *Path Coordinator* under Manhattan Grid Topology network. 134
Figure 6.14: **Allowable hop satisfaction rate:** Improvement of the satisfaction rate under several conditions. 136
Figure 6.15: **Idle gateways:** Improvement of the number of idling gateways under several conditions. 137
Figure 6.16: **Preplacement performance:** Reduced hops when *Path Coordinator* is applied in the preplaced network under various conditions. 140
Figure 6.17: **Performance upper limit:** Limit of the path hop reduction and the comparison with *Path Coordinator*. 143
Figure 6.17: **Performance upper limit:** Limit of the path hop reduction and the comparison with *Path Coordinator*. 144
Figure 6.17: **Performance upper limit:** Limit of the path hop reduction and the comparison with *Path Coordinator*. 145
Figure 6.18: **Timeline:** Dynamically occurring and disappearing flows and corresponding changes in the number of idling gateways. 148

List of Tables

Table 2.1: Comparison of commodity protocols commonly used in wireless ad hoc networks. 11

Table 4.1: Hardware specs of Raspberry Pi Zero W. 29
Table 4.2: Parameter settings for measurement experiments. 32

Table 5.1: Assumed drone hardware specifications. 47
Table 5.2: Common parameters of test case generation. 76
Table 5.3: Performance evaluation for *Link Stabilizer*. 76
Table 5.4: Parameters of the random waypoint model. 77
Table 5.5: Parameters of the SRCM. 77
Table 5.6: Parameters of the hovering model. 77
Table 5.7: Performance evaluation for *Path Optimizer*. 90

Table 6.1: Common parameters of test case generation. 124
Table 6.2: Parameters for Connected RGG test cases. 125
Table 6.3: Parameters for Manhattan Grid Topology test cases. 125
Table 6.4: Parameters for the brute-force searching. 141
Table 6.5: Parameters for the timeline analysis. 146

Chapter 1

Introduction

1.1 Social Background

These days when all things are about to be connected to the internet across the globe, there are many forms of communication networks. Communication networks consist of two major architectural types – centralized architecture and decentralized architecture.

A typical example of former architecture is a cellular network. Wireless base stations are fixed to the structures and connected to the pre-laid optical fiber to forward traffic to the mobile core network. Mobile terminals hang out and communicate with the base station, and the base station entirely control the allocation of resources to each terminal.

The latter architecture, on the other hand, is also called an ad hoc network.

1.1.1 Ad Hoc Network

An ad hoc network is a self-organizing network whose arrangement is independent of pre-existing infrastructures such as base stations and fiber-optic cables because each node in the network act as a repeater. It is also a temporary network that can be built and destroyed quickly. Every node of an ad hoc network should be tolerant of dynamic topology changes and have the ability to organize itself into a network autonomously and cooperatively. Because of these specific characteristics, ad hoc networks have since

the 1990s played an important role as an instant communication means in environments where the network infrastructure is weak or does not exist, such as disaster areas, rural areas, and battlefields.

In general, nowadays, communication entity is shifting from human to thing, demand for MTC (machine type communicatino) is growing. Thus, ad hoc networks are also a hot topic even in urban areas where the broadband mobile communication systems are well developed and always available. More and more applications use ad hoc networks for local area communications, especially in key technologies that are expected to play a vital role in future society, such as an ITS (Intelligent Transportation System) supporting autonomous car driving, CPSs (Cyber-Physical Systems) like smart grids, and applications like the IoT and swarm robotics [1]. It is used not only on land, sea, and air, but also in space, including satellite constellations. Ad hoc networks are old and new technology.

1.1.2 Rise of HANETs

The original concept of an ad hoc network is a domain-specific network, whose protocol is specifically designed only for a specific environment or application. Thus, applications on different ad hoc networks have unique protocol stacks, and the trend of application diversification in recent years causes a situation where too many independent ad hoc networks are deployed in the same area. Because the ad hoc network is future key technology for social infrastructure, it is desirable to realize the interconnection among different ad hoc networks by protocol translation and operate these ad hoc networks as one integrated ad hoc network.

HANETs [2], an abbreviation of heterogeneous ad hoc networks, are huge integrated ad hoc networks consisting of various kinds of ad hoc networks such as WSN (Wireless sensor network), MANET (Mobile ad hoc network), VANET (Vehicular ad hoc network), FANET (Flying ad hoc network), Etc. The concept of HANETs is shown in Fig. 1.1. In HANETs, there are gateway nodes (GWs) that translate protocols to ensure interoperability among different ad hoc networks; that is, by passing communications

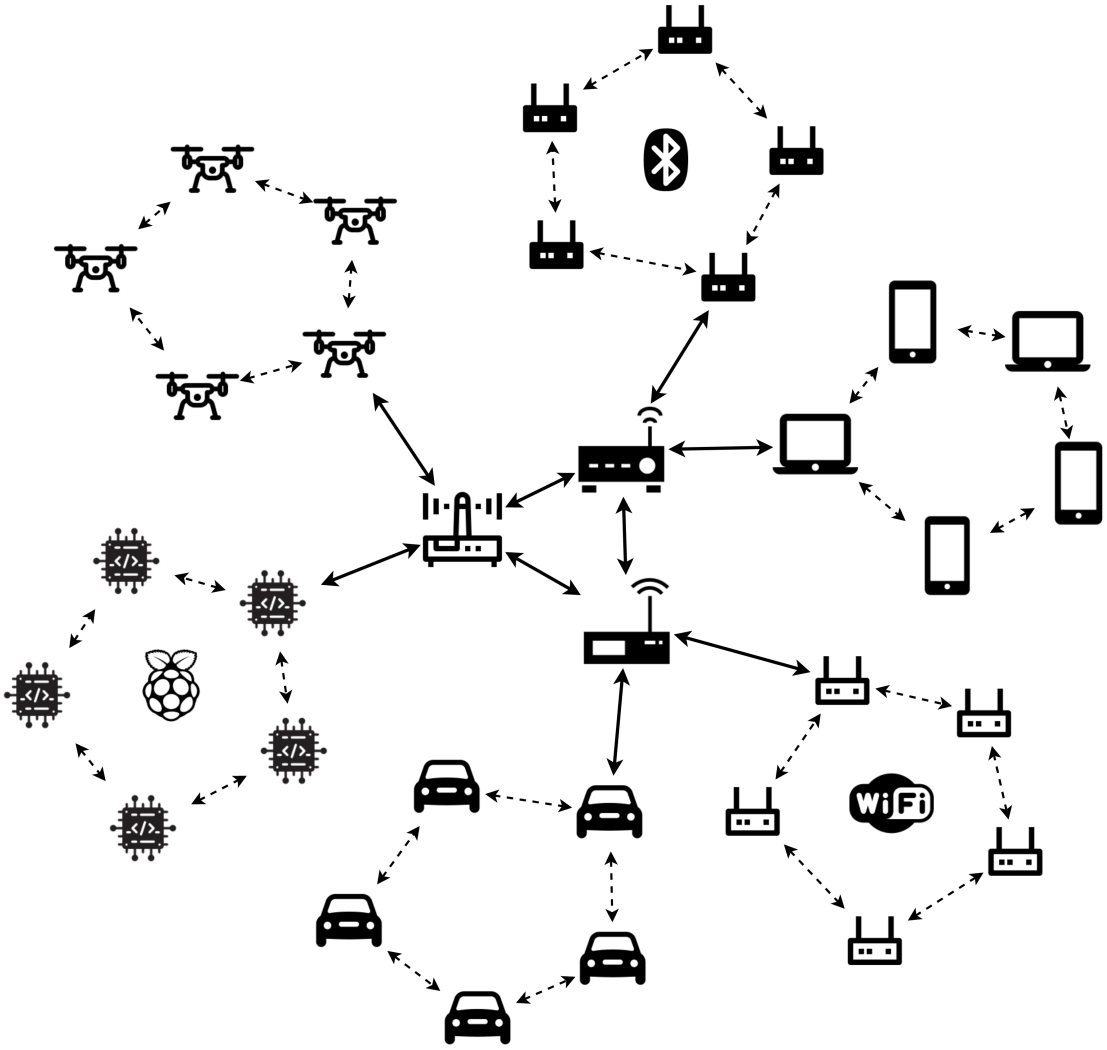


Figure 1.1: Concept of HANETs (Heterogeneous Ad Hoc Networks).

through gateways, every node can communicate with other network nodes without being aware of protocol differences. This innovative architecture enables scalable network construction even in the midst of diverse wireless protocols and active generational changes.

1.1.3 Future Potential of HANETs

Many researchers and developers are exploring the potential of the HANETs design concepts as a key technology to support future society. HANETs are not only used as a means for human-centric communication in both developing and developed countries [3] but are also used in environments where the communication entities are not humans, such as the IoT platform [4]. Ad hoc networks with a mixture of old and new terminals can be constructed as a means of communication in developing countries, and in urban areas, traffic offloading can be achieved through the simultaneous use of multiple protocols [3]. In the IoT domain, HetIoT (heterogeneous IoT), which has almost the same meaning as HANETs, has emerged as a response to recent smart homes and smart cities where various communication standards are in disarray. It is also expected in the swarm robotics – heterogeneous robotic swarms [5] have been proposed, in which robots with different specifications are involved.

1.2 Major Issues

One issue that is common to all communication networks is the detailed requirement for communication latency. Another issue original to HANETs is the challenge of maintaining gateway availability.

1.2.1 Latency

The network infrastructures tend to require more strict delay guarantee. For example, in 5G, which is the next-generation mobile communication system, the round-trip delay

between a terminal and a base station must be less than 1 millisecond due to the specification of URLLC (Ultra-Reliable and Low-Latency Communications) [6]. Low-delay communication increases the ability of applications, especially in a things-centric network. Therefore it is also important to consider communication delay in the study of ad hoc networks.

In HANETs, various applications share a single network. Each application in HANETs requests a different allowable delay, so there are many allowable delay constraints in one network. For such a situation, the best way to assure QoS is not by the conventional approach proposed in existing studies. In other words, if the conventional QoS control is implemented in HANETs, the algorithm computational cost would increase greatly for two reasons: the first is that HANETs have large-scale and complicated delay constraints as mentioned above, and the second is that the conventional approach is the average delay-based QoS which aims to minimize average communication delay in the whole network. However, because most applications only require that the communication delay should satisfy the allowable delay, the benefit of an even lower delay communication is not proportionate to its enormous computational cost.

1.2.2 Gateway Availability

In HANETs, inter-cluster communications always go through gateway, so gateway placement significantly impacts communication stability and end-to-end path hops. It is necessary to keep the gateway reachable from any node at any time. Maintaining gateway availability means maintaining HANETs.

In a static network model where nodes do not move, it can be solved by optimizing the gateway initial placement. In this case, it is partially reducible to the model of a wireless sensor network, and existing optimization algorithms may be applicable. However, if all nodes are mobile, a strategy that continually moves the gateway itself to the optimal position is required. The idea is to provide a mechanism that is robust against network fragmentation resistance.

1.3 Objectives and Motivation

The goal of this dissertation is to solve the above two issues simultaneously.

First, this study adopts the latency-based QoS, which aims to maximize the allowable delay satisfaction rate in the whole network. This approach can reduce the computational cost compared to the conventional one since it does not try excess optimization, and this is suitable in HANETs. Additionally, there are some cases in HANETs that the gateway processing delay is too large to ignore, since all of the gateways translate protocols for all communications passing them. The system model of this study can deal with both transmission delay on a link and processing delay on a gateway. It is very significant to realizing allowable delay-based QoS to consider both link delay and gateway delay in the same time unit.

Next, about the gateway availability maintenance, in general, there are many different types of gateway movement and placement problems, and many different studies have been conducted. There are two major scenarios. The first is the ground deployment type. This is a scenario that has long been used as a means of human-to-human communication, not limited to MTC, and is a hot ITS-related field these days, such as MANET and VANET. In this case, especially when deployed in urban areas, it is necessary to consider radio interference from other radio stations and ground structures. The other is the airborne deployment type. Recently, smart society-related technologies using drones have been attracting increasing attention, and this is a field where research is active in both industry and academia. In this case, it is possible to simplify the model by assuming that the effect of radio interference by ground sources and structures is negligible, and that there is a line-of-sight between neighboring nodes in terms of radio waves.

In this study, heterogeneous drone swarms will be treated and discussed as one of the specific targets, but the discussion will be reduced to an abstracted model, which may be applicable to other scenarios – further description will be provided in Chapter 3.

1.4 Overview of Dissertation

This chapter briefly introduced the background of this study – social demands, major challenges, and motivation. Chapter 2 describes differences between this work and other related studies and summarizes the novelty and importance. In Chapter 3, one specific application scenario is presented for modeling and problem formulation. Chapter 4 summarizes the results of preliminary experiments conducted prior to the actual proposal. Then, Chapters 5 and 6 design and evaluate gateway movement control methods. Finally, the conclusion of this study is put at the last part, Chapter 7. The suggestion for further study of the gateway mobility control in HANETs is leaved and opend for future.

Chapter 2

Related Work

This chapter summarizes the related studies and clarify what this study need to work on. Since this study deal with the design of the gateway relocation algorithm in heterogeneous drone swarms, ***1)*** interoperability of ad hoc networks and the architecture of HANETs, ***2)*** communication design of drone swarm, and ***3)*** a general gateway placement optimization were intensively investigated.

2.1 Interoperability of Ad Hoc Networks and HANETs

The interoperability of ad hoc networks has been discussed since around 2008, owing to the diversification of wireless communication protocols and routing protocols for ad hoc networks [8, 9].

Fujiwara et al. proposed a mechanism, called Ad hoc Traversal Routing (ATR), that enables interconnection between two ad hoc networks running different routing protocols [10]. An ATR node shares all addresses in its local network with other ATR nodes and manages the global address space. Moreover, each ATR node translates and relays route requests generated from the local network to the destination network, ensuring end-to-end reachability. Likewise, various studies have examined to assure the interoperability among different ad hoc networks through protocol conversion [11–16].

Some studies focus on HANETs themselves. For example, as for non-robot-related studies, Al-Saadi et al. reported in [3] that guaranteeing the interoperability of ad

hoc networks builds an integrated communication infrastructure that supports both new and old terminals, and it is instrumental in a developing country. Qiu et al. also predicted that the IoT platform would be HetIoT (Heterogeneous IoT; almost the same meaning as HANETs), which is not a vast homogenous system with a single standardized protocol but an integrated system of various models and algorithms on a large scale because of the recent trend for smart homes and smart cities [4].

A system like HANETs, which allows containing different protocols inside it, is very significant for a field like communication engineering where the technology progress is remarkable, and protocols are likely to diversify. HANETs are getting more and more attention from many researchers and will play a vital role in future society.

2.2 Drone Swarm

Swarm robotics is one of the bio-inspired robotics whose approach coordinates a lot of cheap and simple small robots in collective behavior. It is inspired by the habit of wild animals creating swarms in nature and aiming to acquire new problem-solving skills impossible with conventional robots. In swarm robotics, research on drone swarms is one of the hot topics for the last several years because mass-produced, inexpensive drones, called toy drones, have been widespread in society.

2.2.1 Rise of Protocol Diversity

The advent of inexpensive drones called toy drones has popularized personal ownership of drones for leisure purposes. The drone-related market continues to expand, and these days people can get a unique drone having the desired specification by purchasing parts and building them up [17–20]. As an example, Table 2.1 lists the wireless protocols commonly used for drone control and data transmission. Different scenarios or missions require different wireless protocols, and the communication module selection is a crucial factor in drone design. The most widely used is Wi-Fi; however, WiMAX is suitable for long-range communications, LTE or mmWave for broadband communications, and

other low-power protocols like ZigBee and Bluetooth for beacons of aircraft control signals.

Miniature self-build drones weigh a few hundred grams as of now, but the research and development of the latest small drones have been progressing rapidly – a 33-gram airframe equipped with an RGB camera and an infrared camera can fly at a maximum speed of 21.5km/h in the wind and rain for 25 minutes within a radius of 2 km from the operator, almost silently [21]. This cutting-edge drone is currently only available to the military and law enforcement agencies, but within a few years, we can expect to see some consumer drones having similar specifications. Drones are becoming more and more diverse, not only in communication protocols but also in aircraft performance.

Although there are several concept papers on drone swarms with heterogeneous protocols or aircraft specifications [5, 7], no study has yet reached the level of a proof-of-concept experiment. However, against the social background of the recent dizzying updates surrounding drones, it is evident that the architecture of HANETs will play a vital role in drone swarm applications, which confirms the value and novelty of this study.

Table 2.1: Comparison of commodity protocols commonly used in wireless ad hoc networks.

	Wi-Fi	**WiMAX**	**LTE**	**mmWave**
Spectrum	2.4/5GHz	2.3/3.5/5.8GHz	1.8-2.7GHz	57-64GHz
Family	802.11n/ac	802.16e	3GPP	802.11ad
Bandwidth	20/40MHz	1.75-20MHz	20MHz	2.16GHz
Bitrate	<500Mbps	6-21Mbps	<75Mbps (up) <300Mbps (down)	693Mbps-6.76Gbps
Tx Range	<100-500m	<10km	<2km	<130m (directional) <20m (omni)
Low Power	-	-	-	-

	ZigBee	**Bluetooth**
Spectrum	2.4GHz	2.4GHz
Family	802.15.4	802.15.1
Bandwidth	2MHz	2MHz
Bitrate	20-250kbps	125kbps-2Mbps
Tx Range	<30m	<100m
Low Power	✓	✓

2.2.2 Multi-hop Communication

Swarm robotics has been the subject of many studies, but most focused on the theoretical analysis of behavior models [19, 22–24]; few have proposed technically feasible systems [7, 25–30], and even fewer have concentrated on the communication design or distributed network formation [31–35].

Gutiérrez et al. analyzed the processing delay on embedded Linux based on an actual experiment to prepare to implement real-time communication among swarm robots [31]. They found that the delay caused by Linux's standard network stack is too extensive to be ignored. Yuan et al. investigated the degree of connection reliability of LoRa, Wi-Fi, and LTE, which are frequently adopted as communication protocols among robots [32]. Various numerical facts pointed out in their paper, such as the fact that Wi-Fi can communicate stably up to 80 meters visibility, are reflected in the evaluation parameters described in Chapter 5–6

There are proposals of node mobility control algorithms based on topology control for achieving stabilized communication, reduced delay, or power consumption [33–35]. They target a dense ad hoc network and make the network sparse by cutting redundant multi-paths. Mi et al. also investigated a collision avoidance mechanism by suppressing the excessive proximity of drones to each other [35]. However, all these studies assume that every node in the network is under control and relocatable and does not allow to include multiple independent protocols such as heterogeneous drone swarms. The performance of their algorithms is limited inside a single, homogeneous network - theirs cannot be applied to the model of this study.

Using drones as flying base stations, along with the Sattelite Internet, is the key idea to achieving 100% coverage of the Internet and is attracting many researchers and developers [25, 36–45].

In general, a drone is often called a mobile relay in the model where drones relay the communication among ground stations. Mobile relays are used in a variety of ways [36–40]. P. Zhan et al. dealt with the most common model of communication relay by some UAVs between nodes fixed on the ground [37]. In order to use UAVs

as relays, which cannot remain stationary in the air and are constantly on the move, their focus is on the design of the communication handover mechanism. Wei Wang et al. proposed a method to extend the network lifetime by finding sensors with heavy traffic and bypassing them using mobile relays to load balancing in general WSNs [39]. Although it is close to the model of this study in that some mobile nodes control traffic patterns, their algorithm is for WSNs assuming all communications go to specific destination nodes (sink nodes) and does not apply to the situation assumed in this paper. K. Li et al. tackled communications between ground stations across a mountain relayed by multiple UAVs circling and waiting near the summit [40]. They mainly focus on the TDMA-based resource allocation method. L. Kong et al. proposed an optimal positioning and attitude maintaining method targeting drones with mmWave antennas, which have straightness of radio wave characteristics [36]. Both they and the author proposed a mobility control method to keep the optimal position for each drone; however, their study only considers the physical layer reachability and does not support per-communication optimization.

All of the above studies assume a homogeneous network for inter-drone communications, which is essentially different from HANETs. Their modeling methods and algorithm designs can be used as a reference, though there is no direct overlap with this study.

Introducing a hierarchical structure into the network enables efficient communication, especially when constructing a large-scale wireless multi-hop network [25, 41–45]. J. Wang et al. covered a model of multiple geographically dispersed FANETs (flying ad hoc networks) consisting of mini-drones connected via small drones flying at higher altitudes or satellites [42]. They proposed a so-called cluster-head selection algorithm, in which any drone can become a gateway, and it is different from the model of this study, allowing only specific drones to have the gateway function. The same can be said for the study presented by Q. Zhang et al [43]. They worked on a model in which long-range communication UAVs relay short-range drones; however, they assumed the same communication protocol for both drones and UAVs, focused on the subcluster configu-

ration and the cluster-head selection. On the other hand, N. Saputro et al. attached the Internet reachability to the closed VANET (vehicular ad hoc network) using a drone swarm [45]. In their model, the drone swarm builds a FANET with IEEE 802.11s and covers the ground VANET with 802.11p, and some drones in the FANET function as LTE-gateway. All communications from vehicles go to the LTE base stations – vehicle to vehicle, vehicle to drone, drone to drone, and drone to LTE base stations. It is similar to this study regarding the traffic pattern boarding to heterogeneous networks but is closer to the study of WSNs since all communications are bound for LTE-gateways; theirs corresponds to a simplified version of this study.

The author have not found any studies dealing with essentially the same theme, though many related studies deal with similar models, as reported above.

2.2.3 Realtime Video Transmission and QoS

Realtime video transmission and QoS-aware traffic control are not limited to drone-related products but has been widely discussed for communication networks since 1990s [46–55]. Recently, for example, S. Zaidi et al. proposed a low-latency protocol for live video streaming in vehicle-to-vehicle communication [56]. This field is still being actively researched from various approaches, including video codec improvement.

When taking aerial photographs of wildlife with a drone, it is necessary to fly above a certain altitude so that the noise generated by the rotor does not affect the animals' behavior. In most fieldwork, drones are flown at altitudes from 100 to 200 meters – with a camera view angle of 120 degrees, a radius on the ground from 170 to 340 meters can be captured [57]. This type of fieldwork needs to start with finding the location of target animals, and there are many studies on reducing the human load by drones' autonomous searching, for example, C. Burke et al. used infrared cameras instead of RGB cameras to find targets quickly [58], and J.-A. Vayssade et al. used machine learning to identify and track target animals automatically [59]. This study makes it possible to construct a large-scale drone network; it is a novel solution for wildlife monitoring since it provides a much wider video coverage area than a conventional

method of flying a few drones on patrol.

Other examples include Y. Sato et al.'s facility monitoring of the radioactive area in Fukushima [60]. In summary, realtime video transmission is a hot topic that is expected to play an active role in various situations concerning robot automation technology.

2.2.4 Formation Control

A mid-air collision of drones leads directly to their crash, which causes economic loss and endangers humans and objects on the ground. Formation control is a critical research topic for drone swarms. However, most of them are still in theoretical studies, and only a few have been completed to the level of demonstration [61–63].

B. Yu et al. proposed a control algorithm that maintains the relative positions among drones to realize the drone's formation flight [61]. After the first deployment, every drone moves to an appropriate position to its neighbors and then moves as a swarm while maintaining its initial formation. H.-J. Kim et al. have also made similar considerations, and they have designed and implemented a drone for the drone light show [62]. Their algorithm allows only the leader drone to get commands from the ground flight controller. The other drones participate in a fully meshed network constructed by XBee to exchange GPS-acquired 3D position information with neighbors and maintain a desired relative position to the leader.

In addition to the mere relative position maintenance, some studies autonomously configure an ideal formation, e.g., H. Li et al. covered a mobility control to avoid radio channel interference [63]. The topology control considering the communication quality is similar to the theme of this study. However, their study is an average optimization for future communications, not an adaptive optimization for current communications. Many of the related existing studies only dealt with the former model, which is essentially different from this study that deals with the latter. Therefore, it is impossible to apply these existing algorithms to this study.

In summary, the algorithm to keep the drone's hovering formation and recover them as necessary against some external forces like blasts is mature enough. Researchers are

currently tackling more advanced and sophisticated algorithms enabling drones to three-dimensional high-speed movement. The author has confirmed no technical difficulties in the situation assumed in this study, i.e., scattering drones over a monitoring target and having them wait in hovering.

2.3 Generic Gateway Placement Optimization

The gateway placement optimization problem, which has attracted much attention for the sink node placement problem in WSNs recently, is also related to the network planning for wired backbone networks; thus, so many studies exist around this subject [64–70].

The most part focuses on finding the optimal initial gateway placement using meta-heuristics such as ACO, PSO, or GA, under the consideration of the sensor node distribution as well as the average shortest path length or end-to-end communication delay [64–67]. As an opposite approach, Miao et al. proposed a routing algorithm to minimize average communication delay by optimizing the selection of location fixed gateways, assuming collecting sensing data via satellite links [68]. In another unique study, Kim et al. transformed a generic gateway placement problem into a mathematical optimization problem and discussed its properties, such as computational complexity [69].

In the past studies of WSNs and network planning for wired backbones, as far as the author knows, authors only concentrated on a global optimization for an entire network, e.g., realizing average end-to-end delay reduction. Moreover, the node position is fixed after the optimized initial placement - they do not assume the node mobility like HANETs and the dynamic gateway relocation for each communication. Hence, no other studies can be applied to the target situation of this study.

2.4 Summary

In this chapter, comprehensive survey have been conducted from HANETs to drone swarm and gateway placement optimization problems. Eventually, the following facts have still remained in all existing studies.

- Assuming all nodes join in a single, homogeneous ad hoc network for their communication channel
- Assuming all nodes are under a single swarm control algorithm, which can arbitrarily manipulate node locations
- Designing optimal node location aiming for average performance improvement in a static network – optimization for communications that may occur in the future, not for those that are currently occurring

Chapter 3

Application Modeling and Problem Statement

This chapter illustrates the target situation of this study and the course of the proposal by describing a concrete application scenario.

3.1 Protocol Stack

The protocol stack assumed this time is shown in Fig. 3.1 - it is based on a commonly implemented routing protocol (L3) and allows free stack design for wireless protocols (L1–L2) and application protocols (L7). The L3 protocol is an extended HWMP (hybrid wireless mesh protocol) and is considered to be installed as standard software among all drones. The gateway controller is an application software and is regarded to run only on gateway drones' computers. The main proposal is the gateway controller, which is described in Chapter 5–6, though a conceptual design of the L3 protocol is also proposed.

This study supposes a DIY drone built with a commercial drone kit and target the heterogeneous drone swarms consisting of two protocols – Wi-Fi and WiMAX. There may be enough demand to create cluster mixing protocols having different radio frequencies, communication bandwidth, and communication ranges like Wi-Fi and WiMAX. One of the realistic scenarios would be the following: ***1)*** researchers who own both a

drone swarm using Wi-Fi for remote control and video transmission and one that uses WiMAX *2)* build and deploy a small number of gateway drones equipped with both Wi-Fi and WiMAX communication modules *3)* and then construct a large-scale hybrid swarm consisting of two sub-swarms, i.e., heterogeneous drone swarms.

Alternatively, the part of Wi-Fi and WiMAX can be replaced by LTE, mmWave, or some other radio protocol.

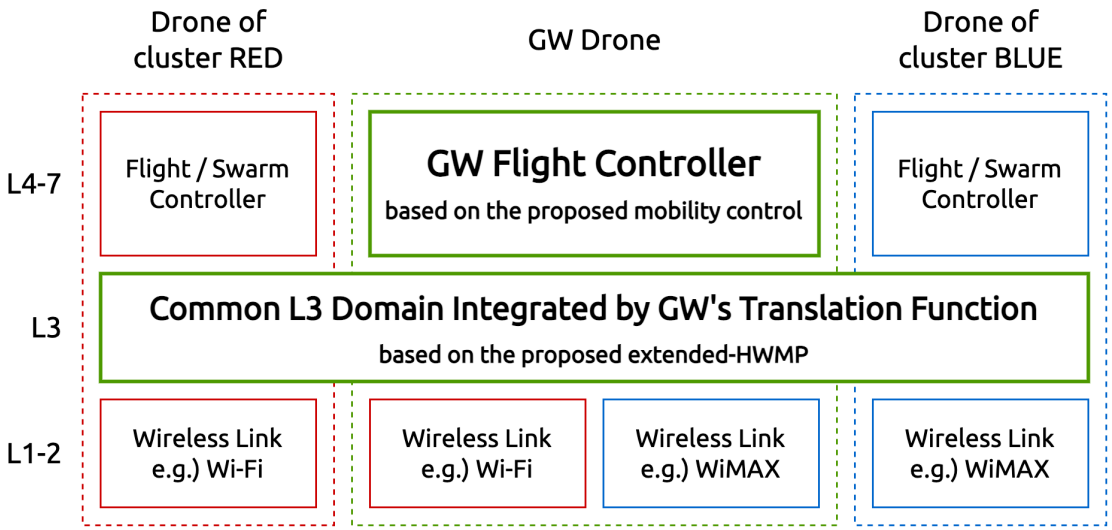


Figure 3.1: Protocol stack of the target application: two clusters are interconnected by one gateway cluster.

3.2 Application Scenario and Objective

As for an application scenario, a field video monitoring system is assumed, shown in Fig. 3.2, such as unmanned plant surveillance for security or wild animals monitoring for academic research, which is in high demand in contemporary society. Note that details of the assumptions around the HANETs formation will be explained in Chapter 5–6.

1. Wi-Fi drones and WiMAX drones are camera-equipped and on standby at high altitudes by hovering above the monitoring target.

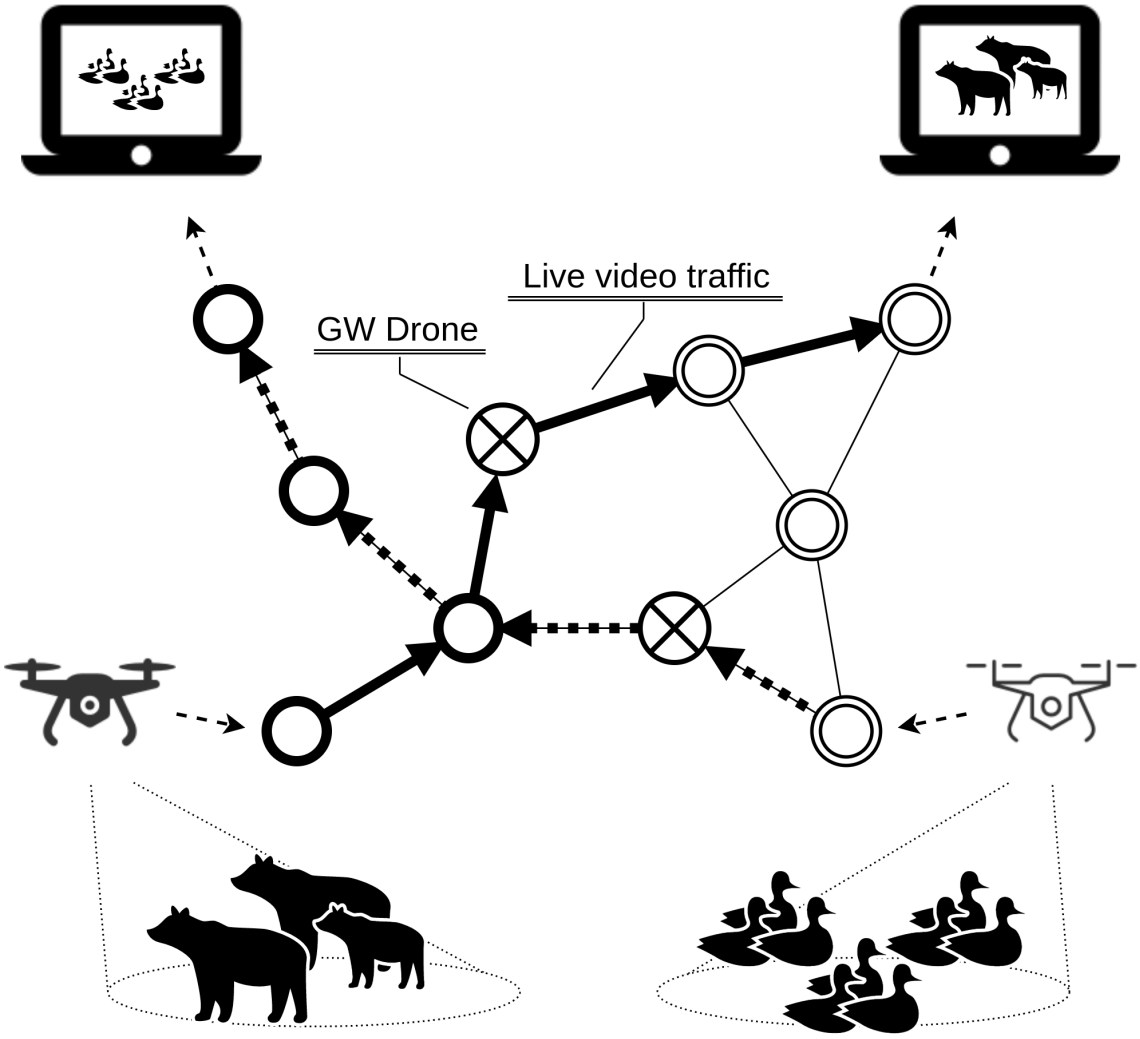


Figure 3.2: Concept of heterogeneous drone swarms.

2. Wireless mesh is configured independently in the Wi-Fi drones and WiMAX drones.

3. With some gateway drones, HANETs are configured among all.

4. Operators connect to the HANETs from any location via the nearest drone and watch the remote video via multi-hop streaming.

Suppose operators are watching a specific video for a relatively long time. In that case, they may want to receive a low-latency, high-quality video by relocating the gateway at each video switching, rather than a high-latency, low-quality video, even though the gateway relocation requires some preparation time, i.e., overhead, due to the physical movement. Besides, in the ideal system, what operators do to link multiple swarms is simply add some gateways – gateways do everything well and hide behind complex processes, like a plug-and-play basis. If we follow this idea, modifications to non-gateway drones also should be avoided as much as possible. Hence, conditions must be set as the algorithm can control only the gateway drones' movement, and other drones are not restricted from freely moving. Non-gateway drones should not think about maintaining inter-cluster connectivity but should concentrate only on maintaining intra-cluster connectivity. Thus, what this study should consider here is summarized as below:

- **Background:** Interconnect heterogeneous clusters by putting some gateways. Occurred inter-cluster communication via gateway.
- **Objective:** Achieve stabilized and low-delay communication by moving gateway autonomously.
- **Condition:** Assuming each cluster has independently implemented flight controllers, non-gateways must not be restricted from freely moving.

3.3 Simplification

As an introductory study, the following additional conditions simplify the problem.

1. Interconnect two clusters.

2. Only the number of path hops affects the end-to-end communication delay and omits the consideration of dynamic link status like RSSI, packet loss ratio, etc.

3. The drone flies at enough high altitude to consider that all communications exist within line-of-sight distance. The free space propagation model is adopted to describe the radio characteristics, and no interference occurs.

4. In addition to the above, all drones fly at the same altitude. No three-dimensional behavior occurs, and the computer simulation is performed in a two-dimensional field.

5. The drone's flight time is only a probabilistic consideration and is not treated numerically.

Besides, this study assumes all drones are equipped with a collision-avoidance system as an independent communication unit; that is, navigation devices like GPS are installed so every drone can advertise its position and speed by sending beacons [71]. The subsequent studies should tackle the more complicated models considering the above simplifications.

Chapter 4

Experimental Analysis of Communication Relaying Delay in Low-Energy Ad Hoc Networks

4.1 Introduction and Related Work

These days, communication entities are shifting from humans to things; the network infrastructures tend to require a more strict delay guarantee, and the ad hoc network is no exception. There have been many prior studies about delay-aware communication in the field of ad hoc networks [46, 68, 72]. Most of these focus on the link delay and only a few consider both node and link delays [68]. However, in some situations where the power consumption is severely limited (e.g., with WSN), the communication relaying cost of small devices with low-power processors may not be negligible for the end-to-end delay of each communication.

It is necessary to discuss, on the basis of actual data measured on wireless ad hoc networks, how much the link and node delays account for the end-to-end delay. In the field of wired networks, there have been many studies reporting measurement experiments of packet processing delay as well as various proposals for performance improvement [79–82] In addition, the best practice of QoS measurement has been discussed in the IETF [83]. In the past, measurement experiments on ASIC routers have been

carried out for the purpose of benchmarking routers working on ISP backbones [73–75]; in contrast, since the software router has emerged as a hot topic in the last few years, recent studies mainly concentrate on the bottleneck analysis of the Linux kernel's network stack [76–78]. There has also been a study focusing on the processing delay caused by the low-power processor assuming interconnection among small robots or embedded systems [31, 84]. However, no similar measurement exists in the field of wireless ad hoc networks. Therefore, many processing delay models have been considered so far, e.g., simple linear approximation [85] or queueing model-based nonlinear approximation [86], but it is hard to determine which one is the most reasonable for wireless ad hoc networks.

This work analyzes the communication delay in an ad hoc network through a practical experiment using Raspberry Pi Zero W, assuming an energy-limited ad hoc network composed of small devices with low-power processors. The goal is to support the design of QoS algorithms on ad hoc networks by clarifying the impact of software packet processing on the end-to-end delay and presenting a general delay model to which the measured delay can be adapted. This is an essential task for future ad hoc networks and their related technologies.

First, the structure of the Linux kernel network stack is described briefly in Sect. 4.2. The details of the measurement experiment is explained in Sects. 4.3 and 4.4, and the results are reported in Sect. 4.5. Sect. 4.6 concludes with a brief summary and mention of future work.

4.2 Network Stack of Linux Kernel

This section presents a brief description of the Linux kernel's standard network stack from the viewpoints of the packet receiving and sending sequences.

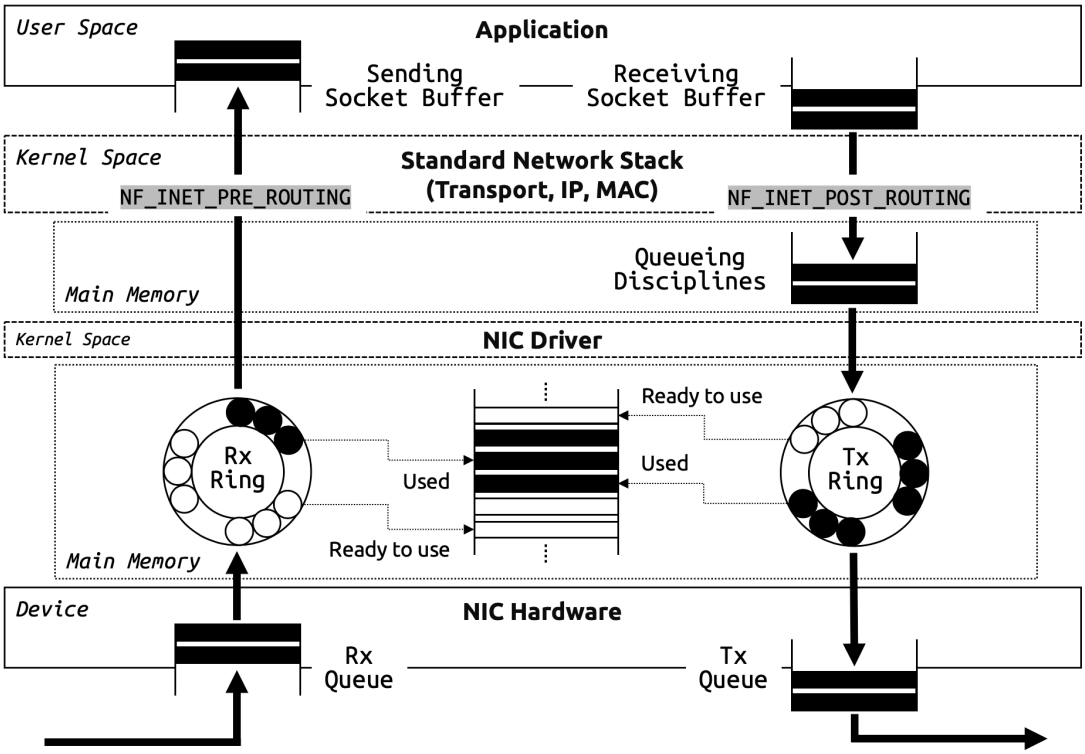


Figure 4.1: Packet queueing in the Linux kernel standard network stack.

4.2.1 Generic Packet Receiving Sequence

Figure 4.1 shows the flow of packets in the network stack from the perspective of packet queueing.

First, as the preparation for receiving packets, the NIC driver allocates memory resources in RAM that can store a few packets, and has packet descriptors (Rx descriptors) hold these addresses. The Rx ring buffer is a descriptor ring located in RAM, and the driver notifies the NIC of the head and tail addresses of the ring. The NIC then fetches some unused descriptors by direct memory access (DMA) and waits for the packets to arrive. The workflow after the packet arrival is as follows. As a side

note, the below sequence is a receiving mechanism called new API (NAPI) supported in Linux kernel 2.6 or later.

i) Once a packet arrives, NIC writes the packet out as an sk_buff structure to RAM with DMA, referring to the Rx descriptors cached beforehand, and issues a HardIRQ after the completion.

ii) The IRQ handler receiving HardIRQ pushes it by `napi_schedule()` to the poll_list of a specific CPU core and then issues SoftIRQ so as to get the CPU out of the interrupt context.

iii) The soft IRQ scheduler receiving SoftIRQ calls the interrupt handler `net_rx_action()` at the best timing.

iv) `net_rx_action()` calls `poll()`, which is implemented in not the kernel but the driver, for each poll_list.

v) `poll()` fetches sk_buff referring to the ring indirectly and pushes it to the application on the upper layer. At this time, packet data is transferred from RAM to RAM; that is, the data is copied from the memory in the kernel space to the receiving socket buffer in the user space by `memcpy()`. Repeat this memory copy until the poll_list becomes empty.

vi) The application takes the payload from the socket buffer by calling `recv()`. This operation is asynchronous with the above workflows in the kernel space. The packet receiving sequence is completed when all the payloads have been retrieved.

4.2.2 Generic Packet Sending Sequence

In the packet sending sequence, all the packets basically follow the reverse path of the receiving sequence, but they are stored in a buffer called QDisc before being written to the Tx ring buffer (Fig. 4.1).

The ring buffer is a simple FIFO queue that treats all arriving packets equally. This design simplifies the implementation of the NIC driver and allows it to process packets fast. QDisc corresponds to the abstraction of the traffic queue in the Linux kernel and makes it possible to achieve a more complicated queueing strategy than FIFO without modifying the existing codes of the kernel network stack or drivers.

QDisc supports many queueing strategies; by default, it runs in `pfifo_fast` mode. If the packet addition fails due to a lack of free space in QDisc, the packet is pushed back to the upper layer socket buffer.

4.3 Design of Experiments

As discussed in Sect. 4.1, the goal of this study is to evaluate the impact of software packet processing, induced by packet relaying, to the end-to-end delay, on the basis of an actual measurement assuming an ad hoc network consisting of small devices with low-power processors. Figure 4.2 shows the experimental environment, whose details are described in Sect. 4.4.

4.3.1 Definitions of Delays in Network

Below defines the classification of communication delays. Both *processing delay* and *queueing delay* correspond to the application delay in a broad sense.

- **End-to-end delay:** Total of node delays and link delays
- **Node delay:** Sum of *processing delay*, *queueing delays*, and any processing delays occurring in the network stack
- **Link delay:** Sum of all other delays occurring under the NIC driver, such as queueing delay of the NIC internal buffer, transmission delay and propagation delay at the communication medium

- ***Processing delay*:** Processing delay for packet relaying generated by the application in user space
- ***Queueing delay*:** Queueing delay at the socket buffer

4.3.2 Proxy: How to Relay Packets

The proxy node (Fig. 4.2) relays packets with the three methods below, and the effect of each in terms of the end-to-end delay is evaluated. By comparing the results of *OLSR* and *AT*, the delay caused by packets passing through the network stack can be clarified.

- **Kernel routing *(OLSR)*:** Proxy relays packets by kernel routing based on the OLSR routing table. In this case, the relaying process is completed in kernel space because all packets are wrapped in L3 of the network stack. Accordingly, both *processing delay* and *queueing delay* defined above become zero, and node delay is purely equal to the processing delay on the network stack in the kernel space.
- **Address translation *(AT)*:** Proxy works as a TCP/UDP proxy, and all packets are raised to the application running in the user space. The application simply relays packets by switching sockets, which is equivalent to a fixed-length header translation.
- **Encryption *(Enc)*:** Proxy works as a TCP/UDP proxy. Besides *AT*, the application also encrypts payloads using AES 128-bit in CTR mode so that the relaying load depends on the payload size.

4.3.3 Measurement Conditions

For each relaying method, measurements with variations of the following conditions are conducted. All the results are expressed as multiple percentile values in order to

remove delay spikes. Because the experiment takes several days, the RSSI of the ad hoc network including five surrounding channels are recorded.

- **Payload size**
- **Packets per second (pps)**
- **Additional CPU load *(stress)***

4.4 Measurement Methods

This section explains: the technical details of the experimental environment and measurement programs.

4.4.1 Building OLSR Ad Hoc Network

This experiment uses three Raspberry Pi Zero Ws (see Table 4.1 for the hardware specs). The Linux distributions installed on the Raspberry Pis are Raspbian and the kernel version is 4.19.97+.

Table 4.1: Hardware specs of Raspberry Pi Zero W.

SoC	Broadcom BCM2835
CPU	ARM1176JZF-S (ARMv6) 1core 1GHz
RAM	LPDDR2 SDRAM 512MB
Wi-Fi	IEEE 802.11b,g,n 2.4GHz
Power	150mA (0.75W)
Size	65mm x 30mm (9g)

It uses OLSR (RFC3626), which is a proactive routing protocol, and adopts `olsrd` as its actual implementation. Since all three of the nodes are location fixed, even if the experiment used a reactive routing protocol like AODV instead of OLSR, only the

periodic Hello in OLSR will change the periodic RREQ induced by the route cache expiring; that is, in this experiment, whether the protocol is proactive or reactive does not have a significant impact on the final results.

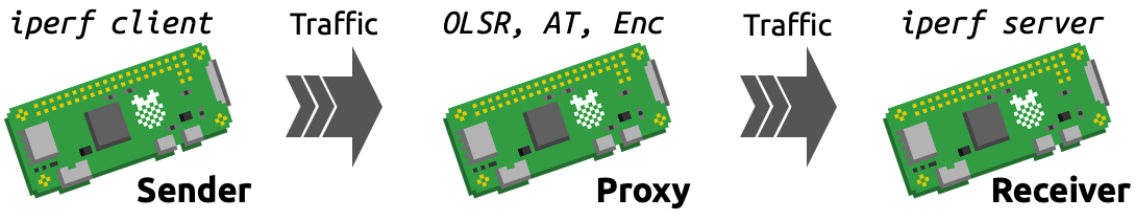


Figure 4.2: Three Raspberry Pis connected logically inline.

The ad hoc network uses channel 9 (2.452 GHz) of IEEE 802.11n, transmission power is fixed to –31 dBm, and bandwidth is 20 MHz. As WPA (TKIP) and WPA2 (CCMP) do not support ad hoc mode, the network is not encrypted.

Although the three nodes can configure an OLSR mesh, as they are located physically close to each other, the sender and receiver drop OLSR Hello from the receiver/sender as well as the ARP response by Netfilter so that the network topology becomes a logically inline single-hop network, as show in Fig. 4.2.

4.4.2 Preparation of Traffic Generator

The traffic generator `iperf` measures the UDP performance as it transmits packets from sender to receiver via proxy. The iperf embeds two timestamps and a packet ID in the first 12 bytes of the UDP data section (Fig. 4.3), and the following measurement programs use this ID to identify each packet. Random data are generated when iperf starts getting entropy from `/dev/urandom`, and the same series is embedded in all packets.

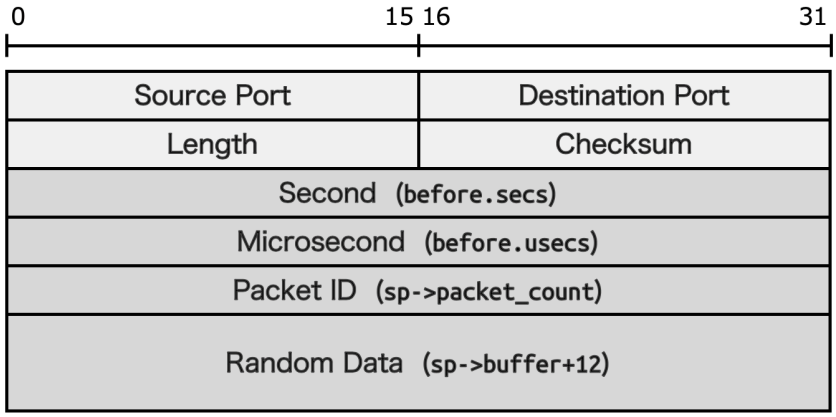


Figure 4.3: Format of UDP probe generated by iperf.

4.4.3 Implementation of the Kernel Module for Measurement

The author creates a loadable kernel module using Netfilter and measure the *queueing delay* in receiving and sending UDP socket buffers. The workflow is summarized as follows: the module hooks up the received packets with `NF_INET_PRE_ROUTING` and the sent packets with `NF_INET_POST_ROUTING` (Fig. 4.1), retrieves the packet IDs iperf marked by indirectly referencing the sk_buff structure, and then writes them out to the kernel ring buffer via `printk()` with a timestamp obtained by `ktime_get()`.

4.4.4 Implementation of the Proxy

The proxy program is the application running in the user space. It creates `AF_INET` sockets between sender and proxy as well as between proxy and receiver and then translates IP addresses and port numbers by switching sockets. Furthermore, it records the timestamps obtained by `clock_gettime()` immediately after calling `recv()` and `sendto()`, and encrypts every payload data protecting the first 12 bytes of metadata

marked by iperf so as not to be rewritten. The above refers to the UDP proxy; the TCP proxy simply uses `socat`.

4.4.5 Additional CPU Load at Proxy

A dummy process whose CPU utilization rate is limited by `cpulimit` is introduced as a controlled noise of the user space in order to investigate and clarify its impact on the node delay.

4.5 Results and Discussion

The delay measurement experiments are performed under the conditions shown in Table 4.2 using the methods described in the previous section. Due to the space constraints, the results of the preliminary experiment are omitted. Note that all experiments were carried out at the author's home; due to the Japanese government's declaration of the COVID-19 State of Emergency, the author have had to stick to the "Stay home" initiative unless absolutely necessary.

Table 4.2: Parameter settings for measurement experiments.

Payload size	100, 200, 300, ..., 1400 [bytes]
Packets per second	200, 400, 600, ..., 1200 [pps]
Packets per second (*Enc*)	20, 40, 60, ..., 400 [pps]
Additional CPU load (*stress*)	0, 50, 90 [%]
Transmission duration	5 [sec]
Number of samples	50

4.5.1 Time Variation of Received Signal Strength Indicator

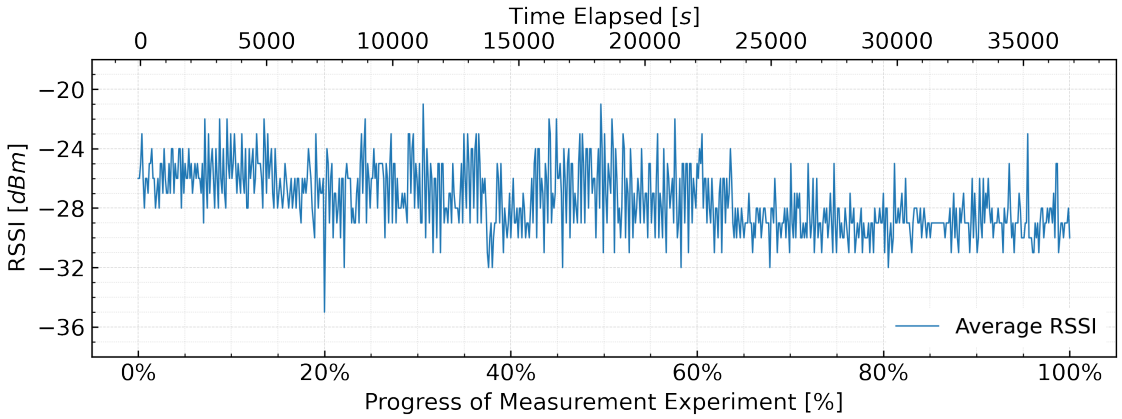
The experiment was divided into nine measurements. Figure 4.4a shows the time variation of RSSI during a measurement. The author was unable to obtain SNRs owing to the specifications of the Wi-Fi driver, and thus the noise floors were unknown, but the ESSIDs observed in the five surrounding channels were all less than –80 dBm. The RSSI variabilities were also within the range that did not affect the modulation and coding scheme (MCS) [87]; therefore, it appears that the link quality was sufficiently high throughout all measurements.

4.5.2 Time Variation of Node Delay and Coffee-break Effect

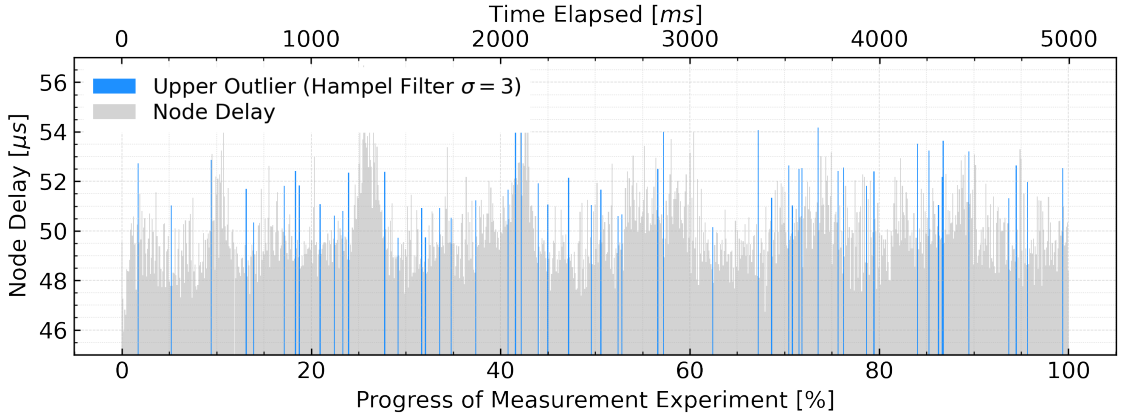
Figures 4.4b, 4.4c, and 4.4d shows the average time variations of node delay, which were the results under the condition of 1000 bytes, 200 pps, and 0% *stress*. The blue highlighted bars indicate upper outliers (delay spikes) detected with a Hampel filter ($\sigma = 3$). There were 53 outliers in *OLSR*, 115 in *AT*, and 9 in *Enc*.

In general, when the CPU receives periodic interrupts (e.g., routing updates, SNMP requests, GCs of RAM), packet forwarding is paused temporarily so that the periodic delay spikes can be observed in the end-to-end delay. This phenomenon is called the "coffee-break effect" [75] and has been mentioned in several references [73, 76, 77].

For this experiment, as seen in the results of *AT* (Fig. 4.4c), in the low-energy ad hoc networks, it is evident that the CPU-robbing by other processes like coffee-break had a significant impact on the communication delay. Incidentally, there were fewer spikes under both ***1)*** *OLSR* and ***2)*** *Enc* than under *AT*. ***1)*** Since the packet forwarding was completed within the kernel space, node delay was less susceptible to applications running in the user space. ***2)*** Since the payload encryption was overwhelmingly CPU-intensive, the influence of other applications was hidden and difficult to observe from the node delay.

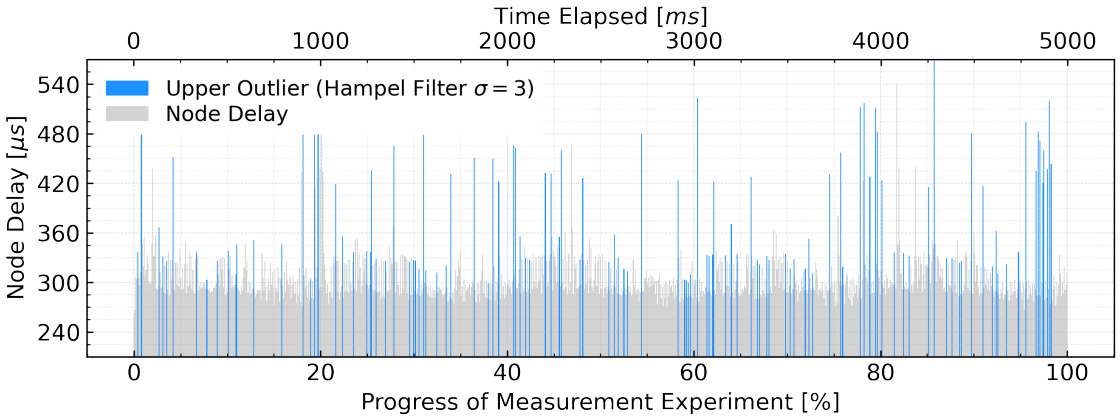


(a) RSSI: Average scores for sender, proxy, and receiver.

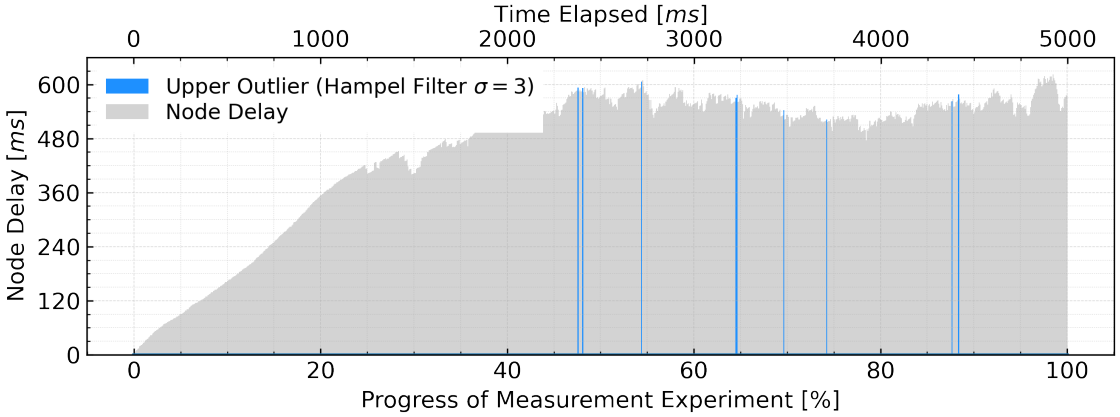


(b) Node delay (*OLSR*): 1000 bytes, 200 pps, 0% *stress*

Figure 4.4: Timelines: Average time variation of RSSI and 100th percentile node delays during the measurement.



(c) Node delay (*AT*): 1000 bytes, 200 pps, 0% *stress*



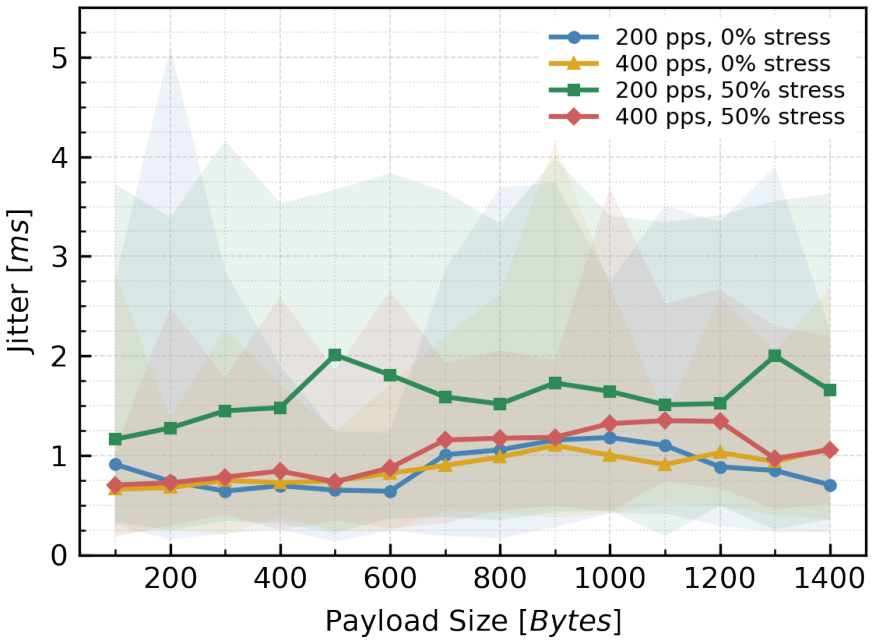
(d) Node delay (*Enc*): 1000 bytes, 200 pps, 0% *stress*

Figure 4.4: Timelines: Average time variation of RSSI and 100th percentile node delays during the measurement.

4.5.3 Jitter and Packet Loss Rate

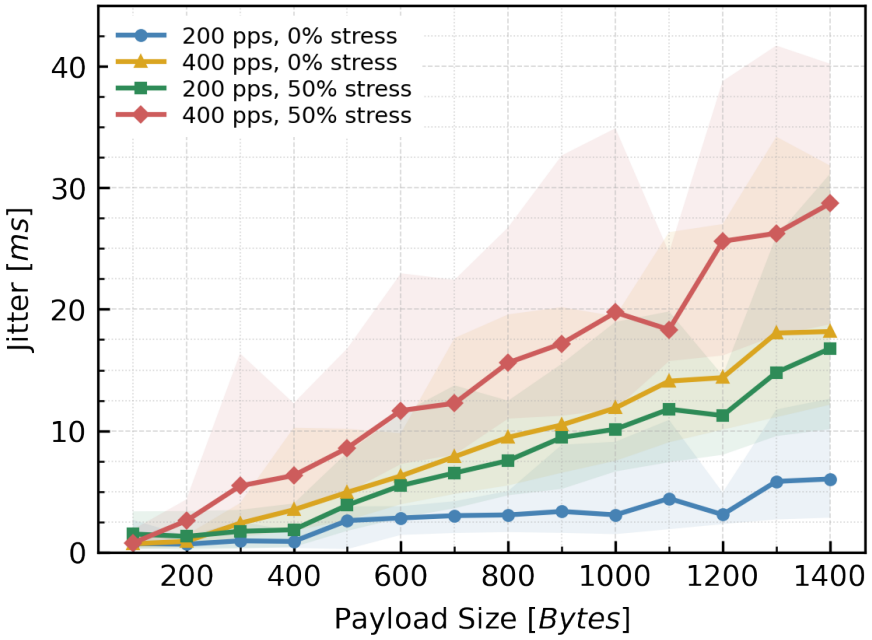
Figures 4.5a and 4.5b shows the jitter of one-way communication delay. Lines represent the average values, and the areas between the minimum and the maximum are filled in. There were no significant differences between *OLSR* and *AT*, which suggests that lifting packets to the application layer does not affect jitter. Jitter increased in proportion to the payload size only in the case of *Enc*.

Similarly, only in the case of *Enc* with 200 pps or more, the packet loss rate tended to increase with payload size, drawing a logarithmic curve as seen in Fig. 4.5c; in all other cases, no packet loss occurred regardless of the conditions.

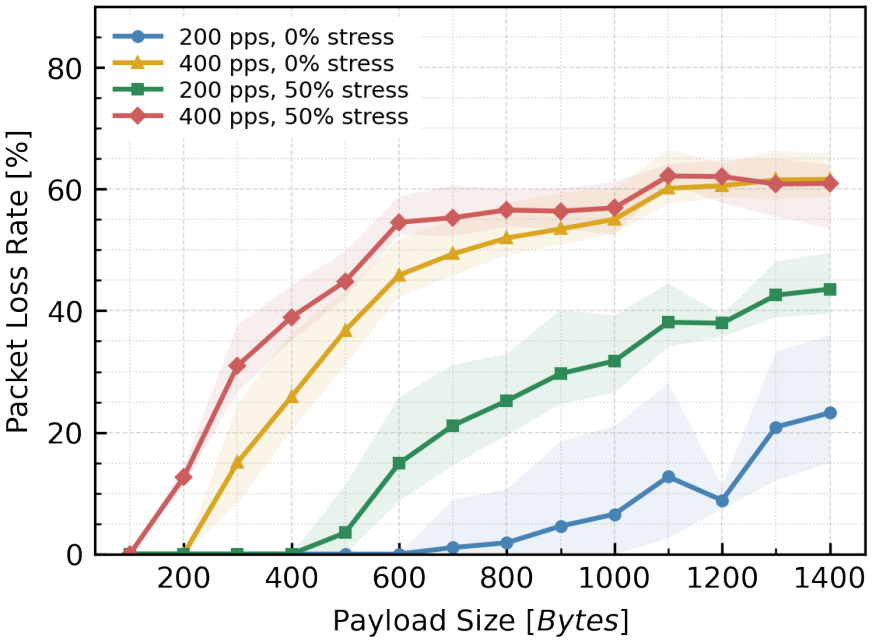


(a) Jitter *(AT)*

Figure 4.5: Jitter and packet loss rates: 100th percentile values under several conditions.



(b) Jitter *(Enc)*



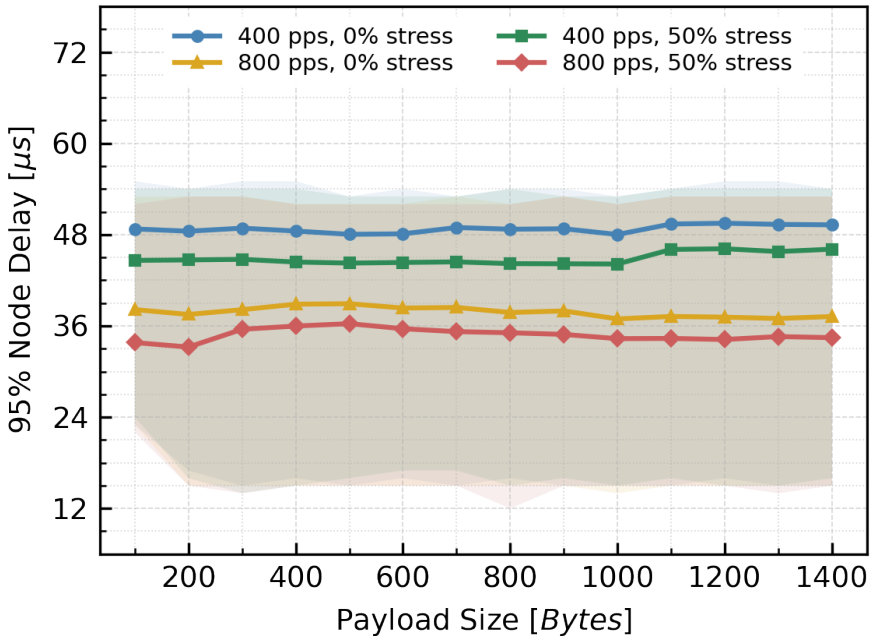
(c) Packet loss rate *(Enc)*

Figure 4.5: Jitter and packet loss rates: 100th percentile values under several conditions.

4.5.4 Node Delay

Figure 4.6 shows the tendency of the node delay variation against several conditions, and Fig. 4.7 shows the likelihood of occurrence as empirical CDF.

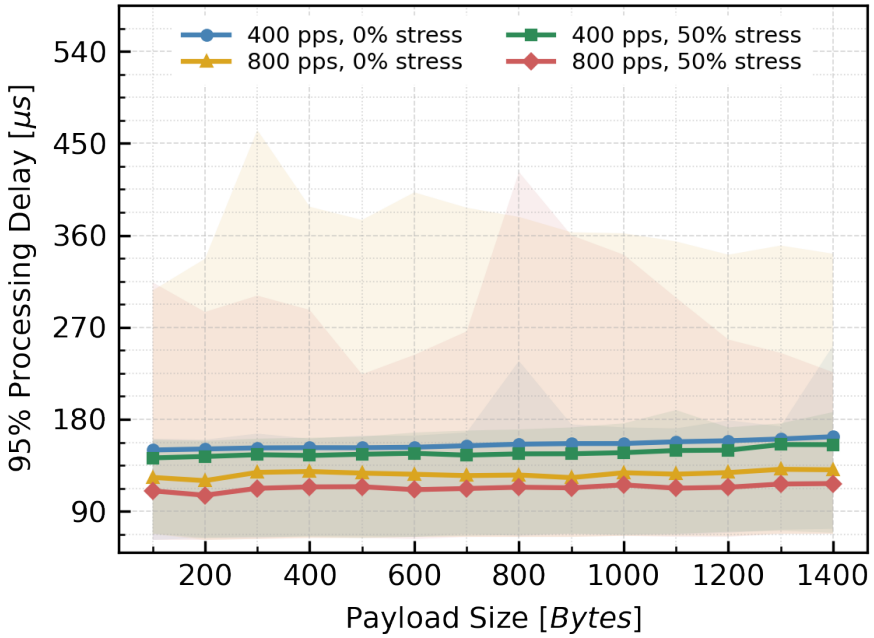
According to these figures, in the cases of *OLSR* and *AT*, the delay was nearly constant irrespective of pps and *stress*. There was a correlation between the variation and pps in *OLSR*, while in *AT* there was not; this suggests that the application-level packet forwarding is less stable than kernel routing from the perspective of node delay. In the case of *Enc*, the *processing delay* increased to the millisecond order and increased approximately linearly with respect to the payload size, and the delay variance became large overall. In addition, the graph tended to be smoothed as the pps increased; this arises from the fact that packet encryption takes up more CPU time, which makes the influence of other processes less conspicuous.



(a) Node delay *(OLSR)*: Only in-kernel delays

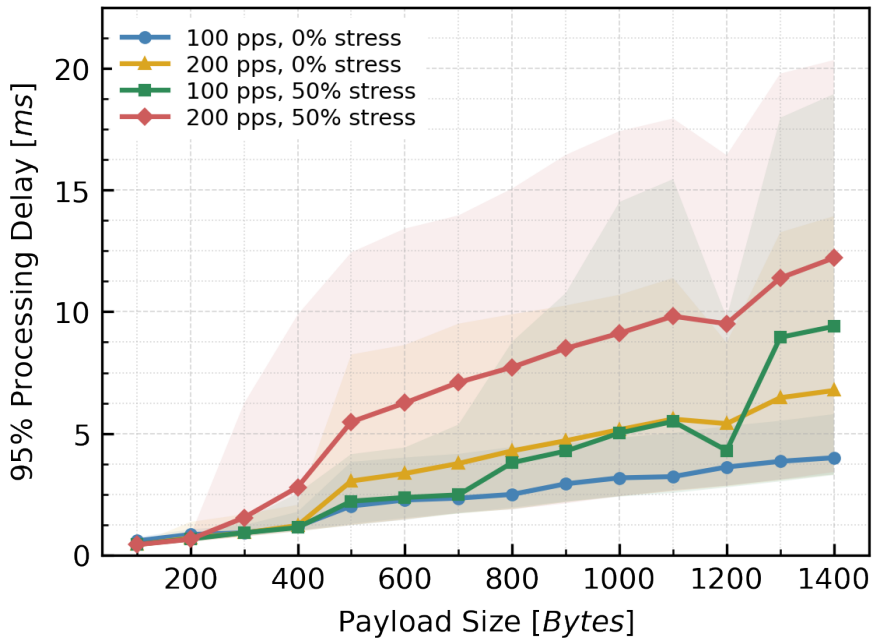
Figure 4.6: Node and *processing delays*: 95th percentile delays under the several conditions.

It appears that the higher the pps, the lower the average delay (Fig. 4.6a and 4.6b), and the delay variance decreases around 1200 bytes (Fig. 4.6c), but the causes of these remain unknown, and further investigation is required. One thing is certain: on the Raspberry Pi, pulling the packets up to the application through the network stack results in a delay of more than 100 microseconds.



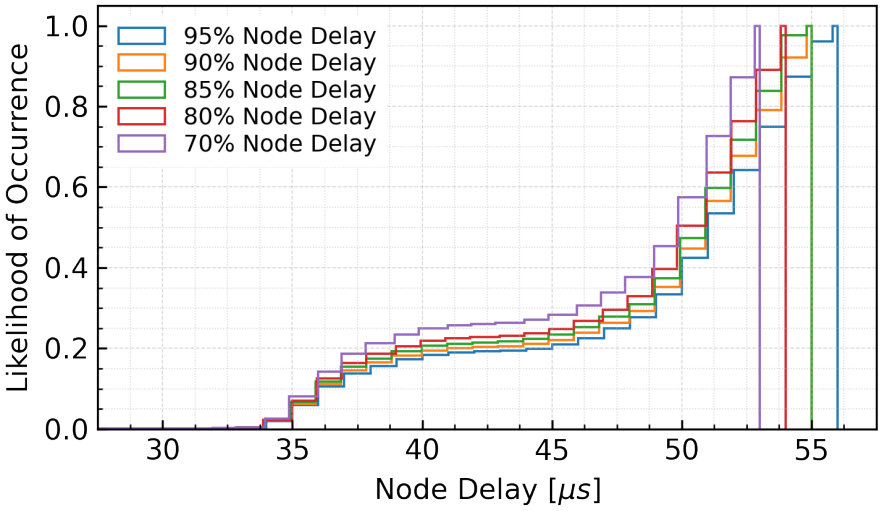
(b) ***Processing delay (AT)***

Figure 4.6: Node and *processing delays*: 95th percentile delays under the several conditions.



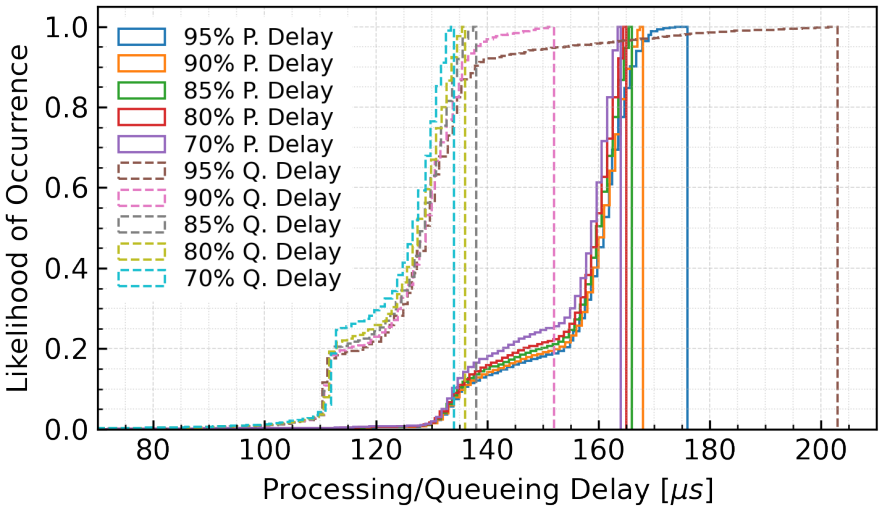
(c) *Processing delay (Enc)*

Figure 4.6: Node and *processing delays*: 95th percentile delays under the several conditions.

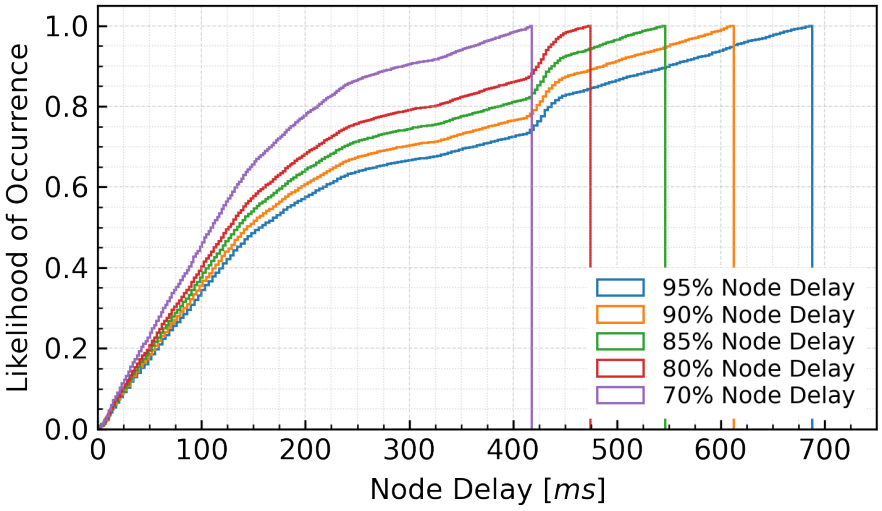


(a) Node delay *(OLSR)*: Only in-kernel delays

Figure 4.7: ECDF: Several percentile node delays under 1000 bytes, 200 pps and 0% *stress*.



(b) ***Processing and queueing delays (AT)***



(c) Node delay ***(Enc)***

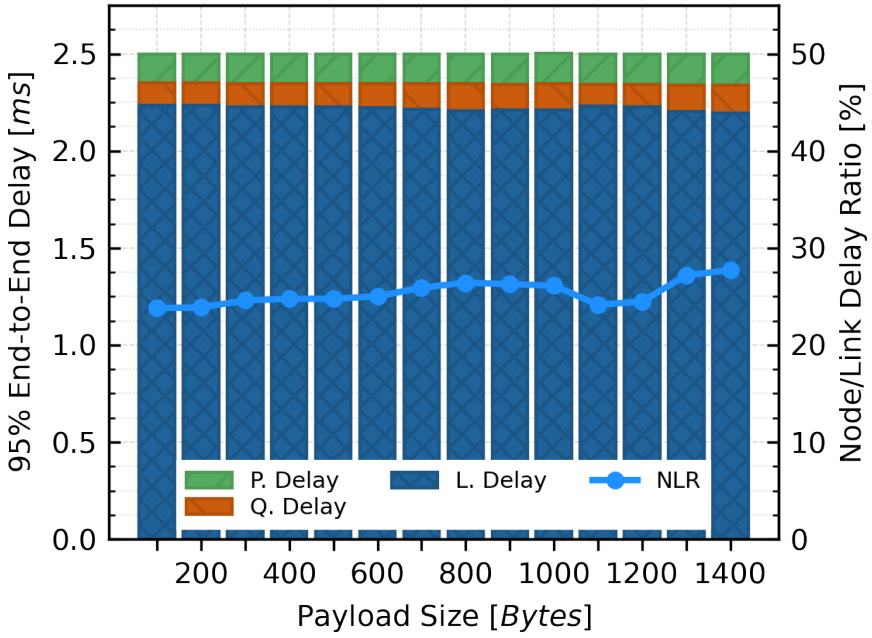
Figure 4.7: ECDF: Several percentile node delays under 1000 bytes, 200 pps and 0% *stress*.

4.5.5 Link Delay vs. Node Delay

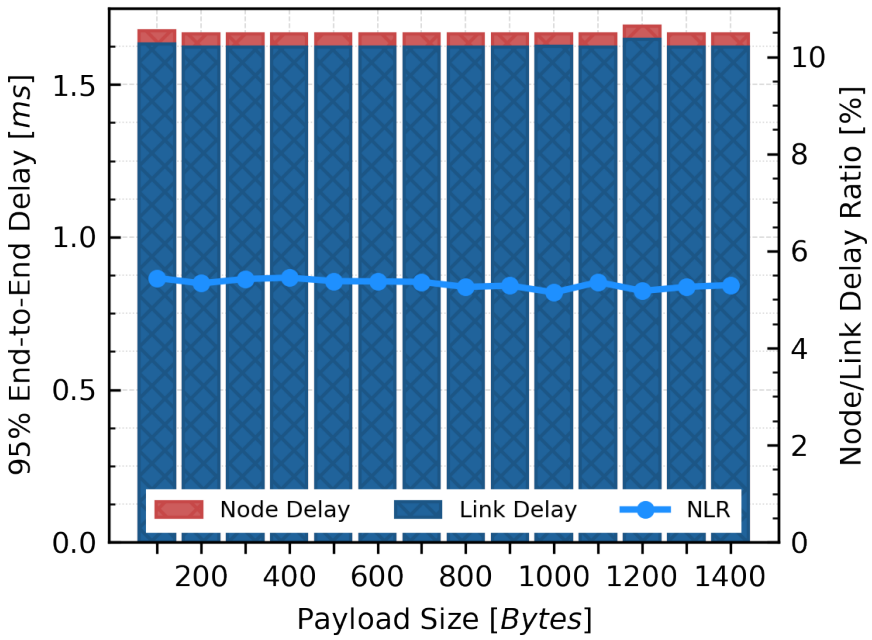
Figure 4.8 shows the breakdown of the end-to-end delay and also describes the node delay link delay ratio (NLR). As in Fig. 4.2, for this experimental environment, the end-to-end delay included two link delays, and the link delay shown in Fig. 4.8 is the sum of them. The link delay was calculated from the effective throughput reported in iperf. As iperf does not support *pps* as its option, the author achieved it by adjusting the amount of transmitted traffic, as

$$bandwidth = pps \times size \times 8. \tag{4.1}$$

The results showed that, in the cases of *OLSR* and *AT*, the NLR was almost constant with respect to the payload size, while in *Enc*, it showed an approximately linear increase. The NLR was less than 5% in *OLSR*, while in *AT*, it was around 30%, which cannot be considered negligible. Furthermore, node delay was greater than link delay when the payload size was over 1200 bytes in *Enc*.

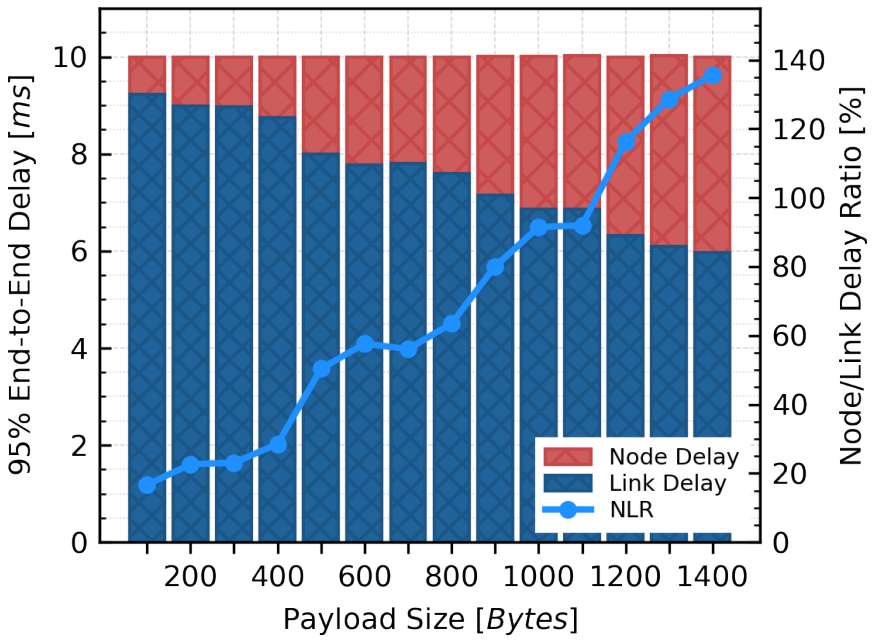


(a) E2E delay *(OLSR)*: 400 pps, 0% *stress*



(b) E2E delay *(AT)*: 400 pps, 0% *stress*

Figure 4.8: End-to-end delay: 95th percentile end-to-end delays including node and link delays.



(c) E2E delay *(Enc)*: 100 pps, 0% *stress*

Figure 4.8: End-to-end delay: 95th percentile end-to-end delays including node and link delays.

4.6 Conclusion

This work has designed and conducted an experiment to measure the software processing delay caused by packets relaying. The experimental environment is based on an OLSR ad hoc network composed of Raspberry Pi Zero Ws. The results were qualitatively explainable, and suggested that, in low-energy ad hoc networks, there are some situations where the processing delay cannot be ignored.

- The relaying delay of kernel routing is usually negligible, but when it is handled by application, the delay can be more than ten times greater, however simple the task is.

- If an application performs CPU-intensive tasks such as encryption or full translation of protocol stacks, the delay increases according to the linear model and can be greater than the link's transmission delay.

For this reason, node-to-node load balancing considering the CPU performance or amount of passing traffic could be extremely useful for achieving delay-guaranteed routing in ad hoc networks. Particularly in heterogeneous ad hoc networks (HANETs), where each node's hardware specs are different from each other, the accuracy of passing node selection would have a significant impact on the end-to-end delay.

No noise countermeasures were taken in this experiment, further studies will involve similar measurements in an anechoic chamber to reduce the noise from external waves and an investigation of the differences in results.

Chapter 5

Link Stabilizer and Path Optimizer

5.1 Introduction

This work proposes a gateway mobility control algorithm determining the optimal position for inter-cluster communication in heterogeneous drone swarms, where only gateway drones are controllable and relocatable, from the aspect of end-to-end communication delay. Precisely, the gateway moves to create a shortcut in the network when a further reduction of communication delay is expected. A conceptual protocol design is also described to implement the algorithm in an autonomous distributed environment.

In the following, the discussion is proceeded with generalized model, but this study can be applied to various real-world scenarios. For better understanding, a wide-area field monitoring system is picked up as an example (Fig. 1.1) and use it for parameter settings of the computer simulation described later:

- Camera-equipped drones are widely deployed to the monitoring area and are on standby at high altitudes by hovering.
- There are two different radio protocols, old and new, that are not compatible with each other.
- Establish interconnection between two clusters by deploying a small number of gateway drones capable of translating two protocols.

- Operators request multi-hop video transmission from any point to any drone.

The drone swarm consists of DIY drones using a drone kit is supposed. These days, as noted in Chapter 2, modules making up a drone – frame, motor, rotor, battery, and flight controller chip – are available as individual parts, allowing people to build a drone with the desired spec at a low cost. Specifically, we can actually build drones with specs like Table 5.1. A drone weighing about 200g equipped with a 4000mAh battery can fly continuously for about 1 hour, depending on the conditions. The flight controller can be installed with not only a dedicated chip but also a tiny single-board computer like Raspberry Pi Zero. Especially in the latter case, installing the Linux kernel allows developers to higher-level programming – embedding advanced functions like autopilot and swarm controller. The simulation experiments presented in Sect. 6.5 were performed assuming drone specifications of Table 5.1.

Table 5.1: Assumed drone hardware specifications.

Flight Controller	Raspberry Pi Zero
Frame Size	360mm
Prop Size	8inch
VTX	RGB camera (FHD quality)
Transceiver	Wi-Fi (2.4GHz) or WiMAX
Positioning	RTK-GPS (ZigBee beaconing)
Battery	3200mAh 4s

Furthermore, in this study, high-quality communication means low-delay communication, and low delay means a lower number of end-to-end path-hops; in short, this study regards the number of relaying hops dominates the communication delay. It has been experimentally demonstrated that in an ad hoc network consisting of low-power processor-equipped nodes, the relay processing delay is up to 140% of the transmission delay occurred in a stable wireless link – a score of IEEE 802.11n 2.4GHz, RSSI over

-30dBm, RNSI under -70dBm, and all nodes are in the line-of-sight distance. Besides, in this scenario, all drones fly and hover at a high altitude enough to be considered free of ground objects' effects so that an ideal radio environment and nearly static network topology can be assumed. Therefore, based on the above experimental facts, there are sufficient reasons for converting the communication delay regarding the number of path hops.

However, with a view to adapting the proposed method to networks with more intense mobility, consideration of dynamic metrics is an important future work. Subsequent studies will consider these simplifications and address more practically complex problems.

5.2 Autonomous Gateway Mobility Control

This section proposes a gateway mobility control algorithm to improve communication quality in heterogeneous drone swarms.

5.2.1 Algorithm Overview

The gateway is described as a two-state machine – the idle state and the in-service state (Fig. 5.1). The former is the state where no passing flows exist, and the latter is where one or more flows are relaying by gateway.

The proposed algorithm is composed of two sub algorithms – *Link Stabilizer* and *Path Optimizer*. *Link Stabilizer* computes fine-grained locations to maintain the current neighbor links as stable as possible, and *Path Optimizer* computes coarse-grained locations to establish a new connection and create a shortcut. *Link Stabilizer* works both in the idle state and the in-service state, while *Path Optimizer* works only when the transition from the idle state to the in-service state.

- ***Link Stabilizer*:** Always working at regular intervals. Move slowly to maintain the neighbor connection.

- ***Path Optimizer*:** Works if necessary when new communication occurred. Move fast to a remote location to make a shortcut.

Path Optimizer does not work in the in-service state; that is, gateways currently relaying one or more flows do not make a gigantic movement that would cause the loss of neighbor connection. Thus the gateway to create a shortcut for newly occurred flow is allocated from idle gateways – if all gateways are in-service state, use the shortest path with the current gateway placement regardless of its end-to-end delay.

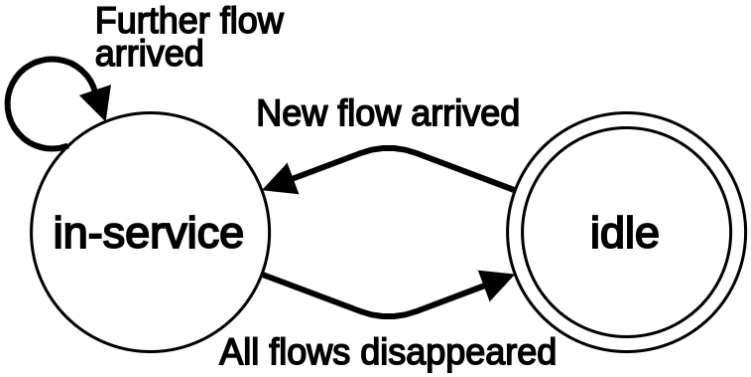


Figure 5.1: A state diagram of gateway function

5.2.2 Link Stabilizer

The gateway availability is one of the communication requirements of the heterogeneous drone swarms. Because every inter-cluster communication goes gateway, ideally, any nodes must have the path reachability to at least one gateway. *Link Stabilizer* (Algorithm 1) realized this by the two-stage control: the micro-connectivity control (Algorithm 2) and the micro-mobility control (Algorithm 3).

- **Micro-connectivity Control** is to create clusters from neighbor nodes based on the relative velocity.

- **Micro-mobility Control** is to determine the gateway velocity vector based on the kinetic model.

Algorithm 1 Link Stabilizer

Input: Control target GW g, physical neighbor set V_n, and GW set V_g
Output: Acceleration vector of GW g
1: **function** STABILIZER(g, V_n, V_g)
2: $Q \leftarrow$ **MicroConnectivityControl**(g, V_n)
3: **return MicroMobilityControl**(g, Q, V_g)

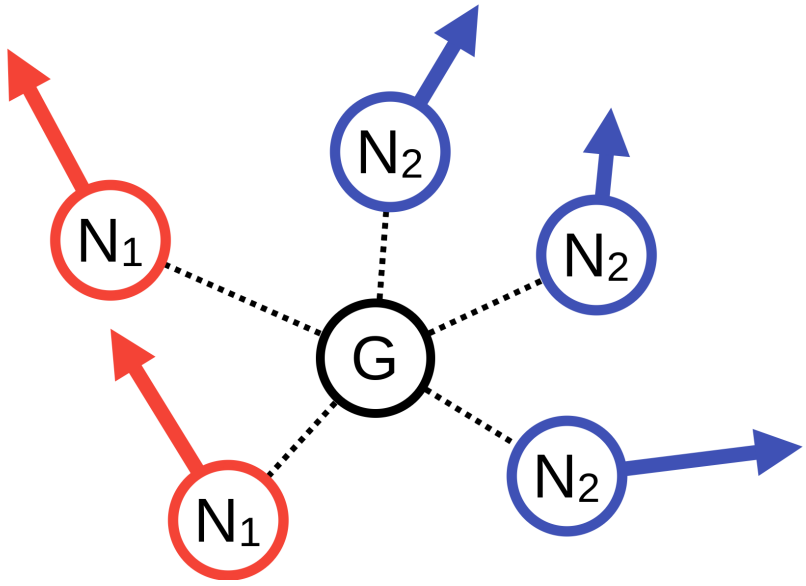
Micro-connectivity Control

Since all nodes in the network move independently of each other, the gateways must also move continuously as their neighbors move. Algorithm 2 constructs logical neighbors from physical neighbors and maintains the link only to the logical neighbors to stabilize the gateway connectivity, as shown in Fig. 5.2. The overall behavior of the algorithm as expressed by the pseudocode is as follows:

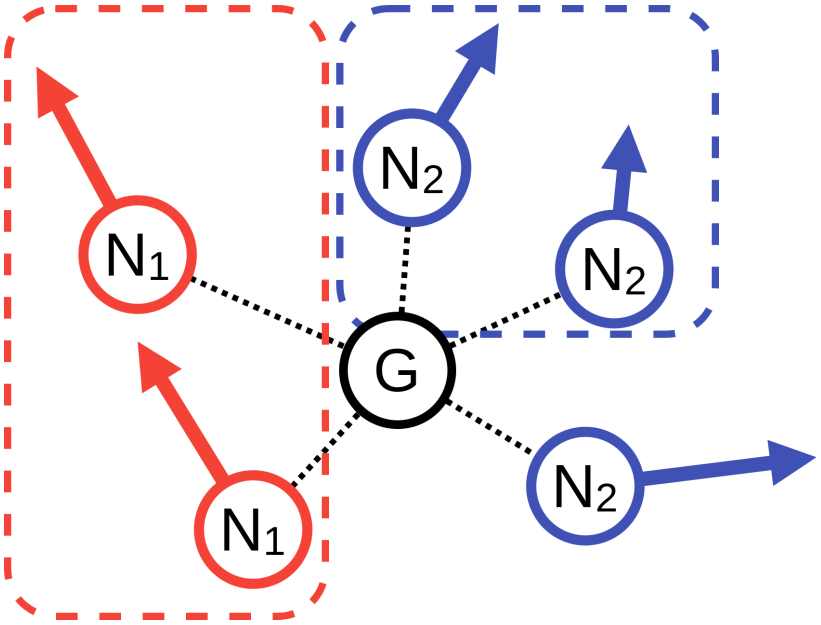
1. **Initialize:** When the gateway is in the in-service state, the algorithm constructs the initial logical neighbor set with all the physical neighbors used by the relaying communications. Otherwise, the algorithm calculates the relative velocity of any two nodes in the physical neighbors and adopts the node pair with the lowest relative speed as the initial logical neighbor set.

2. **Expand:** Calculate the average speed of the logical neighbor set and append all nodes whose relative speed is less than the threshold to the set. Repeat this operation by increasing the threshold until the set compose of heterogeneous clusters.

If the physical neighbors consist of only a single cluster, use the physical neighbors as the logical neighbors because it is impossible to create heterogeneous logical neighbors. The same clustering approach is also used in [88], though their study assumed the homogeneous ad hoc network.

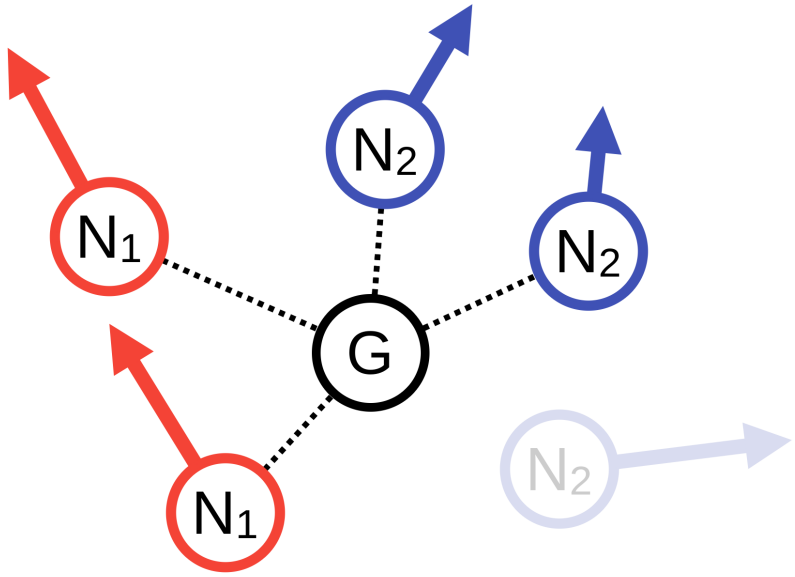


(a) Before: gateway is connecting to all physical neighbors.



(b) Clustering: gateway forms sub-clusters with nodes which are close in relative velocity.

Figure 5.2: Micro-connectivity control: A gateway surrounded by two clusters, cluster red and blue, determines the links to maintain and discard.



(c) After: gateway is connecting to only logical neighbors.

Figure 5.2: Micro-connectivity control: A gateway surrounded by two clusters, cluster red and blue, determines the links to maintain and discard.

Algorithm 2 Neighbor Clustering

Input: Control target GW g and physical neighbor set V_n
Output: Logical neighbor set Q

function MicroConnectivityControl(g, V_n)
  if $|V_n| \leq 2$ **then**
    return V_n
  $t \leftarrow 1$
  while true do
    $Q \leftarrow \{\}$
    for all $u, v \in$ Combination(V_n) **do**
      if $||$ u.velocity $-$ v.velocity $|| \leq t$ **then**
        Q.push(u, v)
    if is_hetero(V_n) **and** $|Q| \geq 1$ **or** is_hetero(Q) **then**
      return Q
    $t \leftarrow t + 1$

Micro-mobility Control

In order to keep the connection to the logical neighbors computed previously, gateway needs to move with the appropriate velocity. Algorithm 3 calculates the optimal velocity based on the kinetic model. Specifically, it considers the neighbor nodes to exert a force on the gateway, represents the network as a potential field, and calculates the velocity vector from the potential energy gradient (Fig. 5.3). As a side note, although their research field is different, Aida et al. have also represented the network as a kinetic model to analyze a social network [89].

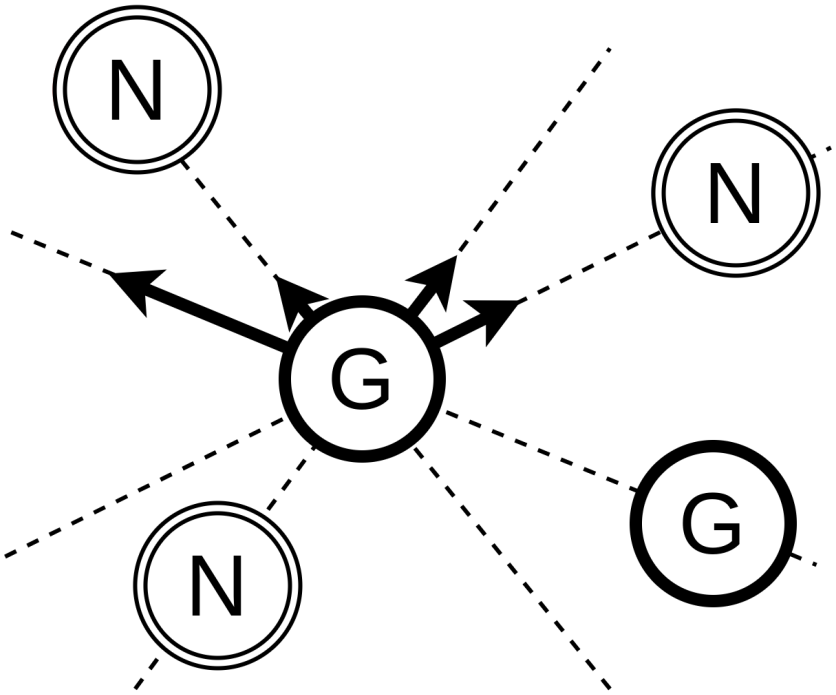
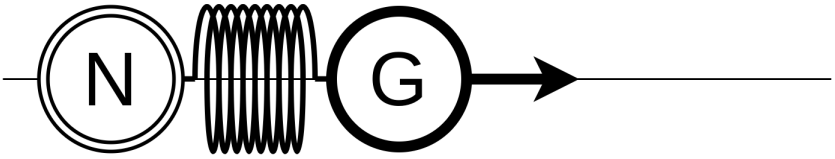
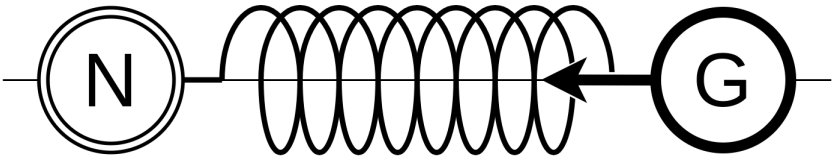


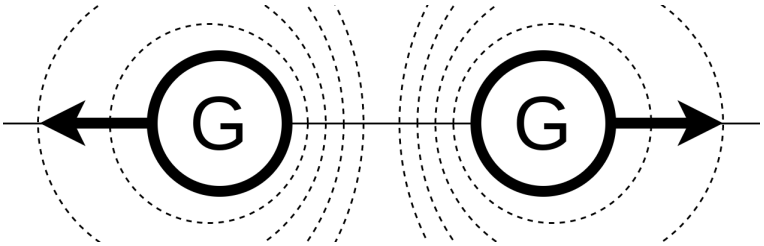
Figure 5.3: The logical neighbors exert a force on gateway.



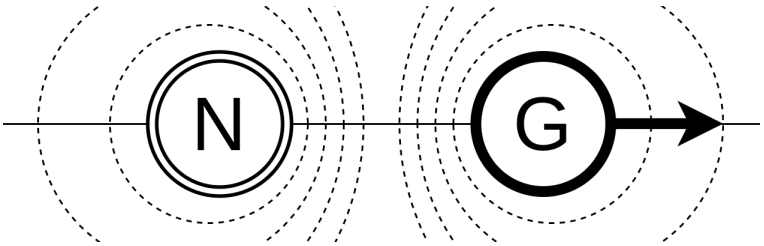
(a) Repulsive force for collision avoidance.



(b) Attractive force for link disruption avoidance.



(c) Repulsive force for distributing gateways widely in the field.



(d) Repulsive force for collision avoidance.

Figure 5.4: Types of forces acting between gateway and other nodes: G means gateway and N means non-gateway.

Algorithm 3 Gateway Velocity Calculation

Input: Control target GW g, logical neighbor set Q and GW set V_g
Output: Acceleration vector of GW g

1: **function** MICROMOBILITYCONTROL(g, Q, V_g)
2: $F \leftarrow \mathbf{0}$
3: **for all** $u \in Q \setminus V_g$ **do**
4: $F \leftarrow F + k_a\ (1 - r||\ g.\text{pos} - u.\text{pos}\ ||^{-1})\ (u.\text{pos} - g.\text{pos})$
5: $F \leftarrow F + k_{r_1}\ ||\ g.\text{pos} - u.\text{pos}\ ||^{-3}\ (g.\text{pos} - u.\text{pos})$
6: **for all** $u \in V_g$ **do**
7: $F \leftarrow F + k_{r_2}\ ||\ g.\text{pos} - u.\text{pos}\ ||^{-3}\ (g.\text{pos} - u.\text{pos})$
8: **return** $F/1$

First, the gateway is assumed to be subjected to forces, which linearly depend on the distance from non-gateways of logical neighbors. It is represented as an elastic force that obeys Hooke's law defined in Eq. 5.1 (k_s is the spring constant and x is the spring elongation from its natural length), as shown in Figs. 5.4a-5.4b. In other words, this force acts both attractive and repulsive to keep the distance between gateways and non-gateways constant

$$F_s = -k_s x. \tag{5.1}$$

In addition, all logical neighbors are considered to exert nonlinear forces on gateway depending on the inverse-square of the distance. It is expressed as the Coulomb force, as shown in Figs. 5.4c-5.4d and Eq. 5.2 (k is the Coulomb constant, q_1 and q_2 are the amounts of charge, and r is the distance between charged particles). Since the Coulomb force diverges at zero distance, this helps to avoid collisions with neighbors. Note that the higher amount of charge is set for gateway-to-gateway interaction than gateway-to-non-gateway interaction because the repulsion between gateways is also expected to prevent multiple gateways from falling into a local hollow of the potential field.

$$F_c = -k_c \frac{q_1 q_2}{r^2}. \tag{5.2}$$

In summary, the forces and accelerations acting on gateway are shown in Eq. 5.3, where k_a is an attractive coefficient, k_{r_1} and k_{r_2}, which satisfy $k_{r_2} \geq k_{r_1}$, are repulsive coefficients, u is the nodes' position vectors, V_g is a gateway set, and V_n is a non-gateway set. r is a parameter with the dimension of distance – the smaller r is, the less the link disruption risk, but on the other hand, the more the node collision risk. So r should be picked under the condition of two factors: the communication radius and the intensity of node mobility, though, typically, it is 50% of the radius.

$$\begin{aligned} \dot{v_g} = F_g &= \sum_{i \in V_n} k_a(1 - r||u_i - u_g||^{-1})(u_i - u_g) \\ &+ \sum_{j \in V_n} k_{r_1}||u_j - u_g||^{-3}(u_j - u_g) \\ &+ \sum_{j \in V_g} k_{r_2}||u_j - u_g||^{-3}(u_j - u_g). \end{aligned} \tag{5.3}$$

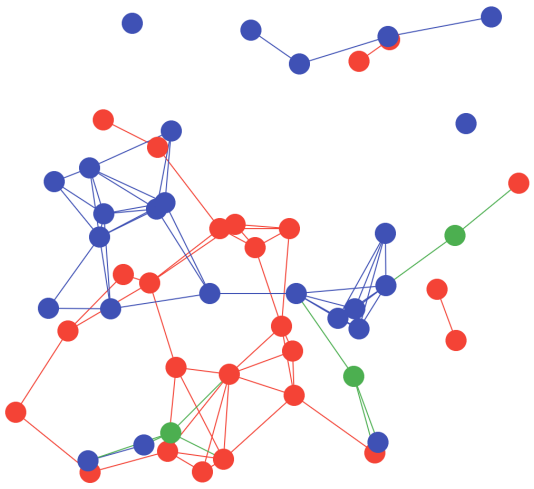
Example

Figure 5.5 illustrates the actual gateway movement under the control of *Link Stabilizer* – 4 gateways are deployed into an environment of 50 nodes, 25 nodes each of red and blue clusters. By observing the gateway link status of each time, we can confirm that the gateways appropriately select logical neighbors and keep an approximately equal distance from them.

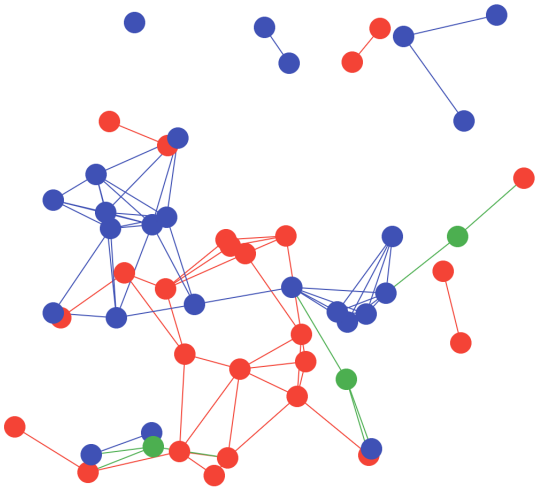
As noted in the previous section, each node is assumed to be able to exchange its location and velocity with its neighbors. Every gateway collects the neighbor information and executes *Link Stabilizer* at regular intervals for autonomous mobility management.

5.2.3 Path Optimizer

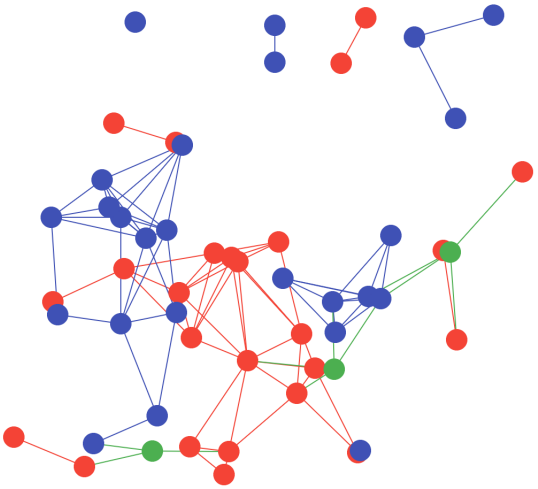
The gateway availability is improved by *Link Stabilizer*, so the RREQ, a signaling packet meaning communication request, now reaches gateway more stably and with higher probability. The next consideration point is a dynamic gateway relocation for



(a) Snapshot of network at time $t = 0$



(b) Snapshot of network at time $t = 2.5$



(c) Snapshot of network at time $t = 5$

Figure 5.5: An actual behavior of gateway movement under the control of *Link Stabilizer*: cluster red and blue are interconnected by the green gateways.

every RREQ arrivals – Algorithm 4 aims to reduce the communication delay by moving gateway to create a shortcut inside the network for the target flow.

Although a general graph shortcut problem is known to be an NP-hard problem, i.e., **the problem of finding the link set** $L \notin E$ **from** $C = \{(s,t)|\forall gateways, t \in V, \exists d > 0, ||s-t|| < d\}$ **such that the shortest paths between any two nodes are minimized with the given undirected graph** $G = (V, E)$, this article will deal with the special case of that, **the problem of minimizing the shortest path between a specific node pair by adding only one link**.

Shortcut Searching

The behavior of the algorithm meant by the pseudocode is as follows. It searches sequentially if a shorter path between ends than the current one can be constructed by adding a new node to the graph.

1. Calculate the shortest path length d_0 between the source and the destination under the current gateway placement.

2. Split G into G_R and G_B across the gateway – G_R and G_B are equivalent to heterogeneous clusters.

3. Run gateway-based Dijkstra's algorithm on G_R and G_B respectively. Calculate the shortest path hop between gateway and every node. Generate new graphs $G_R{}'$ and $G_B{}'$ that added weights for each link of G_R and G_B based on the hop distances. The ***weighting function*** is described later.

4. Calculate the shortest path hop ***1)*** from the source to any other node at G_R, and ***1)*** from the destination to any other node at G_B using Dijkstra's algorithm, and let the result as ***hop's map***.

5. Do the same calculation for $G_R{}'$ and $G_B{}'$, and let the result as ***cost's map***.

6. On the condition that the geographical distance between the node pair is within twice the communication radius, select node u and v from $G_R{}'$ and $G_B{}'$, respectively. Calculate the shortest path hop d between the source and the destination using the ***hop's map***, and if $d \leq d_0 + 1$ is satisfied, then calculate the lowest cost between these nodes using the ***cost's map***.

7. Adopt the node pair (u, v) with the lowest cost as a newly created shortcut link.

The ***weighting function*** is composed of Eq. 5.4 and Eq. 5.5. In order to set a larger link cost as the distance from a gateway increases, it penalizes a link (u, v) with a nonlinear cost $W(m, n, h)$ – m, n are the shortest hop distances from a gateway to nodes u, v respectively, and h is the hop limit. With this mechanism, when multiple gateways are in the idle state, the gateway with the lowest travel distance is adopted for the shortcut formation. Although details are described in Sect. 5.3, the shortest path length calculated with the ***cost's map*** in the seventh item of the above bullets, which takes the positive real numbers unlike the hop length, is the key to gateway selection.

The performance evaluation explained in Sect. 5.5 set h to 5, but this value should be adjusted according to the average shortest path length of the target network, e.g., the communication radius of a wireless protocol, the number of nodes, and the number of gateways.

$$P(x, h) = \{1 - h^{-1}(x - 1)\}^{-1}. \tag{5.4}$$

$$W(m, n, h) = \begin{cases} P(m, h) \cdot P(n, h) & (1 \leq m, n \leq h) \\ \infty & (otherwise) \end{cases} \tag{5.5}$$

Algorithm 4 Shortcut Searching

Input: Current network $G = (V, E)$, communication radius R, source cnode s, destionation node t, transit GW g and GW set V_g
Output: Best shortcut link (u, v)

```
1: function FindShortcut(G, R, s, t, g)
2:     G_0 ← G (V_g g)
3:     d_0 ← astar(G_0, s, t, euclid(G_0))
4:     G_R, G_B ← split(G_0, g)
5:     m_R, m_B ← dijkstra(G_R, g), dijkstra(G_B, g)
6:     M_R, M_B ← dijkstra(G_R, s), dijkstra(G_B, d)
7:     M_R' ← dijkstra(G_R, s, weights(m_R))
8:     M_B' ← dijkstra(G_B, d, weights(m_B))
9:     P, Q ← {}, {}
10:    for all (u, v) ∈ V × V \ E do
11:        if M_R(u) + M_B(v) ≤ d_0 then
12:            P ← (u, v)
13:    for all (u, v) ∈ P do
14:        if ||u − v|| ≤ 2R then
15:            Q ← {M_R'(u) + M_B'(v), (u, v)}
16:    return Minimum(Q)
```

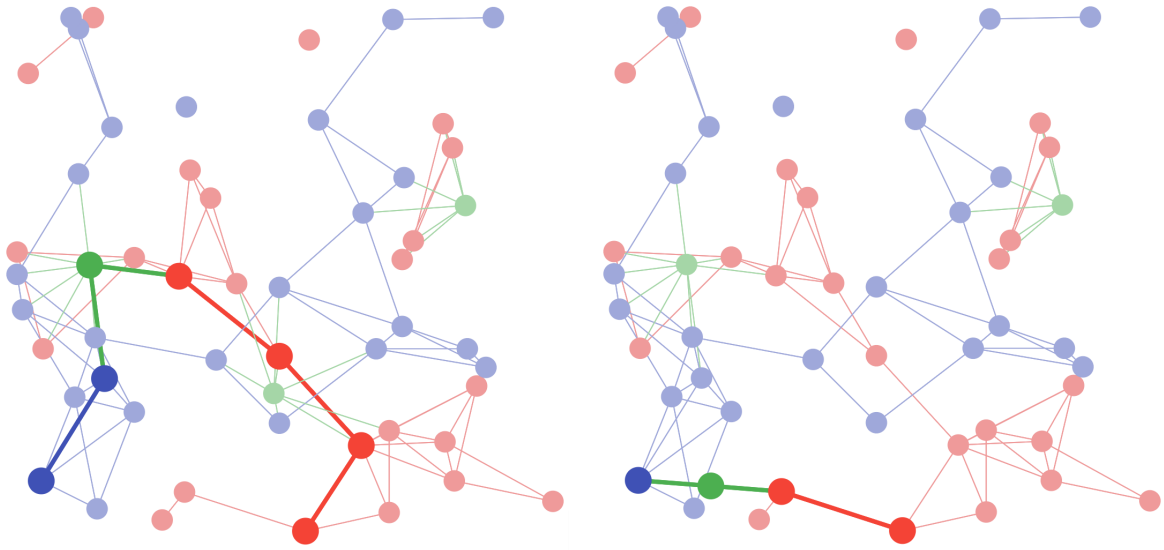
Computational Complexity

The computational complexity is dominated by the shortest path searching of Dijkstra's algorithm. When the network is split into two parts across the gateway, as obvious from the pseudocode (Algorithm 4), the total cost of Dijkstra's algorithm can be $2\{O\Big(V_1 + E_1\Big)+O\Big(V_2+E_2\Big)\}+O\Big(k(V_1+E_1)\Big)+O\Big(k(V_2+E_2)\Big)$. k is the total number of possible link costs and $k \approx 2E - 1$ so eventually the calculation cost becomes $O\Big((2E + 1)(V + E)\Big)$. It also becomes $O(E^2)$ under $E \gg V$.

Typically, for a random geometric graph with about 200 nodes, the processing delay does not make a remarkable difference.

Example

Finally, the actual behavior of *Path Optimizer* is presented as Fig. 5.6. In the initial placement, an end-to-end delay of 6 hops was expected (Fig. 5.6a), but after *Path Optimizer* running, a shortcut is formed, and the communication delay is reduced by 3 hops (Fig. 5.6b). We can also confirm that the nearest gateway, i.e., the gateway of lowest mobility cost, was selected to create a shortcut.



(a) Before the optmization: path length is 6 hops

(b) After the optmization: path length is 3 hops

Figure 5.6: An actual behavior of gateway movement under the control of *Path Optimizer*: one of the gateways has been selected and relocated for shortcut formation.

5.3 Distributed Gateway Selection Mechanism

This section, as an additional discussion about the previous section, illustrates a conceptual protocol design and show that the proposed algorithm can be deployed on actual heterogeneous drone swarms in a distributed manner.

5.3.1 Protocol Overview

As explained previously, the proposed algorithm requires knowledge of the whole network topology and geographic locations of all nodes to work. This paper adopts an approach to infer the topology from all nodes' geolocations instead of directly gathering link status information.

Although it sounds slightly coarse and inaccurate, in some simple situations where the radio environment is ideal, and the nodes are not moving too much, it is possible to highly estimate the network topology only based on the physical distances between nodes. Moreover, there are some advantages to adopting this approach – it is easy to implement and runs fast and stable because there is no complex mechanism.

On the other hand, for example, in an unmanned plant monitoring system for intruder detection, all nodes may not be within the line-of-sight due to the obstacles, and radio interference cannot be ignored; thus, it is difficult to infer the topology from geolocation information, and a more complex and delicate mechanism is required, e.g., by evaluating the quality of each link with RSSI or packet loss ratio.

As this study targets generic HANETs, the discussion should not be limited to a specific situation like wildlife monitoring. However, assuming a radio environment with rapidly changing link quality would complicate the problem and go beyond the scope of the initial study - this is one of the topics for future work.

5.3.2 Extended Hybrid Wireless Mesh Protocol

It is reasonable to assume that IEEE 802.11s is used to construct the ad hoc network for the interconnection between two Wi-Fi standards

IEEE 802.11s is applied to a modified MAC layer to the existing Wi-Fi standards and provides various functions necessary to configure a wireless mesh network without requiring PHY layer modification, e.g., use 11s as MAC and 11ac as PHY. IEEE 802.11s must support HWMP, which is the abbreviation of hybrid wireless mesh protocol [90]. The following shows that the proposed algorithm can be fully implemented in an existing

autonomous distributed system by introducing an extended HWMP.

The basic actions of the extended protocol are the following two points. Note that since the protocol proposal is not the focus of this paper, only a brief conceptual design is presented.

- Grasp geolocations of all nodes proactively.
- Grasp occurred communications reactively.

Extend RREQ/RREP Messages

HWMP supports two routing schemes: reactive routing based on AODV and proactive routing based on the tree-based routing algorithm, and besides, it can activate both schemes simultaneously.

When it acts with the reactive scheme, end-to-end path searching is realized by ***1)*** source node sends RREQ (Route Request) as flooding to the destination node whenever a communication request occurs, and then ***2)*** the destination received RREQ replies RREP (Route Reply) as a unicast to establish the reverse path. In the proactive scheme, ***1)*** gateways floods RANN (Root Announcement) periodically, and ***2)*** every node that received RANN runs the reactive action to the source gateway; as a result, all nodes keep and refresh the path to the gateways regularly.

In both cases, RREQs and RREPs flow through the network. The author make use of this point and extend these message formats as below.

- The sender node stores its own geographic location information in the Originator Geographic Location Field added to the RREQ.
- If necessary, gateway stores its own metric in the Independent Metric Field added to the RREP.

This modification allows the gateway to collect the network topology and nodes' geolocations with a periodic refresh. Detailed usage of RREP is described in the next section.

Introduce PREQ/PREP Messages

The two new message types are defined - PREQ (Proxy Request) and PREP (Proxy Reply) and then design the overall behavior of the protocol as summarized in Fig. 5.7.

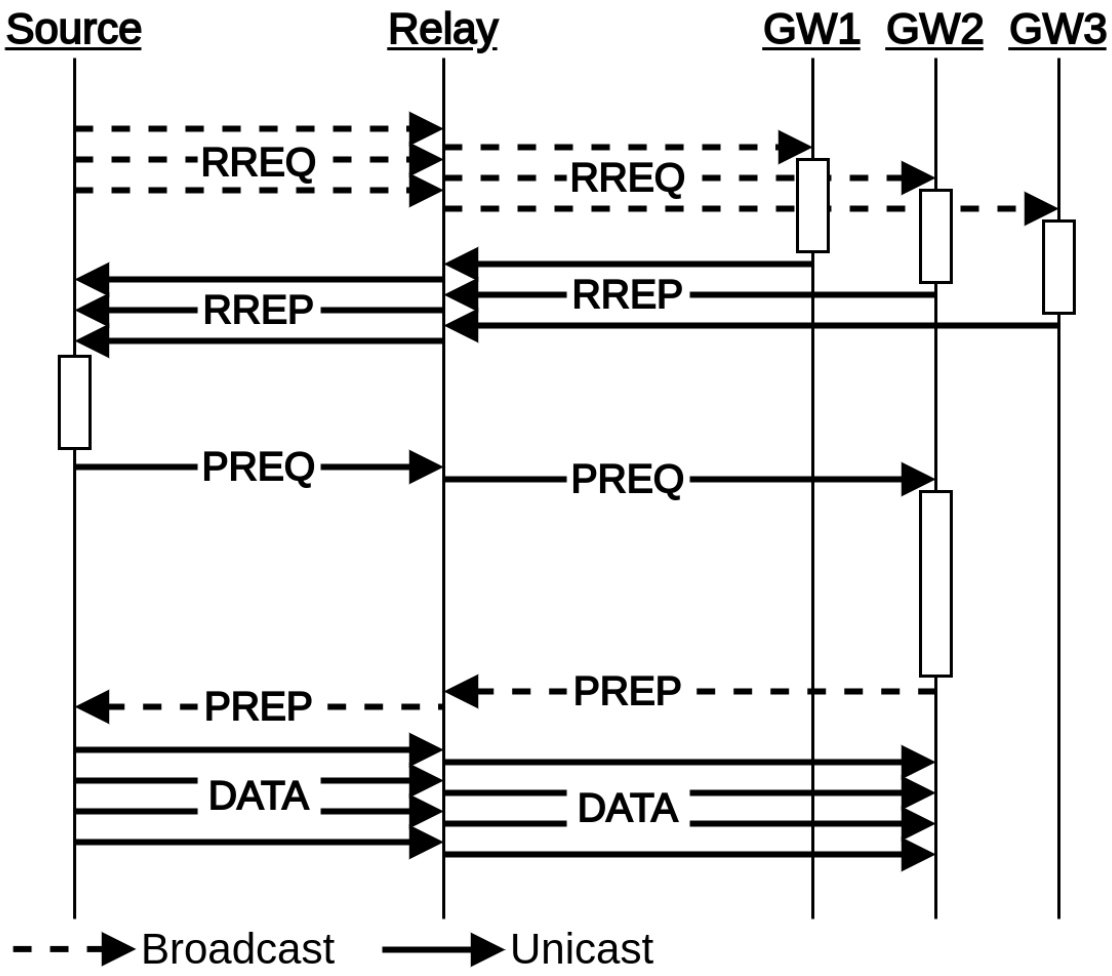


Figure 5.7: Data transmission sequence in extended HWMP.

1. The source node broadcasts the RREQ.

2. GW1 to GW3 grasp the newly occurring flow by receiving the RREQs. Each idle gateway searches the shortest path from the source to the destination via itself, assuming the shortcut formation, evaluates its cost using the ***cost's map*** introduced in the previous section, and unicasts the RREP storing the cost in the Independent Metric Field. Note that in-service gateways return the path hop length via itself as the Independent Metric without considering the shortcut formation.

3. The source node receives RREPs from multiple gateways and then selects the gateway of the least Independent Metric and unicast a PREQ to the gateway.

4. After receiving the PREQ, the gateway starts to move and create the shortcut and then sends back the PREP when completed. As a side note, since the gateway has moved geographically in the network, the reverse path to the source node established earlier would be lost; therefore, the broadcast is needed here.

5. The source node that received the PREP starts sending data to the other network via the gateway.

Since the moving gateway does not receive the RREQ, the number of idle gateways in the network will decrease during this period. If the sequence fails to complete due to the link loss or other reasons, including the case that no RREP is received from any gateway, the source node re-issues the RREQ, and the whole procedure starts over again.

In this way, autonomous decentralized gateway selection and mobility control mechanisms are implemented in the heterogeneous drone swarms.

5.4 Preparation for Performance Evaluation

Since no studies have dealt with the same model of this study, it is impossible to compete for performance with existing methods. Therefore, the characteristics of the proposed algorithm are analyzed by computer simulation with a wide range of parameter settings. This section explains the mobility and topology models to be evaluated and discusses an extended feature of the algorithm.

Aside from that, the evaluation platform is a network simulator that the author implemented entirely on his own using Julia 1.5.4.

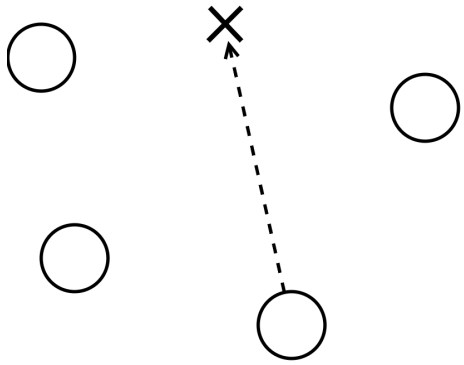
5.4.1 Mobility Models

In order to evaluate the effectiveness of the proposed method in stabilizing connectivity, three different dynamic networks are prepared: random waypoint model, SRCM, and the hovering model.

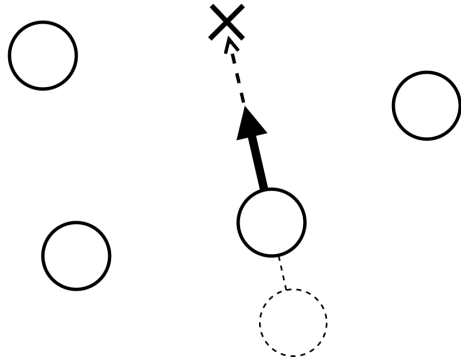
Random Waypoint Model

It is one of the most commonly used mobility models [91]. In this model, node movement has three stages, as shown in Fig. 5.8. First, the node randomly selects its destination in an evaluation field. Then, the node goes to its destination at a randomly chosen constant speed not exceeding some specified maximum speed. Finally, the node pauses for a random time when it arrives at its destination.

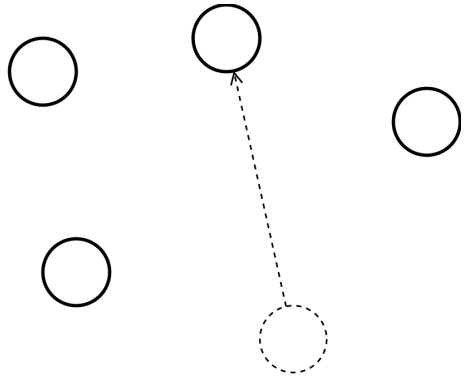
The intensity of mobility can be adjusted by changing the speed range or pausing time range, but in any case, the nodes are scattered over the entire field, and thus it is not suitable for representing the situation where nodes are kept in one point of the field such as hovering. However, for a comprehensive investigation, the experiment adopt this model. Although removed from the evaluation this time due to the intensity of mobility, RPGM (Reference Point Group Mobility) [92] could be used in future evaluations.



(a) Select new destination randomly.



(b) Move toward the destination with randomly chosen speed constantly.



(c) Pause for a random time and go back to (a).

Figure 5.8: Three stages of node movement in the random waypoint model.

SRCM

A model in which nodes are spread over a wide area, such as the random waypoint model, may not be suitable for illustrating the UAV mobility. Wang et al. proposed a new mobility model called semi-random circular movement (SRCM) [93].

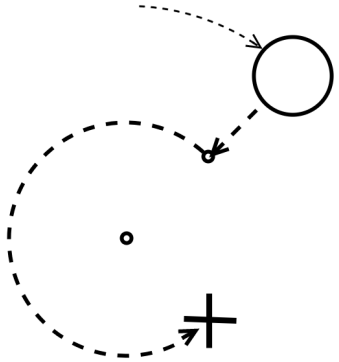
As shown in Fig. 5.9, this model consists of three stages. First, the node randomly selects its distance from a predetermined center position and sets a point on the circumference as its destination. Then, the node goes in a straight line to its center position, and once it reaches the circumference, it moves to the destination at a randomly selected constant rotational speed. Finally, the node pauses for a random time when it arrives at its destination.

In this model, nodes are biased toward 2D disk regions, making it particularly suitable for representing the aerial standby or environmental sensing by UAVs.

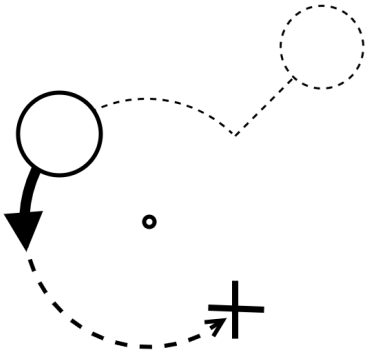
Algorithm 5 Node Movement under the Hovering Model

Input: Node n, wind probability p, wind speed range $[s_1, s_2]$, wind duration t, accelerator a, and calculation period τ
Output: None (update position of given node n)

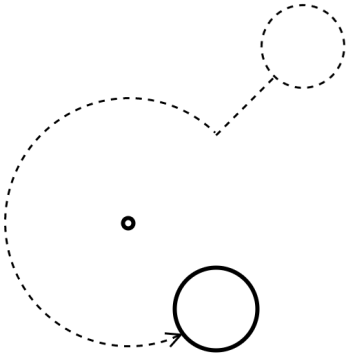
function MOVE(n, p, $[s_1, s_2]$, t, τ)
    $v \leftarrow a \mid\mid n.\text{base} - n.\text{pos} \mid\mid^{-1} (n.\text{base} - n.\text{pos})$
    if rand(Uniform($[0, 1]$)) $\leq p$ **then**
        $L \leftarrow$ rand(Uniform($[s_1, s_2]$))
        $\theta \leftarrow$ rand(Uniform($[0, 2\pi]$)
        $v \leftarrow v + tL(\cos\theta, \sin\theta)$
    $n.\text{pos} \leftarrow n.\text{pos} + v\tau$



(a) Select new destination - radius and angle randomly.



(b) Move toward the destination with randomly chosen rotation speed constantly.



(c) Pause for a random time and go back to (a).

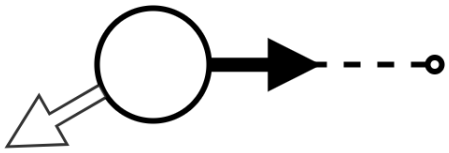
Figure 5.9: Three stages of node movement in the SRCM [93].

Hovering Model

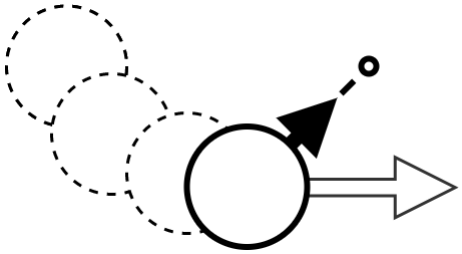
Wang et al. claimed that the SRCM is also the best model to represent the hovering. However, in the SRCM, nodes are always moving in a circular motion, so it is only suitable for aircraft that cannot perfectly stay in the air. The target of this study is a drone swarm consisting of multirotors, so a bit more stationary mobility model is appropriate.

Although the author surveyed various models [94], none of them were satisfactory, and thus the author decided to design own model as shown in Fig. 5.10 – the hovering model. Each node always moves in a straight line to the predetermined locations, but its position is unstable due to a randomly occurred gust of wind (Algorithm 5). The intensity of mobility can be adjusted by choosing the probability of wind blowing, the range of wind speed, and the node's acceleration coefficient. As described in Chapter 3, for this time, since the target situation does not include the 3D node mobility, and the SRCM is also the 2D model, so the author designed the hovering model as a 2D.

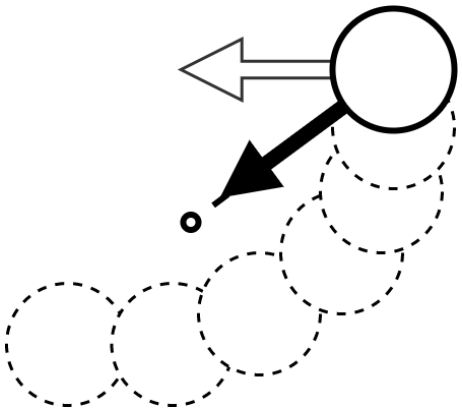
The hovering model represents the situation where a drone hovering at a high altitude is blown away by the wind; hence it is the most appropriate mobility model for the target scenario.



(a) First, the wind is blowing toward eight o'clock.



(b) Then, the wind started blowing towards three o'clock.



(c) And now, it has changed to nine o'clock.

Figure 5.10: Image of node movement in the hovering model: the white arrow represents an external force, such as wind.

5.4.2 Random Geometric Graph

The initial nodes' positions are generated based on the random geometric graph (Fig. 5.11). As mentioned in Chapter 3, two types of clusters managed by independent flight controllers are targetted. For this reason, the graph generator eliminates initial position that does not satisfy the following two conditions:

- Each cluster is an independent connected graph, and there are no isolated nodes exist.
- The distance between any two nodes is larger than the threshold - 10% of the communication radius.

Algorithm 6 Probabilistic Movement Judgement based on Centrality Measures

Input: Lengths of the path d, d', allowable delay m, list of degrees of the centrality c, c', and index of the list i
Output: Whethre the GW should move or not

```
1: function JUDGEMENT(d, d', m, c, c', i)
2:     function EVALSHORTCUTSCORE(d, d', m)
3:         a, b ← m − d, m − d'
4:         if a + b = 0 then
5:             return ∞
6:         return b/a
7:     function EVALCENTRALITYSCORE(c, c', i)
8:         a, b ← c[i] / sum(c), c'[i] / sum(c')
9:         return b/a
10:    N_s ← EvalShortcutScore(d, d', m)
11:    N_c ← EvalCentralityScore(c, c', i)
12:    if (d' ≤ m and d > m) or (d' ≤ d and N_c > 1) then
13:        return true
14:    P_s, P_c ← Sigmoid(N_s), Sigmoid(N_c)
15:    return random(Uniform([0, 1])) ≤ (P_s + P_c)/2
```

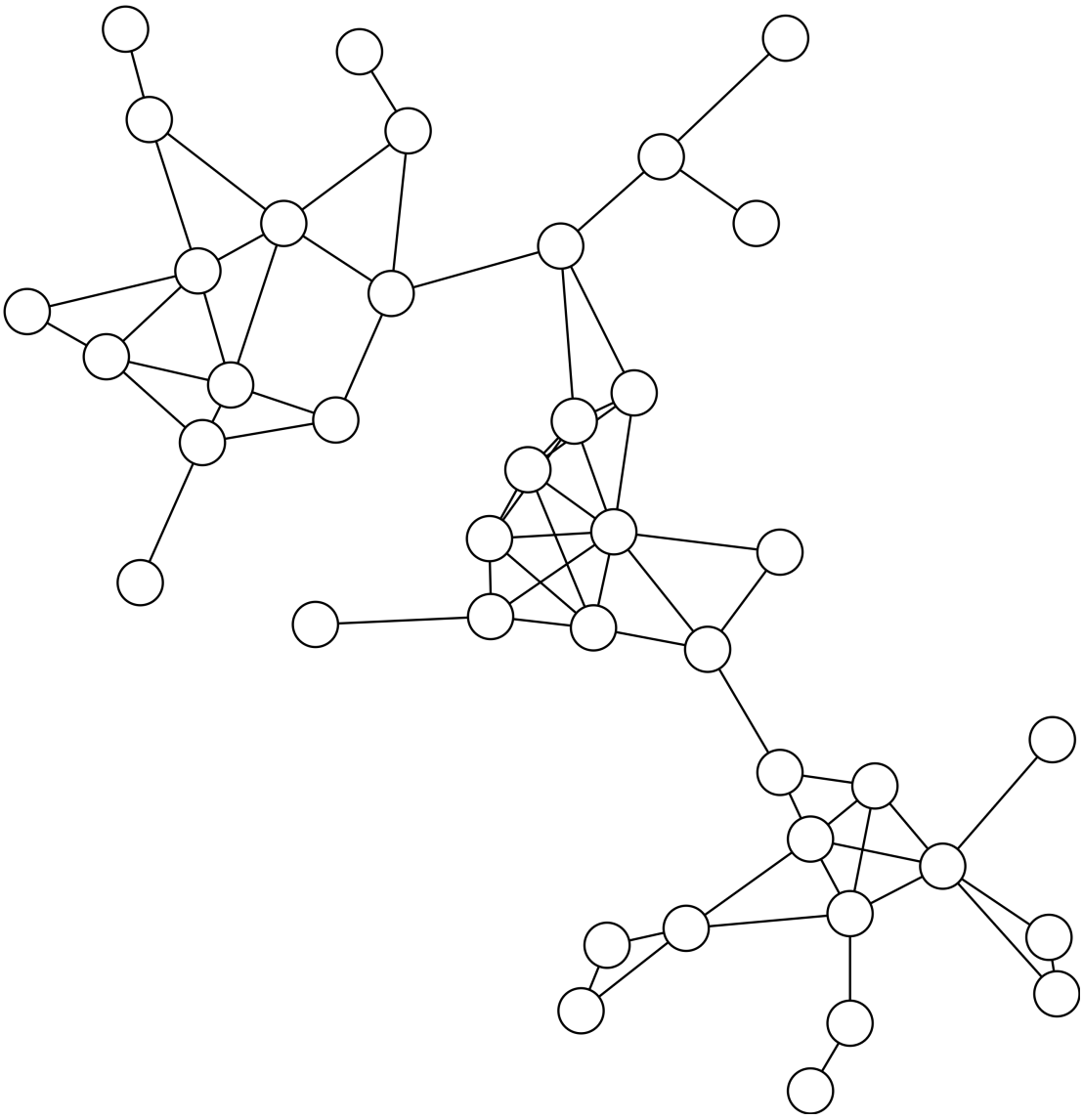


Figure 5.11: Sample topology of a random geometric graph [95].

5.4.3 Probabilistic Movement using Graph Centralities

In the previous section, *Path Optimizer* will always move gateway to create a shortcut for newly occurred communication **if there is at least one idle gateway and the path length after the shortcut creation is lower than the original.** Here we have one hypothesis.

Hypothesis: If the gateway moves to a bad location for all other nodes, it will cause an increase in the average shortest path length of inter-cluster communications, which eventually leads to a degradation of communication quality, even though the quality is improved in the short term.

In order to test this hypothesis, the author introduce a probabilistic movement into the *Path Optimizer* (Algorithm 6); that is, if a new communication satisfies its allowable delay under the current gateway placement and if there will be a significant degradation of the locational convenience relative to the improvement of communication delay, abort the moving.

The shortcut effect is defined as the ratio of the reduced hop count to the allowable delay N_s, and the locational convenience is evaluated using the centrality measure – algorithm computes the gateway centrality score in each of the graphs before and after the relocation and then calculate the ratio of these two scores N_c. If the ratio N_c is greater than 1, i.e., moving gateway improves both the delay and the location, immediate relocation would perform.

Input N_s and N_c into the sigmoid function of Eq. 5.6 (Fig. 5.12), and obtain P_s and P_c as continuous values of $[0, 1]$. The responsiveness for the input value can be adjusted by controlling the inflection point of Eq. 5.6. Next, substitute P_s and P_c into Eq. 5.7, and use the resulting P as the final probability of the moving. Note that all constant values in Eq. 5.6 and Eq. 5.7 are parameters to be tuned up for each environment by the operators, but simply using the standard sigmoid in Eq. 5.6 and $a = 0.5$ in Eq. 5.7 will also work.

This extended *Path Optimizer* as *Optimizer**.

$$\varsigma_a(x) = \frac{1}{1 + e^{-a(x-x_0)}} = \frac{\tanh\Big(a(x - x_0)/2\Big)+1}{2}, \tag{5.6}$$

$$P = \alpha P_s + (1 - \alpha)P_c. \tag{5.7}$$

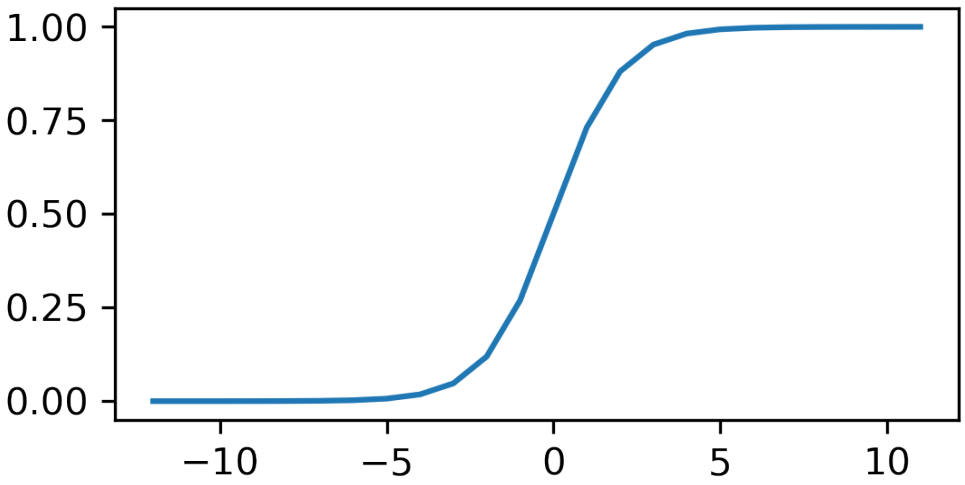


Figure 5.12: Standard sigmoid function ($a = 1, x_0 = 0$)

5.5 Performance Evaluation and Discussion

This section discusses the performance evaluation results of the proposed algorithm.

Four experiments using computer simulation were carried out to reveal the performance characteristics of *Link Stabilizer* and *Path Optimizer* in detail. Common parameters of test case generation are on Table-5.2 – the number of nodes and the communication radius assuming outdoor Wi-Fi use are set based on the experimental results presented by Yuan et al. in [32]. Due to the space limitation, only some of the

graphs appear here.

Table 5.2: Common parameters of test case generation.

Field size (square field)	250 [m]
Number of nodes per cluster	20, 30, 40
Number of clusters	2
Number of gateways	1, 2, 3, 4
Communication radius	30, 40, 50 [m]

5.5.1 Gateway Availability Improvement by Link Stabilizer

First, the connection stabilization by *Link Stabilizer* was investigated. Test cases were prepared under the condition of Table-5.3 - the refresh interval is the period in which *Link Stabilizer* collects the neighbors' positions and speeds and updates its own velocity. The computer simulated 600 seconds of node movement and took a snapshot of the graph every 0.1 second to get statistics of the gateway availability. Mobility models are the three models introduced in the previous section, and the

Table 5.3: Performance evaluation for *Link Stabilizer*.

Stabilizer refresh interval	0.5, 1, 2 [s]
Evaluation time	600 [s]
Snapshot interval	0.1 [s]

Table 5.4: Parameters of the random waypoint model.

Speed	1–5, 5–10 [m/s]
Pausing time	1–10 [s]

Table 5.5: Parameters of the SRCM.

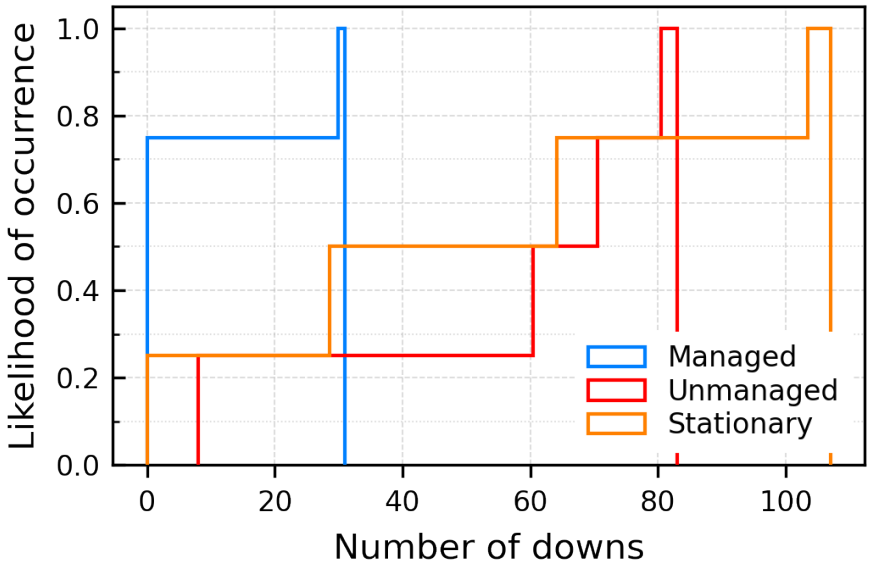
Speed	1–5, 5–10 [m/s]
Pausing time	1–10 [s]
Mobility radius	1–10, 1–30 [m]

Table 5.6: Parameters of the hovering model.

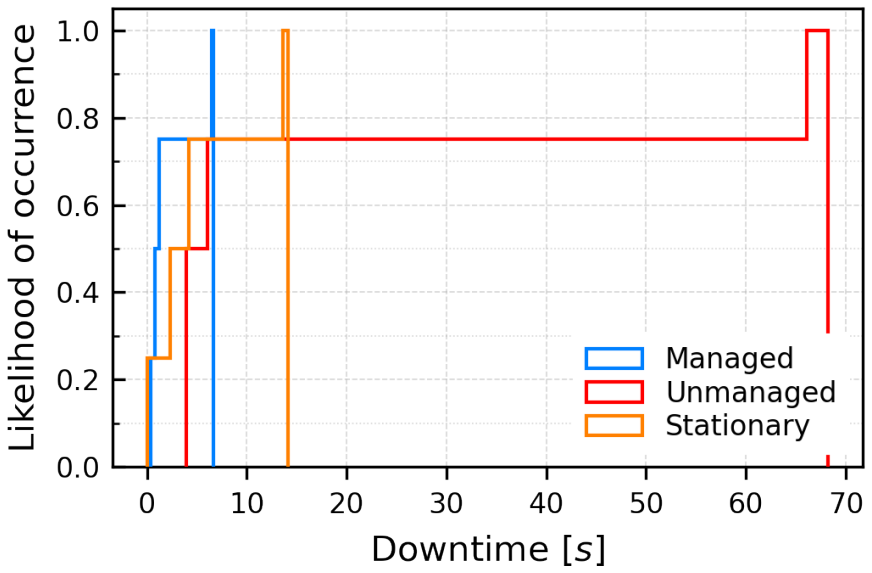
Probability of wind blowing	0.5, 1.0
Wind speed	1–10 [m/s]
Duration of wind	1, 3, 5, 7, 9 [s]

configurations of them are shown in Table-5.4 to Table-5.6.

The results are in Fig. 5.13-5.15. In every graph, the vertical axis is the probability of occurrence, and the horizontal axes are the number of gateway downs, the downtime, and the down interval in 600 seconds. The computer regarded gateway as down when it lost reachability to both clusters, i.e., when it could not function as an inter-cluster repeater. The Managed is where *Link Stabilizer* is used, the Unmanaged is where the gateway moves with mobility model like any other nodes, and the Stationary is where the gateway is fixed at its initial position. Each cluster has 20 nodes, the communication radius is 30 meters, and 4 gateways exist.

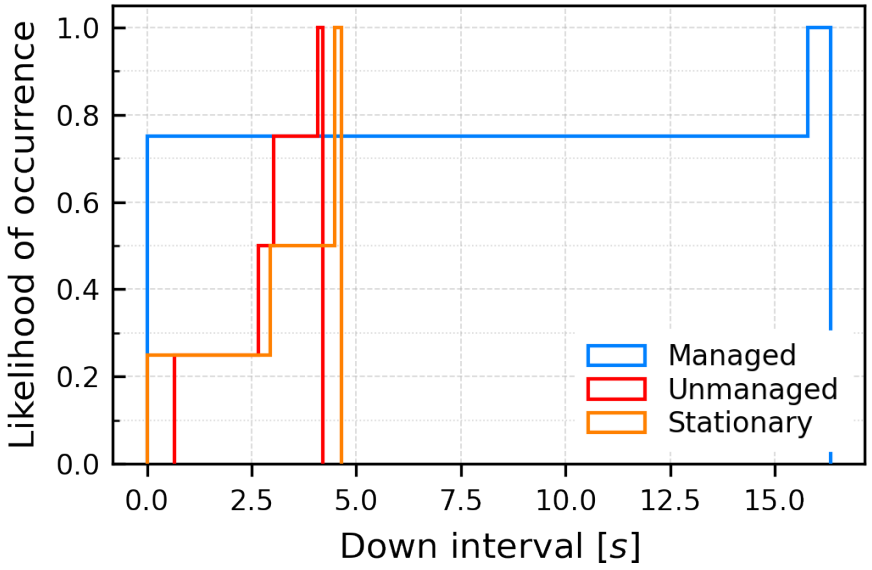


(a) Number of gateway downs.



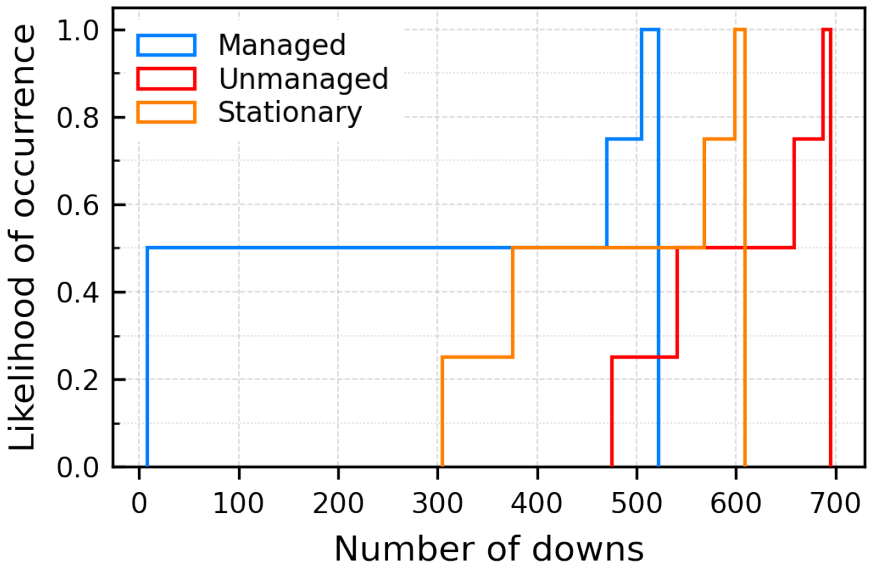
(b) Gateway downtime.

Figure 5.13: ECDF: Performance of the *Link Stabilizer* under the SRCM with 5–10 m/s of node speed and 1–10 meters of mobility radius.



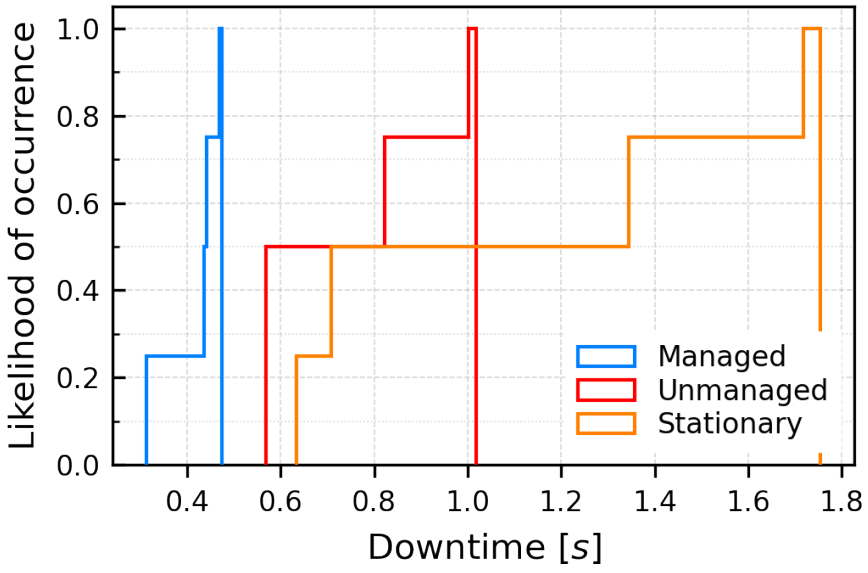
(c) Gateway down interval.

Figure 5.13: ECDF: Performance of the *Link Stabilizer* under the SRCM with 5–10 m/s of node speed and 1–10 meters of mobility radius.

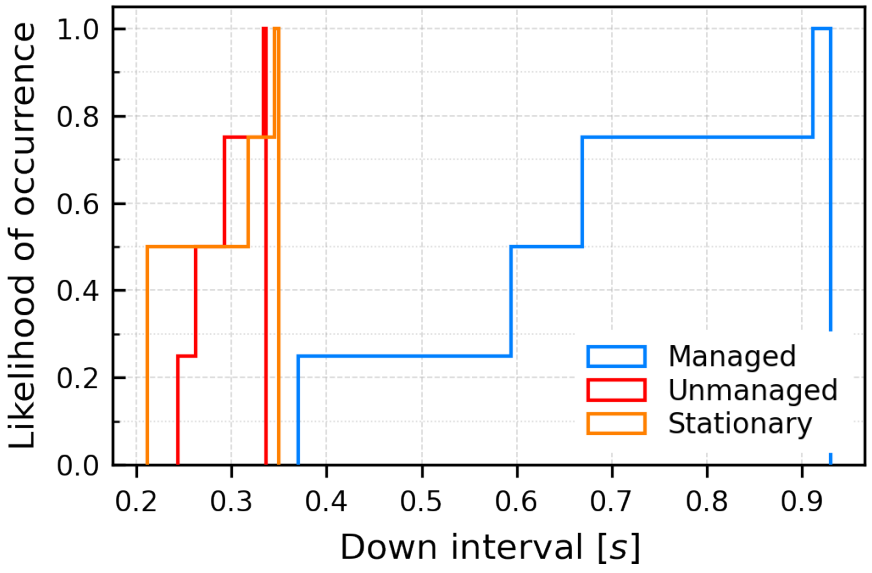


(a) Number of gateway downs.

Figure 5.14: ECDF: Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability.



(b) Gateway downtime.



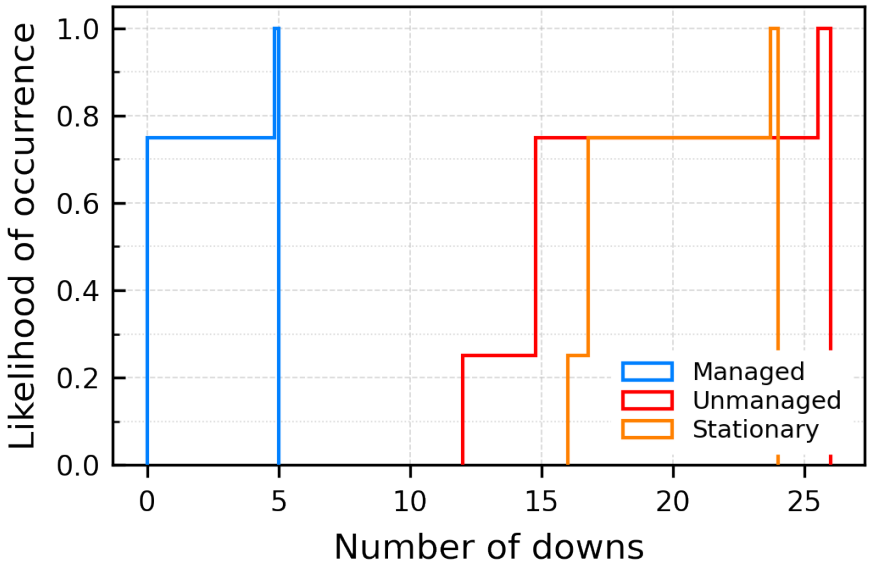
(c) Gateway down interval.

Figure 5.14: ECDF: Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability.

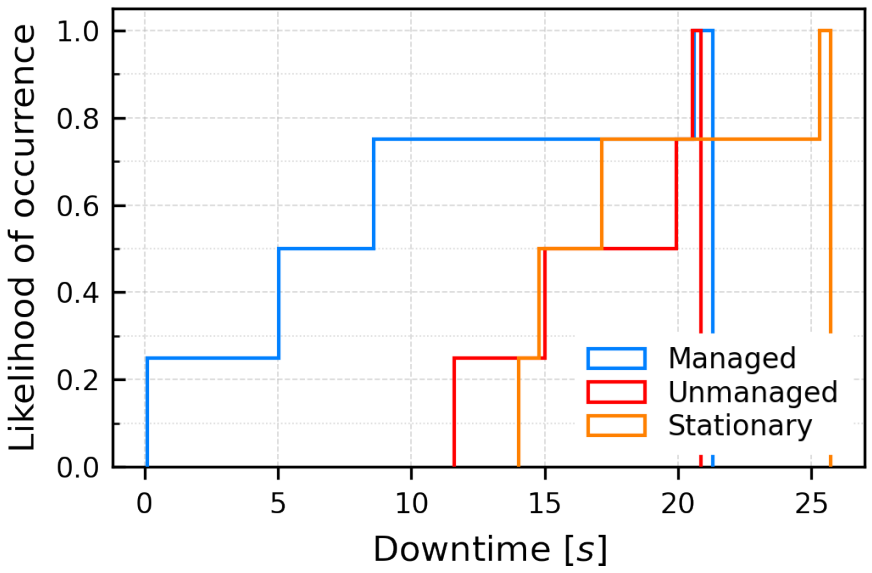
As clarified by the overall trend, *Link Stabilizer* has successfully stabilized the connectivity and improved the gateway availability. *Link Stabilizer* was designed as an algorithm to maintain the current connection as long as possible, and it works as expected; that is, it reduced the number of gateway downs and downtime and increased the down interval.

We can see that the hovering model has ten times more gateway downs and ten times shorter downtime than that of the SRCM, indicating a high frequency of very short-interval link disconnection and reconnection (Figs. 5.13-5.14).

In the random waypoint model, unlike the SRCM and the hovering model, the topology changes significantly from the initial one over time, so the connectivity from the perspective of non-gateway nodes are visualized as Fig. 5.16. Each of the three graphs plots the number of contacts, i.e., the number of times path reachability to one of the gateways was established, the sustained period of the gateway reachability, and the interval of every contact. Figures 5.15 - 5.16 suggest that *Link Stabilizer* can follow the neighbors and move long distances in the field if the node mobility is within the evaluation range. Therefore, we can say that *Link Stabilizer* is effective not only in the hovering environment but also in a more dynamic environment with widely distributed node movement.

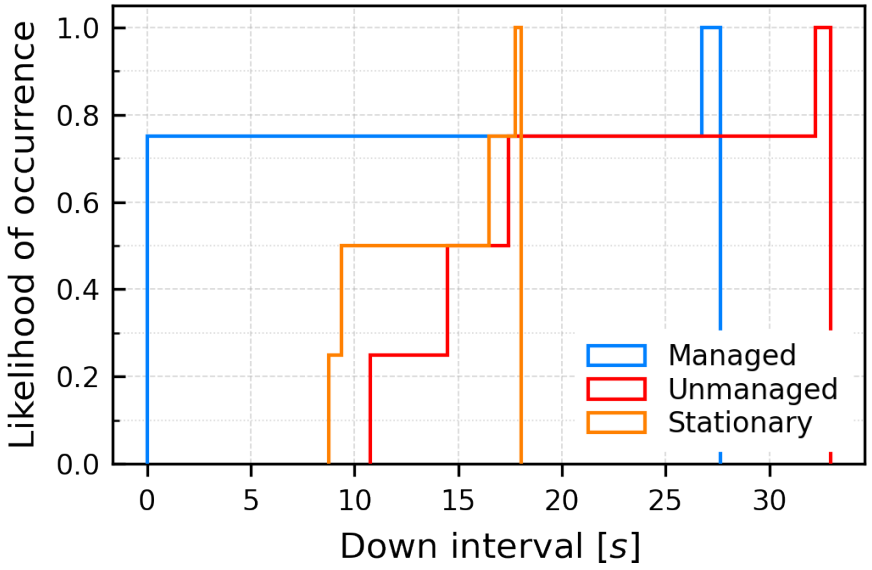


(a) Number of gateway downs.



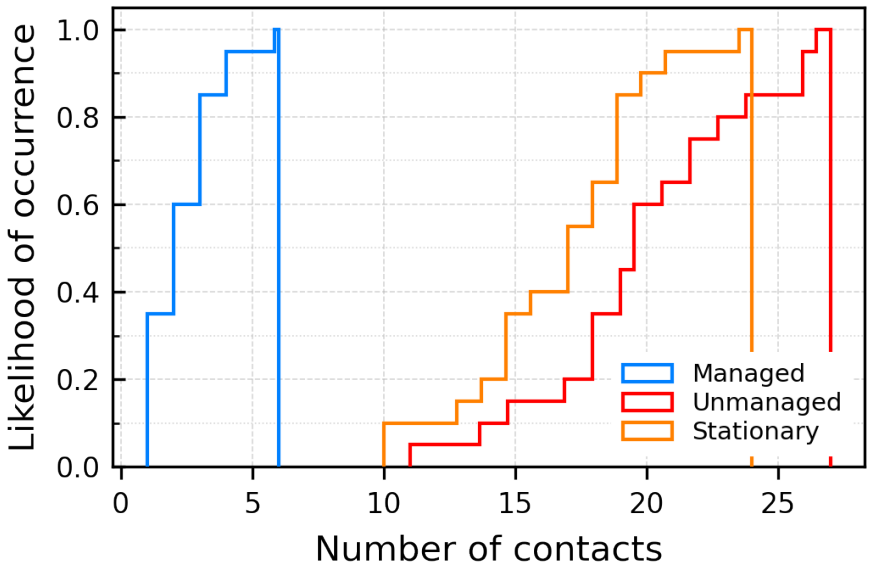
(b) Gateway downtime.

Figure 5.15: ECDF: Performance of *Link Stabilizer* under the random waypoint model with 5–10 m/s of node speed and 1–5 seconds of pausing time.



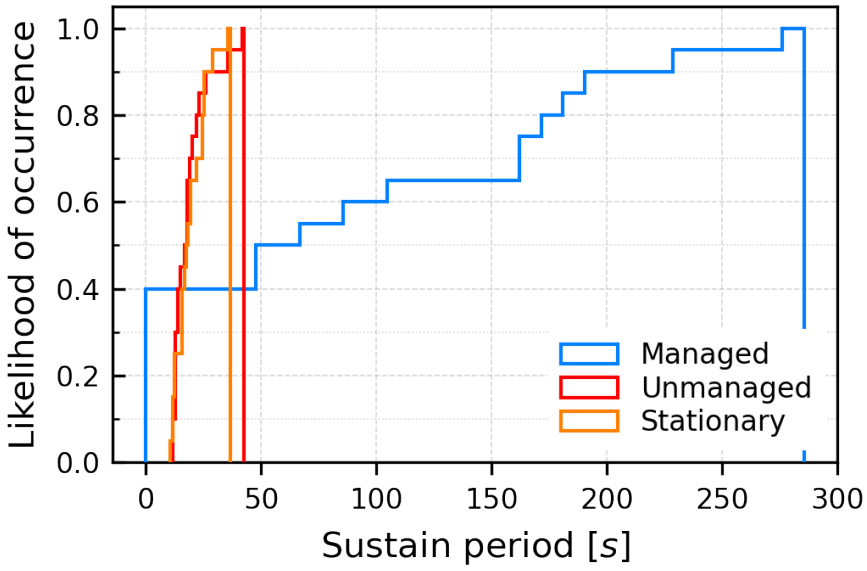
(c) Gateway down interval.

Figure 5.15: ECDF: Performance of *Link Stabilizer* under the random waypoint model with 5–10 m/s of node speed and 1–5 seconds of pausing time.

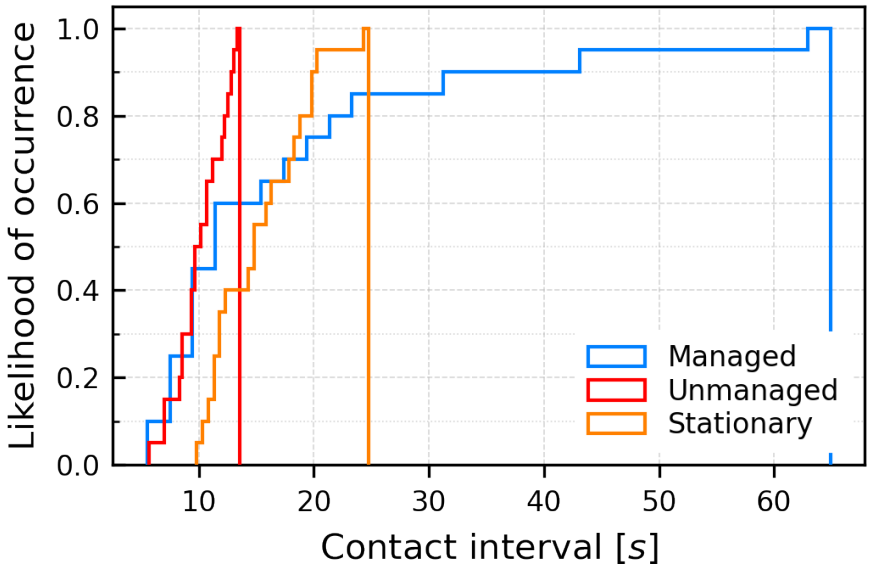


(a) Number of gateway contacts.

Figure 5.16: ECDF: Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability.



(b) Sustain period of gateway reachability.



(c) Gateway contact interval.

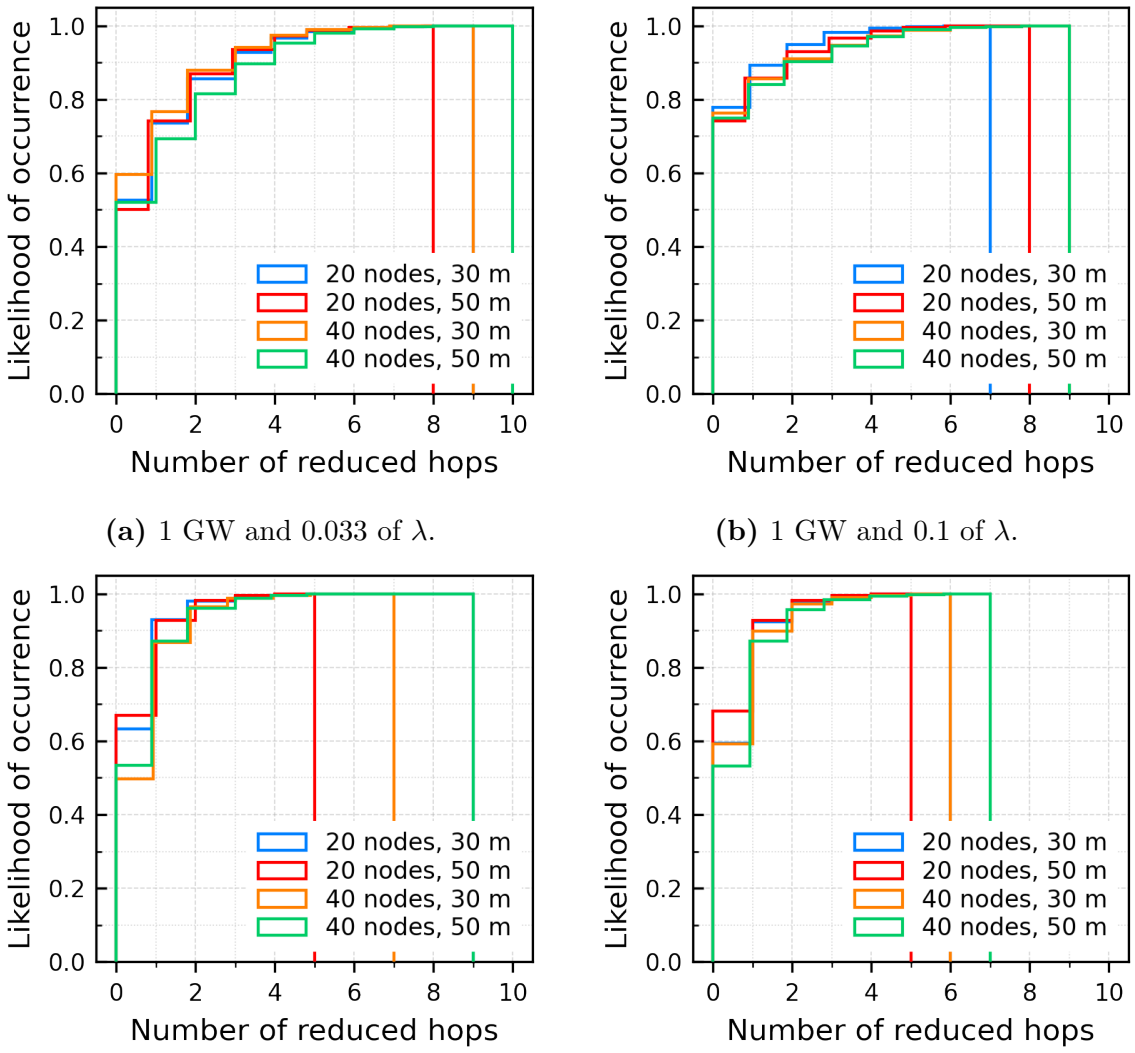
Figure 5.16: ECDF: Performance of *Link Stabilizer* under the hovering model with 7 seconds of duration and 100% of wind probability.

5.5.2 Delay Reduction by Path Optimizer

Next, the delay reduction achieved by *Path Optimizer* was investigated.. To focus on evaluating the shortcut formation ability, random geometric graph-based network topologies were used with the graph generator described in the previous section. In each test case, a total of 500 inter-cluster communications were generated sequentially at several frequencies shown in Table-5.7. The evaluator also took into account the traveling time of the gateways; that is, as explained in Chapter 3, broadcasted RREQs were not received by the traveling gateways, and thus the number of idle gateways decreases during this period.

Since *Path Optimizer* considers the path optimization every time new communication occurs, it is sure to reduce the average communication delay better than doing nothing - what we should pay attention to is the degree of improvement. Results are shown in Fig. 5.17 to Fig. 5.19.

Figure 5.17 is the ECDF of the number of reduced hops by *Path Optimizer*, comparing the performance in the different number of nodes and communication radius. In each case, the average delay was reduced by about one hop, which is equivalent to 33% of the allowable delay, and the maximum improvement in the allowable delay satisfaction rate was around 160%. Although test cases used this time are connected graphs, the number of reduced hops and the satisfaction rate must improve further if disconnected graphs were used instead.



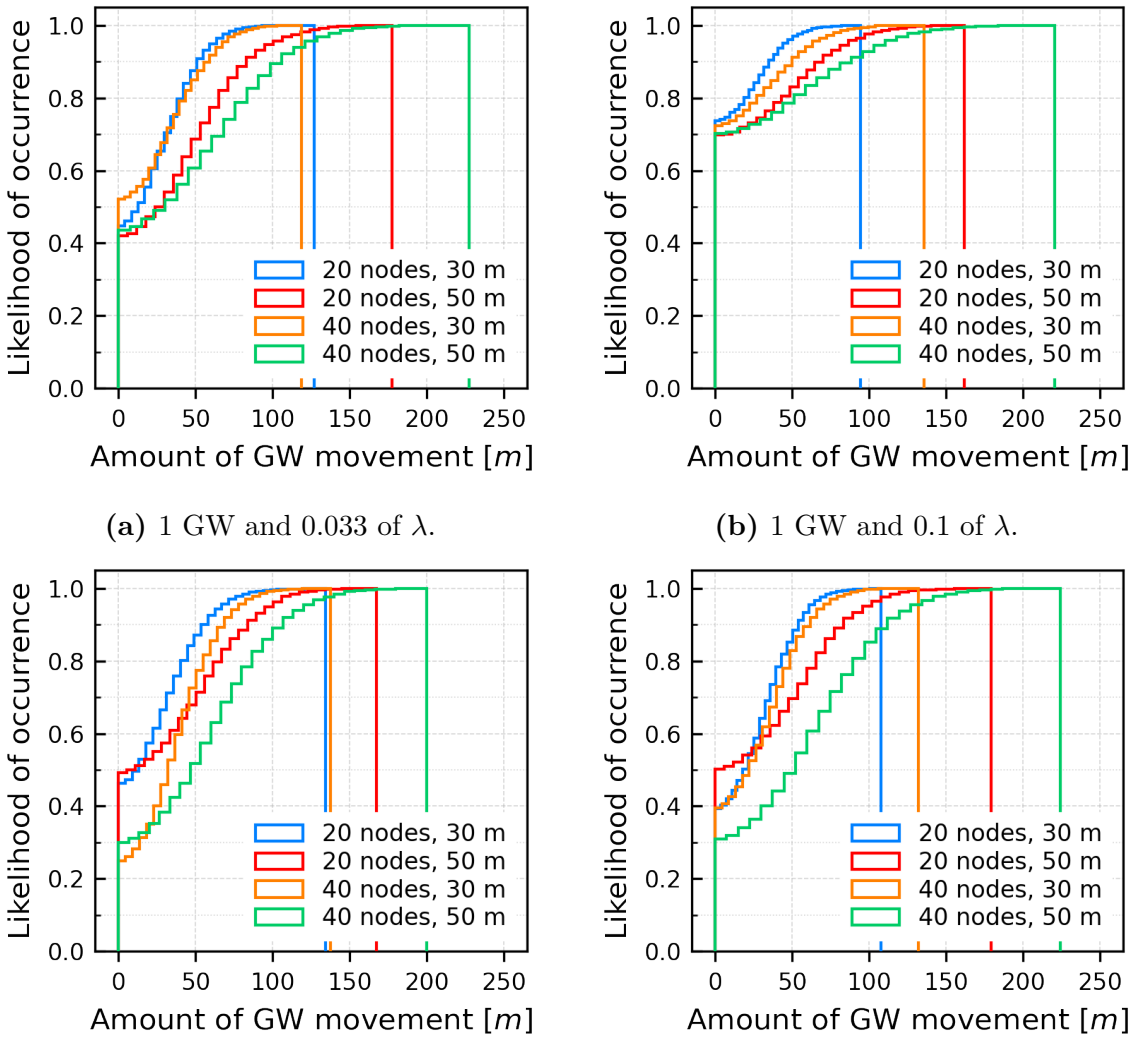
(a) 1 GW and 0.033 of λ.

(b) 1 GW and 0.1 of λ.

(c) 4 GWs and 0.033 of λ.

(d) 4 GWs and 0.1 of λ.

Figure 5.17: ECDF: Number of reduced hops by *Path Optimizer* under several conditions.



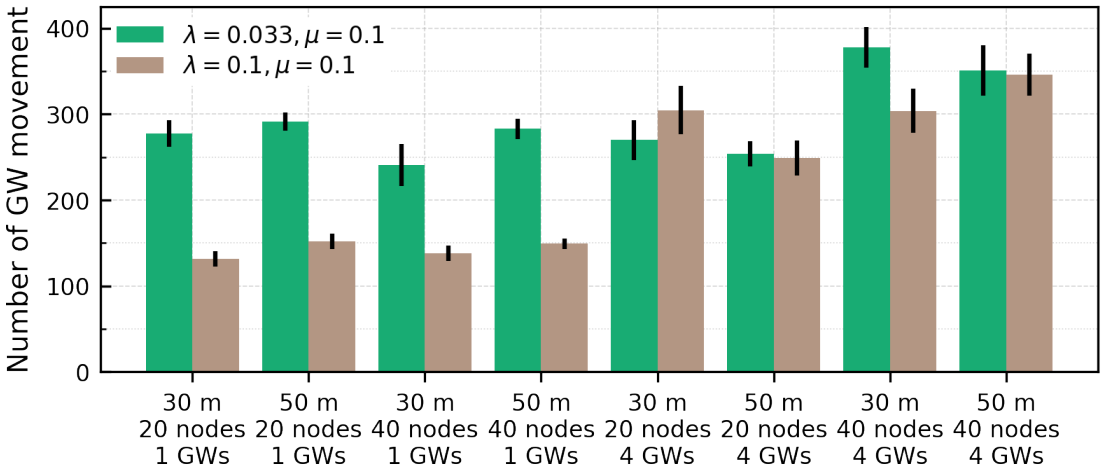
(a) 1 GW and 0.033 of λ.

(b) 1 GW and 0.1 of λ.

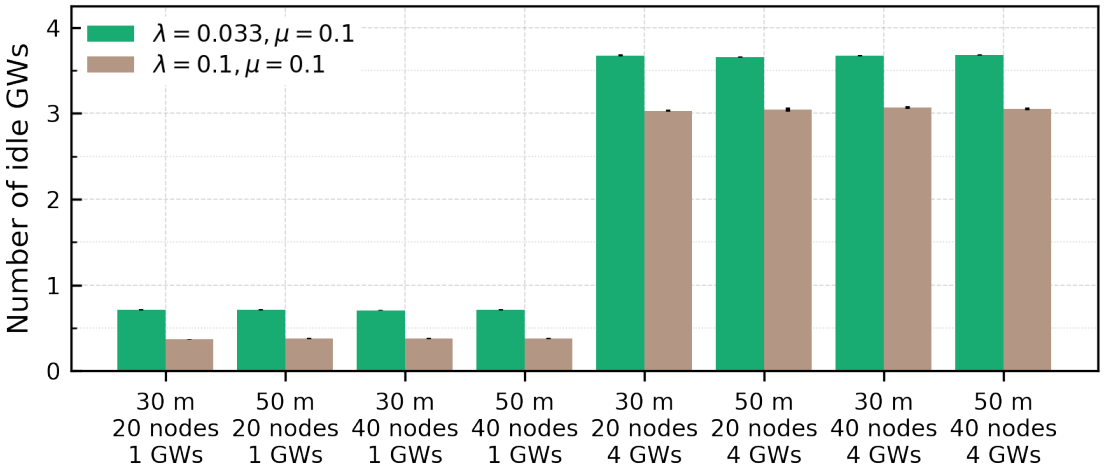
(c) 4 GWs and 0.033 of λ.

(d) 4 GWs and 0.1 of λ.

Figure 5.18: ECDF: Gateway traveling distance for every shortcut formation under several conditions.



(a) Total number of gateway relocations for shortcut formation.



(b) Average number of idle gateways.

Figure 5.19: Cost and availability: Total number of gateway relocations and average number of idle gateways.

Figure 5.18 is the ECDF of the traveling distance for every shortcut formation. In a 250 meters square field, the maximum traveling distance can be 354 meters, but the average is within the range of 50 to 100 meters under all conditions - doubling the number of nodes roughly doubles the distance. The maximum speed of a small drone weighing less than 1 kg is typically about 10 to 20 m/s, so the source node has a waiting time of about 5 seconds between the PREQ transmission and the PREP reception in Fig. 5.7 of Chapter 3.

In the last, the total number of gateway relocations and the average number of gateways in the idle state were shown in Fig. 5.19, comparing two different arrival rates λ, which mean the frequencies of inter-cluster communication occurrences. Throughout the evaluations, the following two facts can also be confirmed, and both of them are qualitatively convincing:

- The higher the arrival rate, the closer *Path Optimizer's* performance is to the Stationary, i.e., do nothing. It is because multiple communications go through a particular gateway simultaneously, making it difficult for the gateway to be released from the in-service state. Eventually, it leads to the decreased idle gateways of Fig. 5.19b, inadequate path hop reduction of Fig. 5.17, and slumped gateway relocation of Fig. 5.18 and Fig. 5.19a.

- If the arrival rates are equal, *Path Optimizer's* performance will improve as the number of gateways increases.

Path Optimizer reduced the delay under any conditions, though, generated test cases were relatively dense. For a sparser network, the average shortest path length will increase, and thus *Path Optimizer* must make a more noticeable improvement in the delay reduction.

Table 5.7: Performance evaluation for *Path Optimizer*.

Total number of flows	500
Dist. of arrival interval	Poisson process
Dist. of service time	Exponential distribution
Arrival rate: λ	0.033, 0.05, 0.1
Mean service time: μ	0.1
Allowable delay	3 [hops]

5.5.3 Optimizer vs. Optimizer*

The hypothesis proposed in the previous section was tested – will the additional mechanism decide the relocation timing make further improvement for the long-term allowable delay satisfaction rate?

The seven well-known graph centrality measures were selected for *Optimizer**: Degree, Closeness, Betweenness, Eigenvector, Katz, Stress, and Radiality. The centrality measure is the key to the mechanism – it numerically evaluates how graphically highly situated a gateway placement is.

Figure 5.20 shows the result. By introducing the mechanism, although the number of relocations and the total amount of the traveling distance was reduced for some measures, no significant difference in the satisfaction rate was observed, and the initial goal was not achieved. According to the hypothesis, the performance difference will be noticeable when the following two conditions are satisfied simultaneously:

- Gateway moves to a bad location where the average shortest path length increases significantly.
- Gateway is detained for a long time in there.

However, each condition satisfies only rarely, and thus the case for both satisfied at the same time is extremely rare. Even if such a rare case were to occur, the effect

of the performance improvement would be completely buried when we observe over a long-term period.

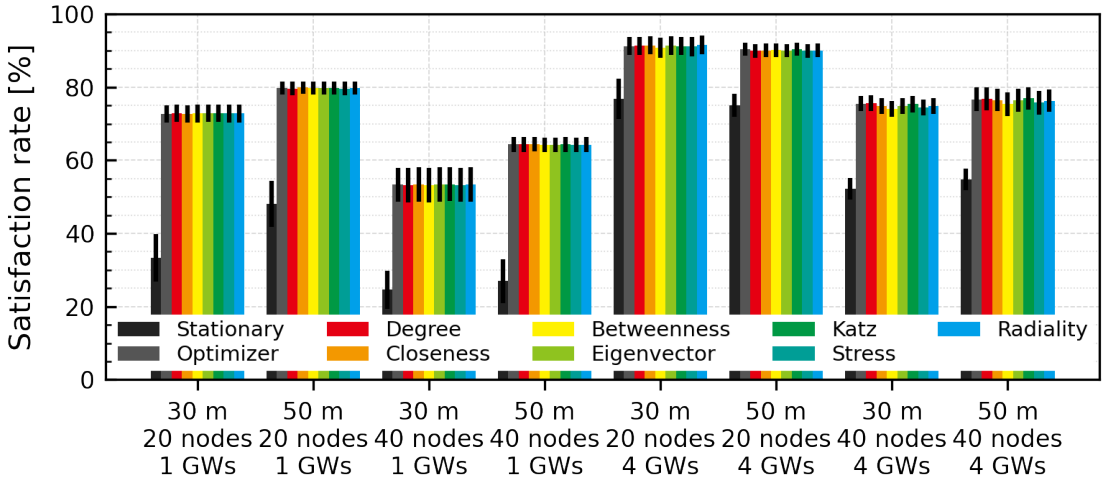
In short, the hypothesis was rejected. It was found that the naive implementation of *Path Optimizer* can ensure sufficiently high performance without an extra judgment mechanism - it would be a rather pleasant conclusion; a simple system is always better than an unnecessarily complex system.

5.5.4 Optimizer vs. Brute-force

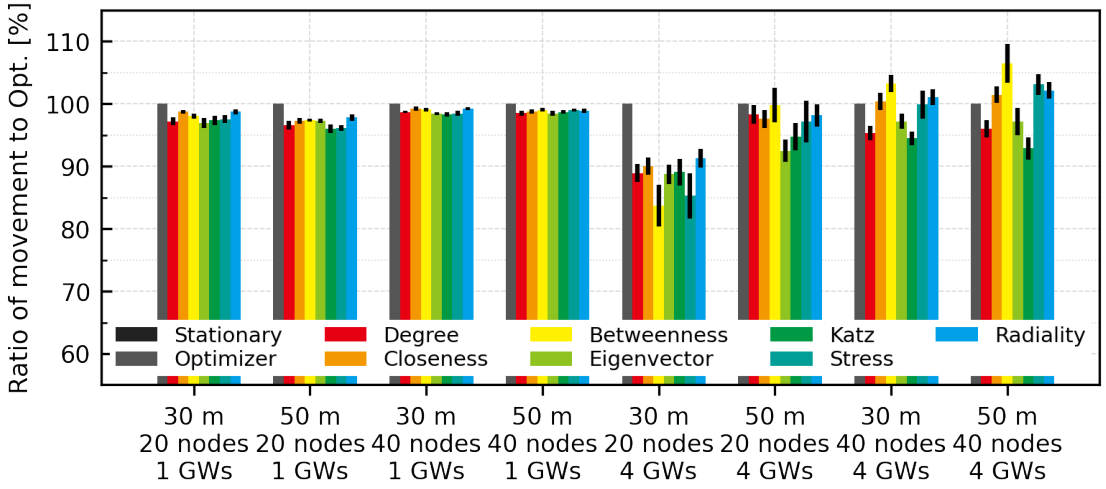
Finally, the performance of *Path Optimizer* was compared to the performance upper limit found by brute-force searching. Since the gateway's traveling time is ignored in the brute-force searching, the best flow accommodation patterns may include infeasible patterns in terms of time. Besides, due to the enormous computational cost for brute-force searching, this evaluation was performed with up to 6 flows.

Figure 5.21 shows the result – the allowable delay satisfaction rate and the total amount of gateway movement, and both are for 4 gateways. The *Path Optimizer's* score has reached the upper limit under all conditions, while the Stationary's score is unstable and around 60% of the limit.

Therefore, we can conclude that *Path Optimizer* is very useful for the target application, heterogeneous drone swarms, as it is lightweight and provides a stable delay improvement enough to reach the performance limit in the short term.

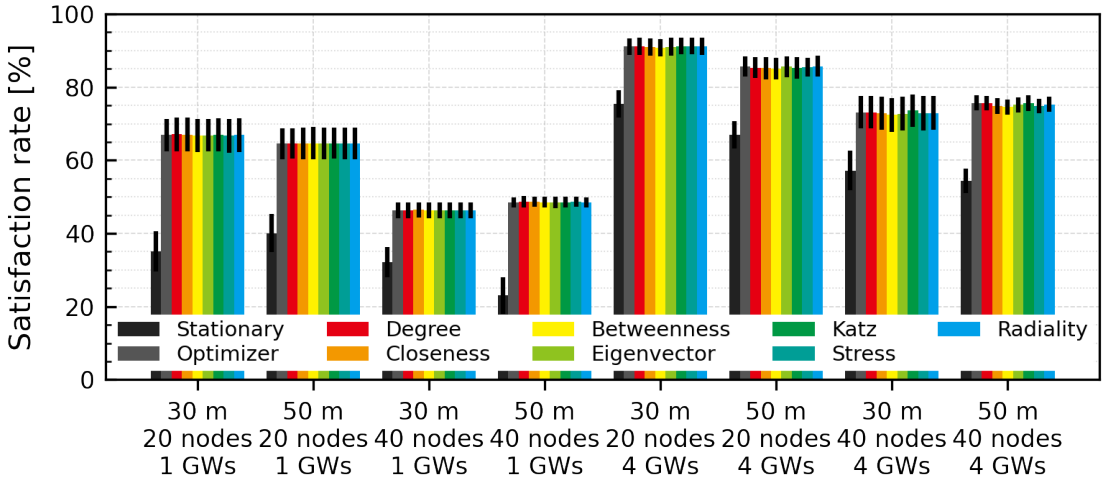


(a) Allowable delay satisfaction rate with 0.033 of the arrival rate.

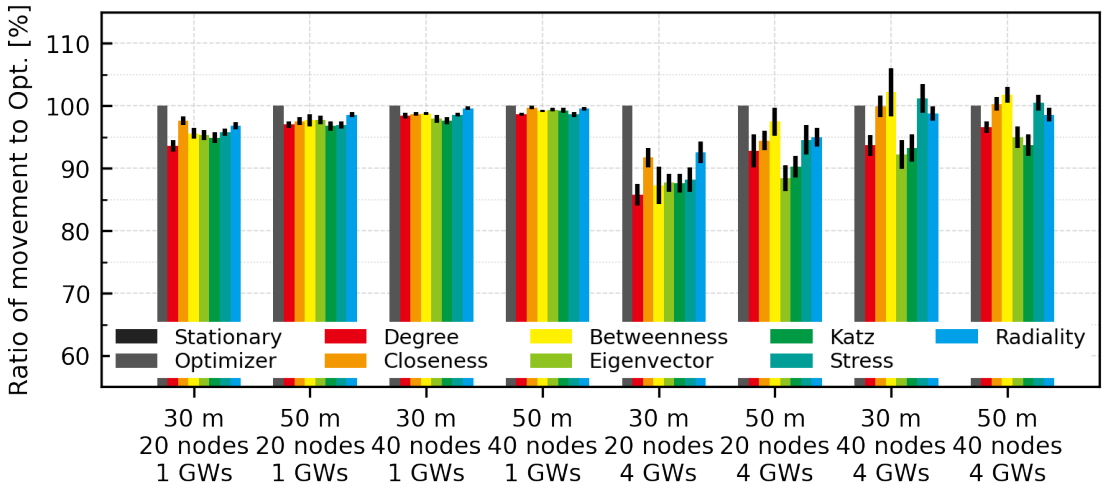


(b) Total amount of gateway movement with 0.033 of the arrival rate.

Figure 5.20: Satisfaction rate and mobility cost: Performance of the Optimzier* compared with the Stationary and *Path Optimizer*.

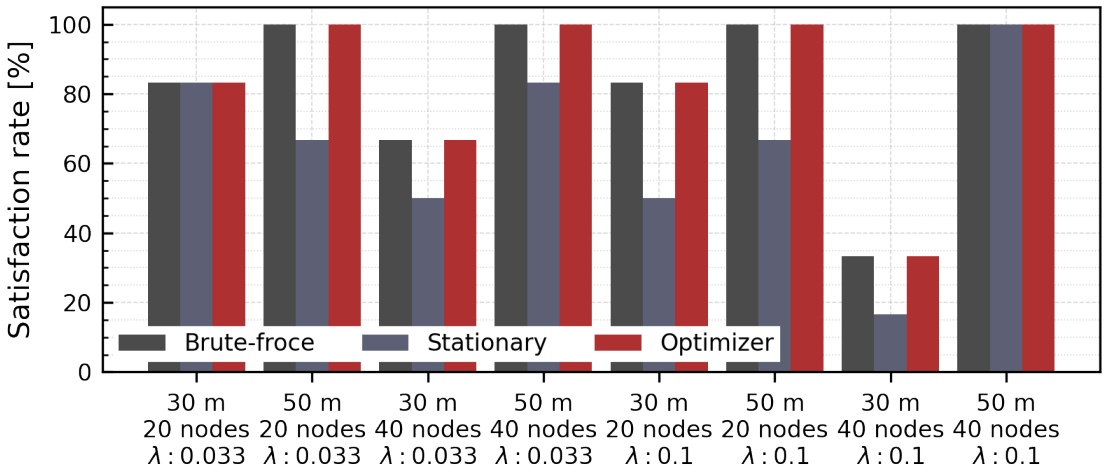


(c) Allowable delay satisfaction rate with 0.1 of the arrival rate.

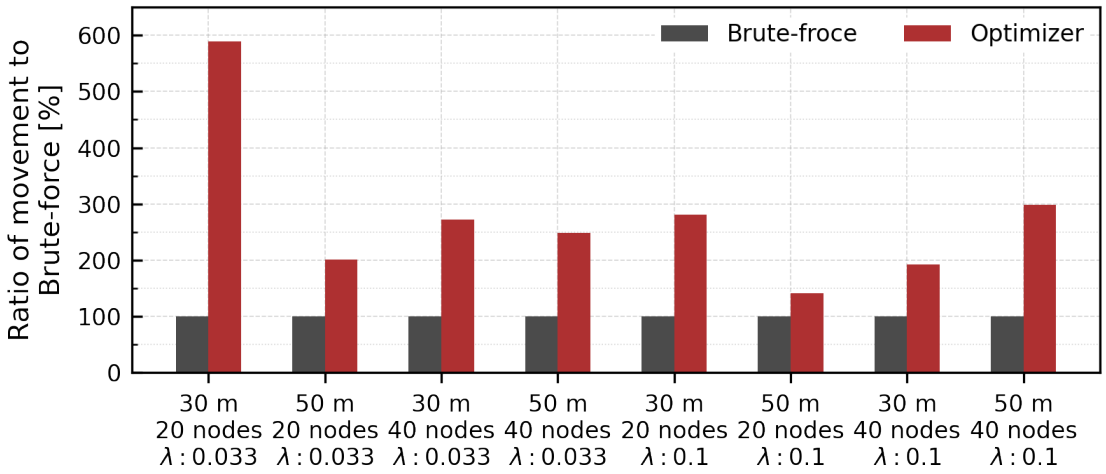


(d) Total amount of gateway movement with 0.1 of the arrival rate.

Figure 5.20: Satisfaction rate and mobility cost: Performance of the Optimzier* compared with the Stationary and *Path Optimizer*.



(a) Allowable delay satisfaction rate with 4 gateways.



(b) Total amount of gateway movement with 4 gateways.

Figure 5.21: Performance upper limit: Performance of *Path Optimizer* compared with the Stationary and Brute-force.

5.6 Conclusion

This work proposed a gateway mobility control algorithm assuming the heterogeneous drone swarms, i.e., two drone clusters with independent wireless protocols are interconnected by some translator (gateway; GW) drones. Besides, only gateways are assumed to be controllable and relocatable, while other existing studies regarded all nodes as controllable.

The proposed algorithm is composed of two sub-algorithms – *Link Stabilizer* and *Path Optimizer*. *Link Stabilizer* maintains the neighbor links and consists of two schemes: the neighbor clustering based on the relative velocity and the velocity calculation based on the kinetic model. *Path Optimizer* creates a shortcut reactively to reduce the end-to-end delay for newly occurred communication.

After proposing a conceptual protocol design to implement the algorithm into real-world HANETs in a distributed manner, computer simulations were carried out for performance evaluation. The results revealed that *Link Stabilizer* improved the connection stability even when the node mobility is slightly high, and *Path Optimizer* reduced the communication delay by the optimal shortcut formation under any conditions of the experiments.

Chapter 6

Path Coordinator

6.1 Introduction

First, the updates from the previous work are explained through the description of the application scenario. The target application is for wildlife activity surveillance, though, as mentioned above, this study can be applied to various situations such as monitoring critical facilities or inspecting disaster areas.

1. Camera-equipped drones with two different radio protocols hover and standby above the monitoring area.

2. Two independent ad hoc networks for each radio protocol are configured with HWMP.

3. Some gateway drones are deployed to integrate the two networks into HANETs – completing the building of the heterogeneous drone swarms.

4. The operator connects to the HANETs from any location via the nearest drone and requests multi-hop live video transmission of any remote drone.

5. One of the idle gateways move automatically for each new communication request to add a shortcut in the network, minimizing the end-to-end hop distance – *Path Optimizer*.

6. The operator waits for shortcut creation completion, then begins watching a high-quality video.

The above has been dealt with in the previous study, and the following is an updated situation that dealt with in this study (Fig. 6.1).

1. New video requests are occurred one after another without discarding the existing video sessions.

2. Every gateway in the network relays at least one video, and idle gateways are exhausted.

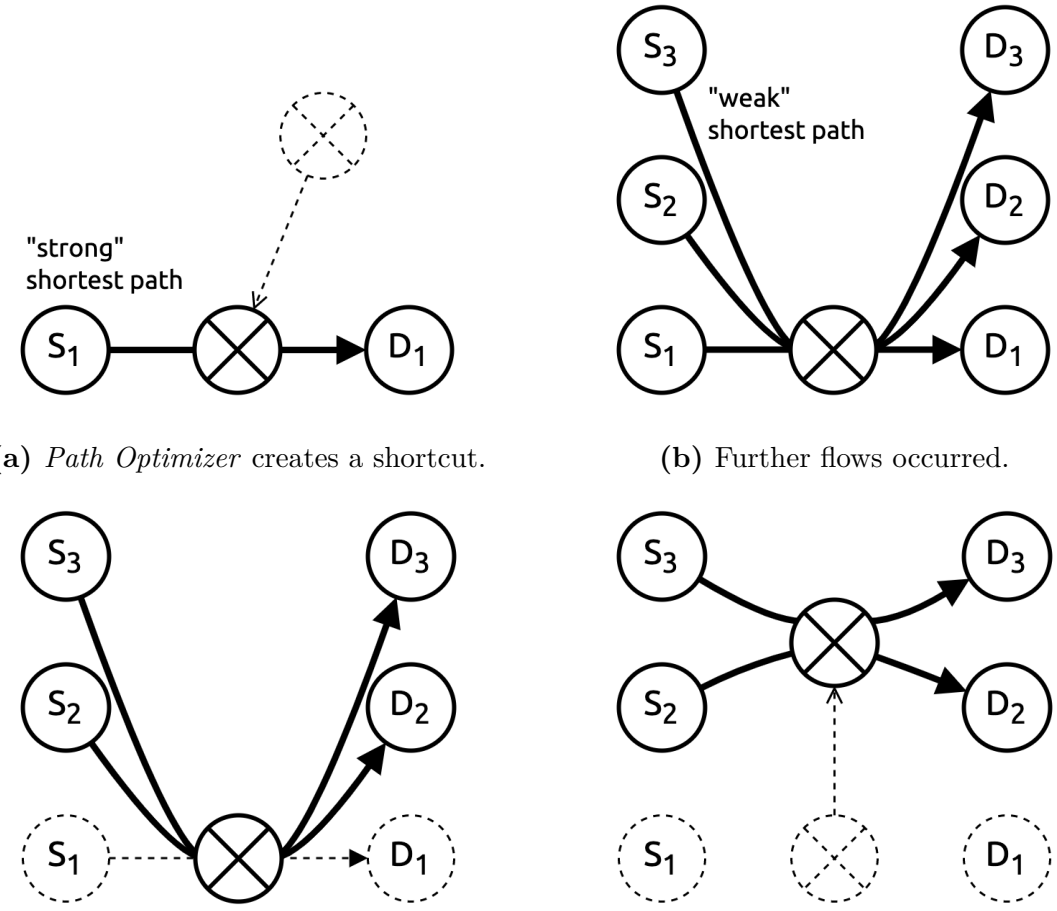
3. After the occurrence of further video requests, shortcut creations are no longer performed, and all the following video sessions will select the nearest gateway and pass through it. Instead of the "strong" shortest path, every session has to use the "weak" shortest path. Again, note that "strong" is the true best path prepared by the on-demand shortcut creation, though "weak" is only the best for the current topology.

4. There are differences in video quality before and after the exhaustion – some videos may have extremely low quality.

5. The operator executes the gateway relocation command.

6. After all videos are temporarily disconnected, all gateways cooperate to autonomously move to the optimal position.

7. The operator waits for the relocation completion, then resumes watching uniformly high-quality videos.



(a) *Path Optimizer* creates a shortcut.

(b) Further flows occurred.

(c) The first flow has disappeared.

(d) Optimize location.

Figure 6.1: Target situation of this paper: dynamically relocate gateways according to the situation to keep the locations optimal for the current flows.

6.2 Proposed Algorithms

This section shows the proposed algorithm, *Path Coordinator*, at length and explains how it improves the communication quality in heterogeneous drone swarms.

6.2.1 Algorithm Overview

Extending the previous design, the gateway service is extended to a 3-states machine of Fig. 6.2.

- **Idling state:** gateway is not relaying any communication flows but is ready to accept new flows – position unlocked and can move freely.
- **Working state:** gateway is relaying one or more flows – position locked and cannot make a large move enough to lose its neighbor connectivity.
- **Pending state:** gateway has temporarily suspended relaying and accepting flows.

The earlier proposal, *Path Optimizer*, worked only with gateways in the idling state. *Path Coordinator* works in the pending state-s, transited from the idling or working states after receiving the command from the operator. Note that whether the pending gateways return to the idling or working states depends on the situation.

Path Coordinator consists of two phases – the relocation phase and the rerouting phase. First, in the relocation phase, the gateway relocation algorithm calculates the gateway's local optimal position for the active flows and moves the gateway geographically. Then, in the rerouting phase, the flow rerouting algorithm searches the shortest path of each flow again and migrates flows between gateways as necessary. Each phase is independent, and it is possible to perform only the relocation phase and skip the rerouting phase or vice versa.

- **Relocation phase:** Simultaneous gateway relocation with geographic movement (Algorithm 7)

- **Rerouting phase:** Recalculate the optimal path for each flow under the current gateway positions (Algorithm 8)

The operators are presumed to send the command after the idle gateway exhaustion occurred to fire *Path Coordinator*. Besides, *Path Coordinator* requires that each gateway have full knowledge of both clusters' network topology. Sect. 6.3 discusses this more, but properly defining the procedure makes it possible to implement *Path Coordinator* in an autonomous, distributed, and cooperative manner.

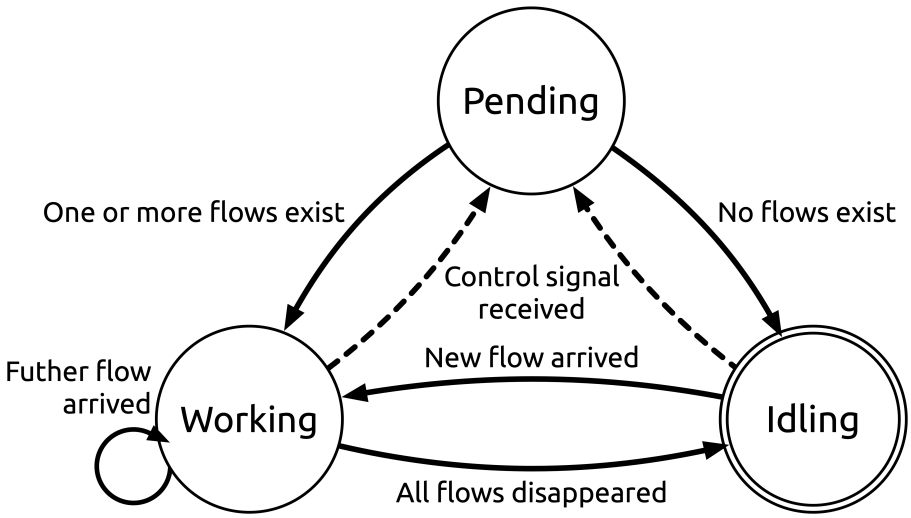


Figure 6.2: Gateway as a 3-states machine.

6.2.2 Gateway Relocation Algorithm

As shown in Fig. 6.1, there is no guarantee that every gateway is always on the end-to-end shortest path for each relaying flow when more flows than the number of gateways are repeatedly generated and disappear. The gateway relocation algorithm tries to improve this situation by geographical relocation of all gateways.

Each gateway searches for its ideal geolocation for the flows currently relaying; that

is, it calculates for the position where the sum of the number of end-to-end hops of relaying flows is minimized or less than the budget. The pseudocode expressed in Algorithm 7 means the following procedure. There are three steps.

1. Calculate vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ using Dijkstra's algorithm
2. Create a set Ω by combinatorial enumeration
3. Calculate all possible values of cost C and find the minimum value

Let G_A and G_B be the two clusters making up the network. Suppose the interest gateway relays m flows of $f_1, f_2, \ldots, f_m \in F$. Using Dijkstra's algorithm, calculate the shortest path with source node $s_i \in G_A$ and destination node $t_i \in G_B$ of flow f_i as starting points – a total of $2m$ searches. Define $\boldsymbol{a_{i,n}}$ as the shortest path length from the source node s_i to node $n \in G_A$, $\boldsymbol{b_{i,m}}$ as the shortest path length from node $m \in G_B$ to the destination node t_i, and a_n and b_n as in Eq. 6.1.

$$\boldsymbol{a_n} = \begin{pmatrix} a_{1,n} \\ a_{2,n} \\ \vdots \\ a_{m,n} \end{pmatrix}, \quad \begin{pmatrix} b_{n,1} \\ b_{n,2} \\ \vdots \\ b_{n,m} \end{pmatrix} = \boldsymbol{b_n}. \tag{6.1}$$

With the communication radius R, define a node pair set Ω as in Eq. 6.2. The node pair included in this set can build a new link bridging the two nodes by placing gateways in their intermediate position.

$$\Omega = \{(\boldsymbol{u}, \boldsymbol{v}) \mid \boldsymbol{u} \in G_A, \boldsymbol{v} \in G_B, \ \|\boldsymbol{u} - \boldsymbol{v}\| \leq 2R\}. \tag{6.2}$$

Define the evaluation cost $C_{(u,v)}$ of node pair (u, v) as in Eq. 6.4, where $\boldsymbol{J_m}$ is an all-ones vector of length m. Note that $C_{(u,v)}$ has a scalar value by the inner product and represents the total end-to-end hop distance of flows $f_1, f_2, \ldots, f_m$ when the gateway is placed to bridge node u and node v.

$$\boldsymbol{L}_{(u,v)} = \boldsymbol{a}_u + \boldsymbol{b}_v, \tag{6.3}$$

$$C_{(u,v)} = \boldsymbol{L}_{(u,v)}^T \boldsymbol{J}_m. \tag{6.4}$$

Since the total number of flow's path hops should be minimized, the objective function is set as Eq. 6.5. Obviously, its computational cost is $O(|\Omega|)$.

$$\arg\min_{(u,v)\in\Omega} C_{(u,v)}. \tag{6.5}$$

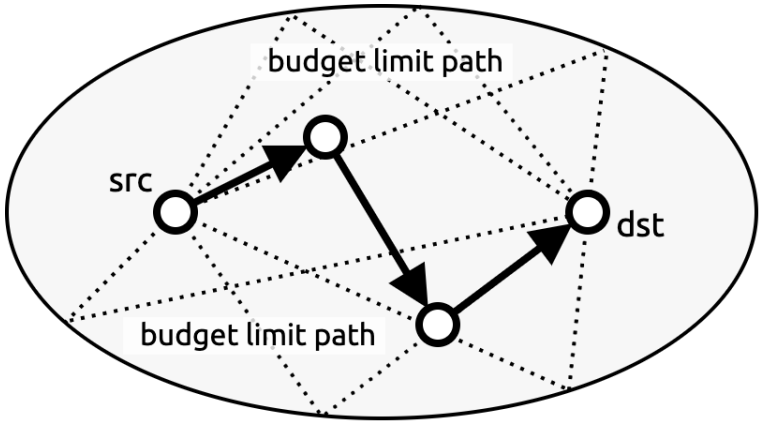


Figure 6.3: A 3-hop flow and its hop budget area surrounded by an ellipse curve.

The algorithm has some room for performance improvement in terms of implementation. For example, it may not be necessary always to minimize the sum of the hop counts in a practice scenario. If the number of path hops dominates the communication delay, and if an allowable delay exists for each application, the objective of minimizing the total hops (communication delay) can be simplified to maximizing the allowable hop (allowable delay) satisfaction rate. Using allowable delay also brings reduced computational costs.

Modifying the above procedure as follows achieve the best-effort allowable delay satisfaction rate maximization. First, let ϕ_f be nodes inside the hop budget area of flow f (Fig. 6.3), and define a node set Φ as in Eq. 6.6. If the flow f satisfies its allowable delay, then the end-to-end path consists of the nodes in ϕ_f. ϕ_f is an ellipse with the source and the destination at its focus.

$$\Phi = \bigcup_{f \in F} \phi_f. \tag{6.6}$$

Introduce a judgement function h_d as in Eq. 6.7. It determines whether the end-to-end path-hops is less than or equal to the hop budget d.

$$h_d(x) = \begin{cases} 1 & (x \leq d) \\ 0 & (x > d) \end{cases} \tag{6.7}$$

The new evaluation cost $\hat{C}_{(u,v)}$ becomes as in Eq. 6.9. This cost represents the number of allowable hop satisfying flows out of all flows $f_1, f_2, \ldots, f_m$ when the gateway is placed to bridge node u and node v.

$$\boldsymbol{L}^T_{\boldsymbol{(u,v)}} = (l_1, l_2, \ldots, l_m). \tag{6.8}$$

$$\hat{C}_{(u,v)} = \sum_{i=1}^{m} h_d(l_i). \tag{6.9}$$

Since the allowable delay satisfaction rate should be maximized, the objective function is set as Eq. 6.10. Basically, the evaluation is repeated for all node pairs (u, v) in the search space, but the optimal solution can be found probabilistically earlier by prioritizing the area where ϕ_f overlaps most.

$$\underset{(u,v)\in\Omega\cap\Phi}{\arg\max}\ \hat{C}_{(u,v)}. \tag{6.10}$$

The smaller the number of flows m is, the smaller the hop budget d is, the smaller Φ is. Therefore, the search space of node pairs $\Omega \cap \Phi \subseteq \Omega$ gateway can be reduced and the computational cost is lessen to $O(|\Omega \cap \Phi|)$ at most. The greater the hop budget d is, the greater the Φ is; however, the number of node pair (u, v) satisfying $\hat{C}_{(u,v)} = m$ will increase, so eventually, it might be able to find the solution earlier.

In the computer experiments conducted in Sect. 6.5, multiple hop budgets is set to evaluate the algorithm's performance. Nevertheless, since the hop budget varies depending on the topological characteristics of the network, operators are required to find the optimal value by themselves in an actual scenario.

Algorithm 7 Adaptive Gateway Relocation

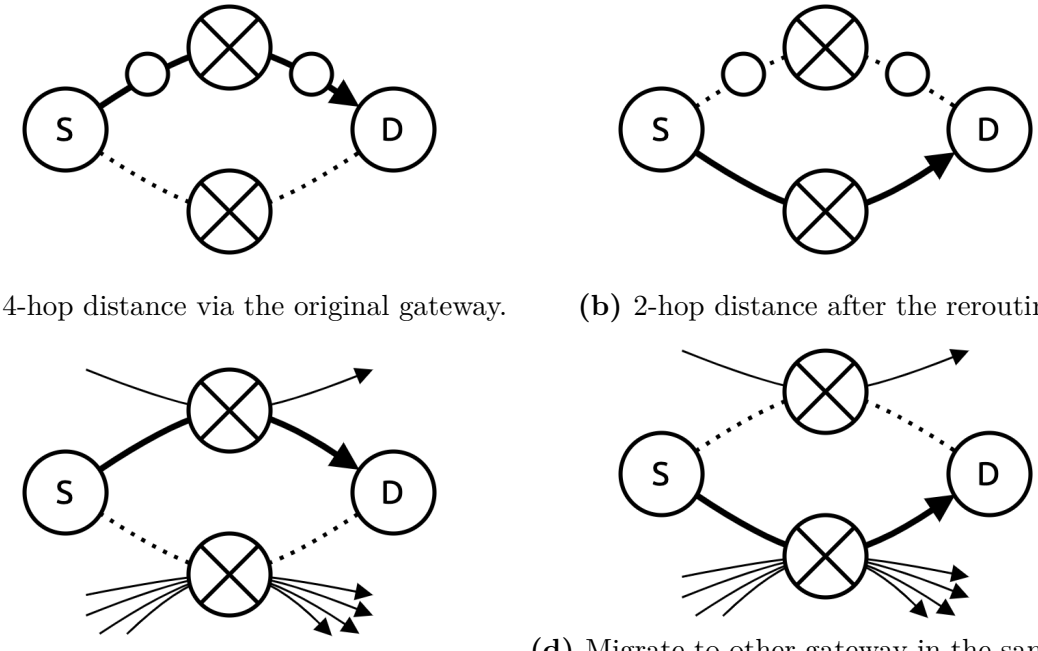
Input: Current network $G = (V, E)$, communication radius R, flow set F, and transiting gateway g

Output: Optimal 's position vector $\boldsymbol{p}$

function FindOptimalPosition(G, R, F, g)
  G_R, $G_B \leftarrow$ Split(G, g)
  $\boldsymbol{a_n}, \boldsymbol{b_n} \leftarrow \mathbf{0}, \mathbf{0}$
  for all $f_{s_i,d_i} \in F$ **do**
    $a_{i,n} \leftarrow$ Dijkstra(G_R, s_i)
    $b_{n,i} \leftarrow$ Dijkstra(G_B, d_i)
  $\Omega \leftarrow \{\}$
  for all $(u, v) \in V \times V \setminus E$ **do**
    if $||\boldsymbol{u} - \boldsymbol{v}|| \leq 2R$ **then**
      $\Omega \leftarrow \{\boldsymbol{a_n} + \boldsymbol{b_n}, (u, v)\}$
  $(u_0, v_0) \leftarrow$ Minimize(Ω)
  return $(\boldsymbol{u_0} + \boldsymbol{v_0})/2$

6.2.3 Flow Rerouting Algorithm

After running the gateway relocation algorithm, every gateway is located in the optimal position for its relaying flows. However, the optimal gateway itself for each flow may have been changed due to the topology updates caused by gateway's geographic movement; that is, the gateway on the shortest path connecting source and destination nodes may have changed. Here the flow rerouting algorithm reconfiguring all 's flow accommodation based on the following strategy (Fig. 6.4) is proposed. These are listed in order of priority and are evaluated from the top.



(a) 4-hop distance via the original gateway.

(b) 2-hop distance after the rerouting.

(c) Minimum distance but few other flows.

(d) Migrate to other gateway in the same distance.

Figure 6.4: Flow rerouting strategies: first minimize the path hops, then aggregate flows among gateways.

1. If there is a gateway that can further reduce the flow's end-to-end distance, then move the flow to the gateway.

2. If there are multiple gateways with the same end-to-end distance, then move the flow to the gateway accommodating the most flows.

3. If the above two rules are not applicable, continue to use the current gateway without taking any action.

The algorithm tries to concentrate flows to specific gateways as much as possible to make it easier for the working gateways to return to the idling state. *Path Optimizer*, is a powerful method to shorten the end-to-end distance – physically moving the gateways in response to new communication requests and building shortcut links between clusters. Since *Path Optimizer* works only with idling gateways, keeping as many idle gateway as possible is advisable to prepare for future communications.

The algorithm's behavior described in the pseudocode (Algorithm 8) is summed up as follows. Note that the concrete ways to share information and move flows among gateways are proposed in Sect. 6.3.

1. Collects flow information accommodated in other gateways to grasp all existing flows in the network.

2. Calculate the end-to-end distance of each flow, assuming relaying it – Dijkstra's algorithm is good, but heuristics such as the a-star algorithm are more efficient (Algorithm 9).

3. Share the above results with other gateways and decide the flow accommodation pattern.

While the gateway relocation algorithm performs a physical network reconstruction, the flow rerouting algorithm does a logical network reconstruction. By applying both algorithms in succession, the communication quality of the network can be optimized from both physical and logical perspectives.

This time, the relocation and the rerouting phase are designed as independent algorithms because the algorithm's computational complexity would be enormous if the two were combined. It is known that the target problem becomes NP-hard when generalized [96]. Thus, there is a fundamental computational difficulty for the drone's low-power CPU to calculate the global optimal positions for all gateways at once from the bird's-eye view telling all nodes' geolocations and flows' paths, even leaving aside how to realize such an eye.

Since the gateway relocation algorithm searches for the optimal solution for each gateway, it is but a local optimal after all. Even if it is combined with the flow rerouting algorithm, its performance may be inferior to that of the global optimal solution using the bird's eye. Sect 6.5 tests and discuss this point again.

Algorithm 8 Strategic Flow Rerouting

Input: Current network $G = (V, E)$ and Flow Summary set $\boldsymbol{Q}$
Output: Optimal g for flow $f_{s,d}$

```
1: function FindOptimal(G, Q, s, d)
2:     C ← {}
3:     for all g, Q ∈ Q do
4:         for all (s, d), c ∈ Q do
5:             C_{s,d} ← {c, g}
6:     P_{s,d} ← Minimize(C_{s,d})
7:     N ← 0
8:     for all g ∈ P_{s,d} do
9:         N_g ← N_g + 1
10:    P'_{s,d} ← ReverseSort(P_{s,d}, N)
11:    return P'_{s,d}.pop()
```

Algorithm 9 Flow Summary

Input: Current network $G = (V, E)$ and flow set F
Output: Flow Summary Q

```
1: function GenerateFlowSummary(G, g)
2:     Q ← {}
3:     for all f_{s,d} ∈ F do
4:         c ← A*(G, gateways, d, Euclid(G))
5:         Q ← {(s, d), c}
6:     return Q
```

6.2.4 Computational Complexity

For the number of nodes n_1 in G_A and the number of nodes n_2 in G_B, the computational costs of the node set Ω is equal to the total number of possible combinations of all nodes, as shown in Eq. 6.11.

$$\begin{aligned} {}_{n_1+n_2}\mathrm{C}_2 &= \frac{(n_1+n_2)!}{2(n_1+n_2-2)!} \\ &< (n_1+n_2)(n_1+n_2-1). \end{aligned} \tag{6.11}$$

Dijkstra's algorithm dominates the computational costs of the column vectors $\boldsymbol{a_n}$ and $\boldsymbol{b_n}$. Assume $k = n_1 = n_2$, 's number of accommodation flows m, and the final computational complexity becomes as in Eq. 6.12.

$$\begin{aligned} 2m(k^2+k^2) + 2k(2k-1) &= 4mk^2 + 4k^2 - 2k \\ &< O(mk^2). \end{aligned} \tag{6.12}$$

This is the average computational complexity of the search in Eq. 6.5 and at the

same time is comparable to the worst-case computational complexity of the search in Eq. 6.10 – the worst-case is the condition in Eq. 6.13.

$$\Omega \setminus \Phi = \emptyset,\ \forall (u,v) \in \Omega,\ \hat{C}_{(u,v)} \neq m. \tag{6.13}$$

The computational complexity of the flow rerouting algorithm is dominated by the shortest path searching for each flow. Dijkstra's algorithm yields $O(mk^2)$, which is similar to the relocation cost, but the best-first searching using a heuristic function such as a-star algorithm can lower the cost.

Typically, for $m = 30, k = 100$, the computation of $O(mk^2)$ is completed instantaneously, even in low-power processors. Therefore, the processing delay caused by the proposed algorithms is very small compared to the s' physical relocation time and can be ignored.

6.2.5 The Behavior of Algorithms

Finally, the actual behavior of the proposed algorithms is shown in Fig. 6.5. From left to right, these are the snapshots of flow status in the initial network, after the gateway relocation, and after the flow rerouting. The 20-node red/blue cluster is interconnected by 3 gateways represented as green nodes, with 6 inter-cluster communications occurring. Every communication flow has a 5-hop budget.

In the initial state, the total number of path hops for all flows was 31 – the average end-to-end path of 5.2 hops and the allowable hop satisfaction rate of 50%. After the relocation algorithm was performed, all three gateways moved to the optimal locations for all existing flows. As we can see in the figure, the algorithm achieved the eight hops reduction, the average path length decreased to 3.8 hops, and the satisfaction rate increased to 83%. Then, after the rerouting was carried out, the yellow flow moved from 1 to GW3, reducing 1 path-hop.

In this example, the proposed algorithm worked as designed, realizing a reduction of 9 path-hops and a 33% increase in the allowable hop satisfaction rate.

6.3 Cooperative Behavior of Distributed Gateways

This section discusses an implementation method of the algorithms proposed in the previous section in a distributed environment. The conceptual protocol design proves that the algorithms are deployable cooperatively and autonomously in actual heterogeneous drone swarms.

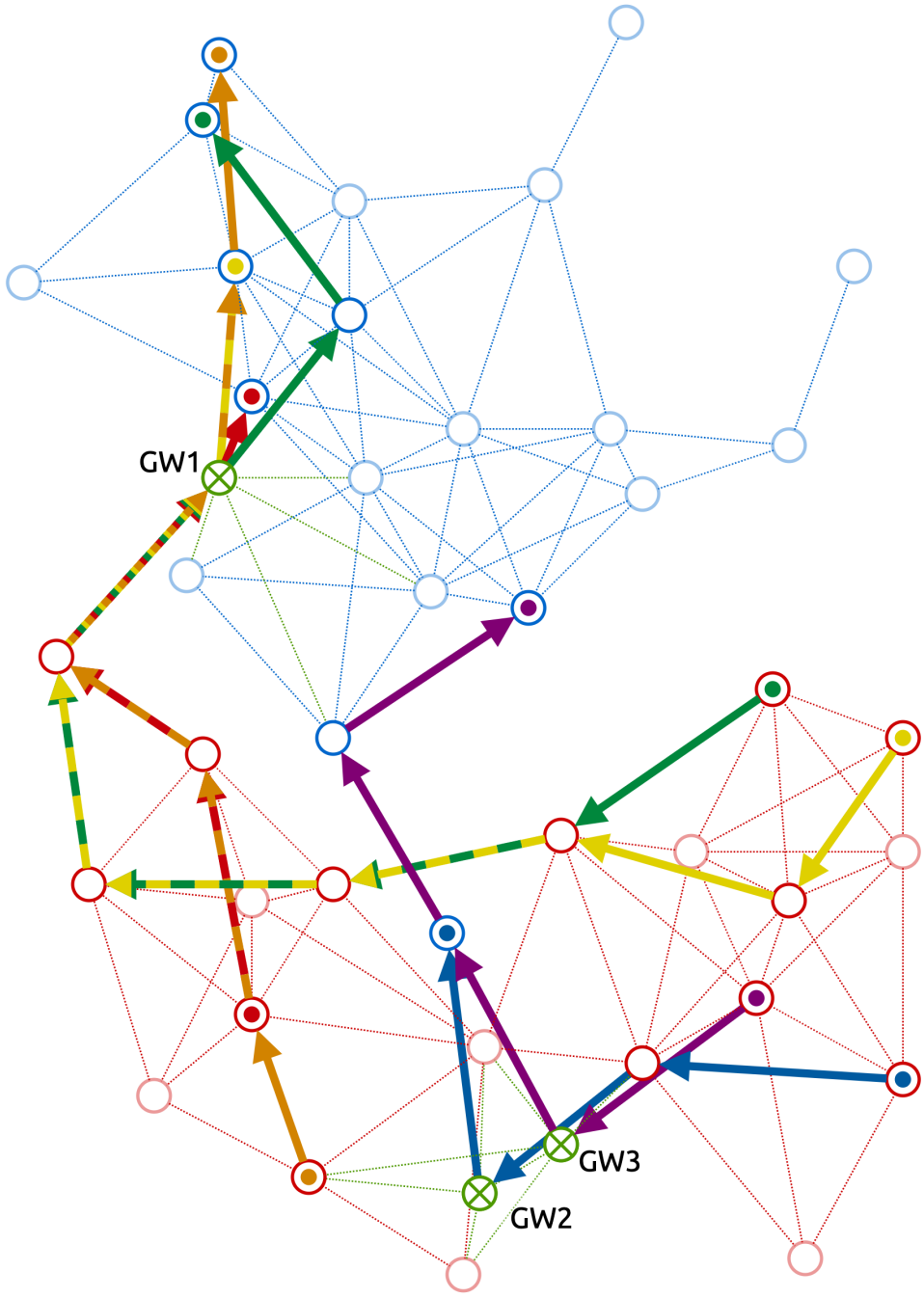
6.3.1 Protocol Overview

The basic concept of the previous protocol is to extend the HWMP (hybrid wireless mesh protocol) specification, which is standard in wireless mesh network construction, to achieve the following:

- Every gateway collects all nodes' geolocations proactively.
- Every gateway grasps a newly occurred communication reactively.

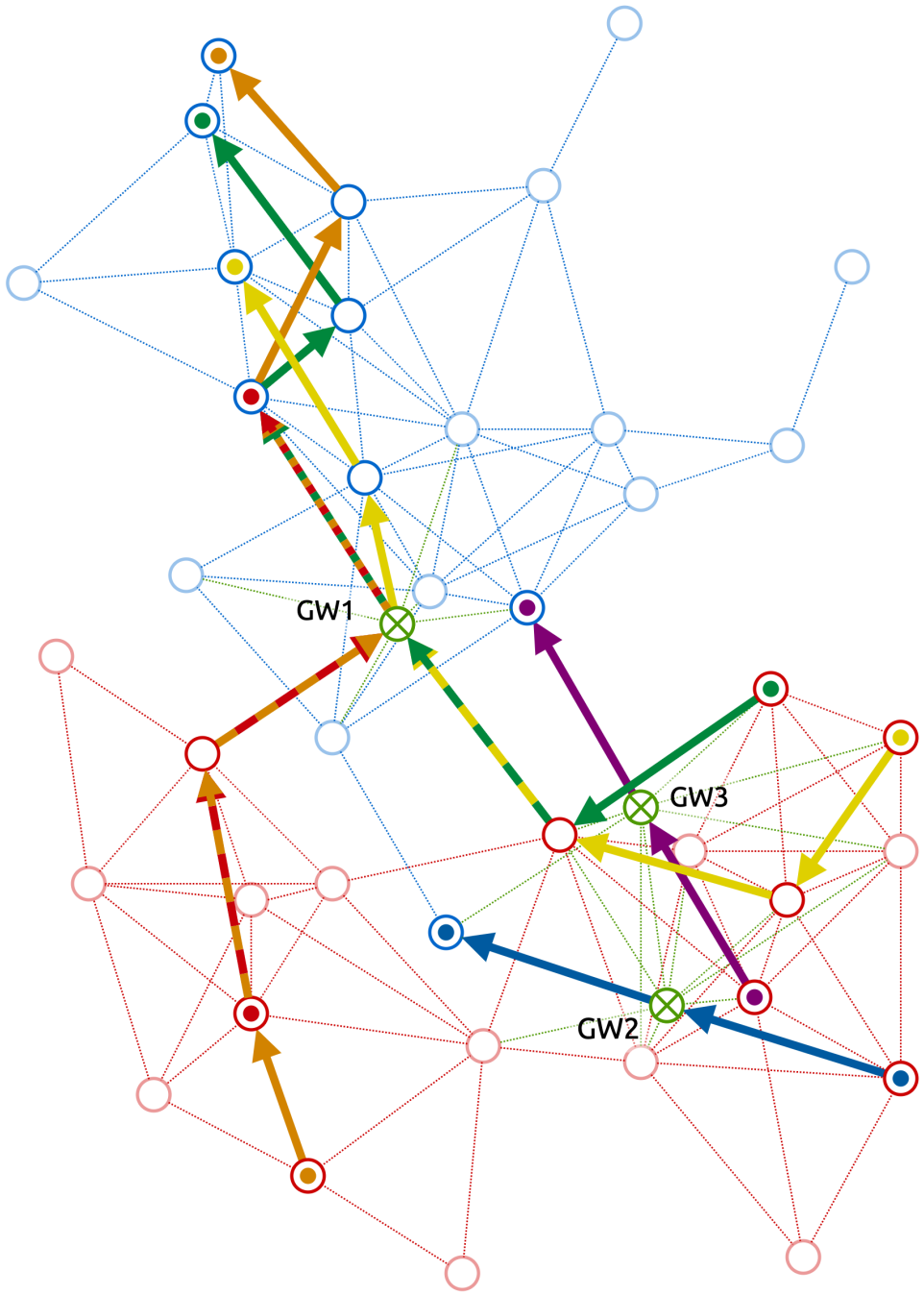
Every gateway collects all drones' geographic locations and estimates the network topology by storing location information in the RREQ (route request) message, which the source node broadcasts at the beginning of the communication. This time, all drones are assumed to be hovering at a high altitude over a nature conservation area as a concrete application scenario. In other words, the topology estimation from the geolocation is reasonable enough since it can be assumed that all neighbors are within the line-of-sight distance and no interference from other radio stations exists.

On the other hand, there are some cases where the topology cannot be inferred simply from the location information – the drones fly at low altitudes and cannot establish the LoS with their neighbors due to ground obstructions, or the radio quality is unstable due to noise or interference waves. In such cases, it is effective to understand



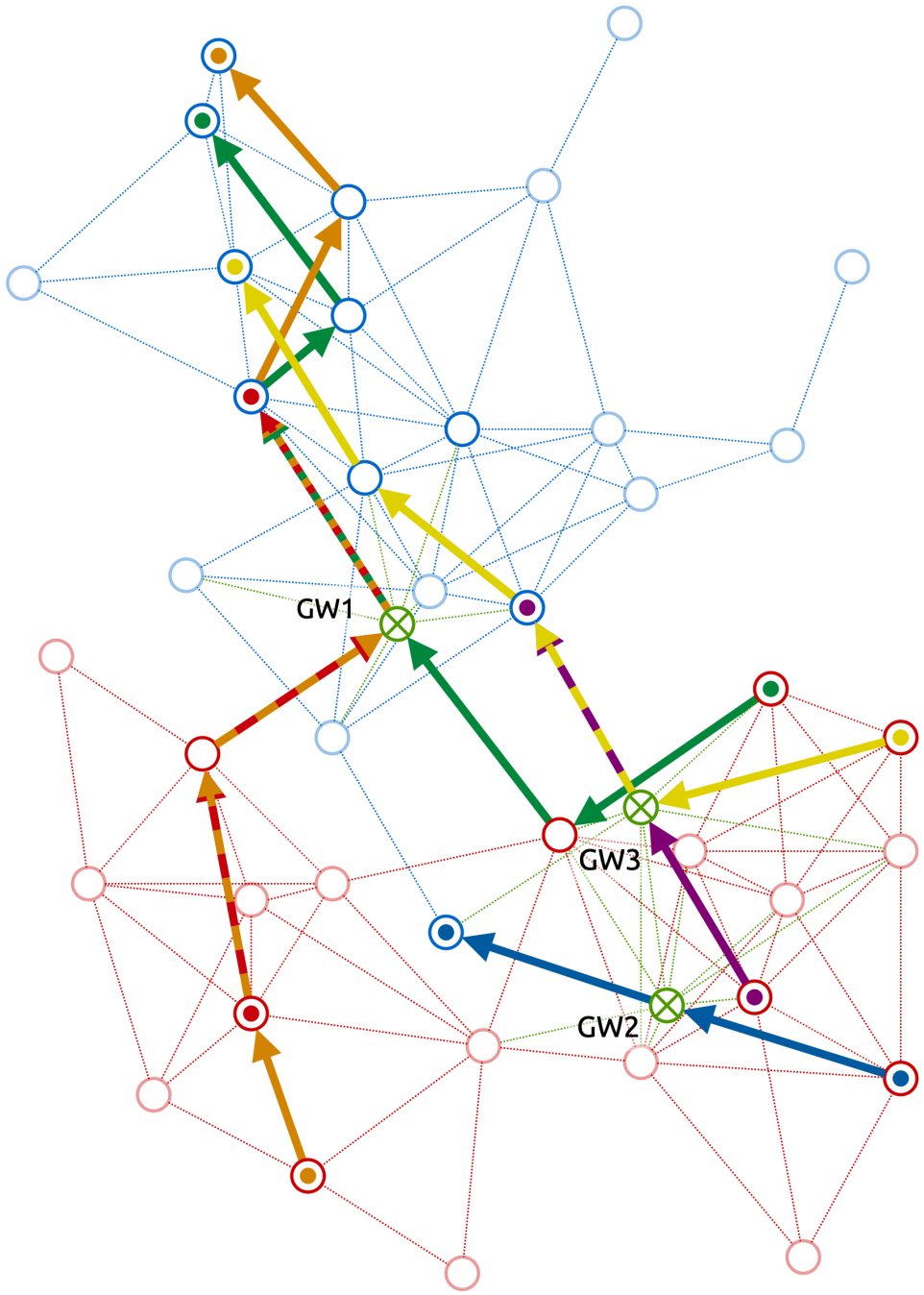
(a) Network snapshot of the initial state (i.e., test case).

Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur.



(b) Network snapshopt after the relocation performed.

Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur.



(c) Network snapshot after the rerouting performed.

Figure 6.5: An actual example of the proposed algorithm's behavior: 3 gateways interconnect two 20-nodes clusters, and 6 inter-cluster communications occur.

topology strictly by a link-state routing protocol such as OLSR. It is also practical to judge whether the link quality is sufficient for video streaming by using RSSI, packet loss ratio, and other indicators, and then disconnect the poor links intentionally.

There are no technical difficulties in incorporating the link-state mechanism into the proposed protocol. However, to avoid the complication of the discussion, this section do not deal with this mechanism but rather keep it to a simple conceptual design.

6.3.2 Definitions of New Message Types

The protocol requirement is to implement the information sharing needed to execute the relocation and rerouting algorithms and to pause and resume video sessions in a distributed manner. The following control messages are newly defined.

Relocation Request: A command sent by the operator to all gateways. s received this request enter the pending state, send the Suspend Notify, and then execute the relocation algorithm to move itself to the optimal position. This message is broadcasted on the assumption that the end-to-end path from the operator to each gateway is unknown.

Suspend Notify: This message is sent from every gateway to its ends; both source and destination nodes of each relaying flows. All nodes receiving it must suspend sending and receiving traffic. This mechanism prevents the network from being flooded with destination-unreachable video packets. Since all gateways maintain a route to the ends, this message is forwarded via unicast.

Resume Notify: Similar to Suspend Notify, it is sent to the ends by s; however, it allows ends to resume sending and receiving the suspended traffic. The message body includes the address pairs of the ends corresponding to the session identifier and the gateway address to be routed through. Every gateway does not necessarily grasp routes to the ends, so this message is forwarded by broadcast. It also serves as a backward route request and informs the end of the route to the gateway.

Relocation Reply: A reply message to Relocation Request. It is sent from each gateway to the originating operator and notifies the completion of relocation. The message body includes Flow Summary. Since the gateway after geographic movement does not know the route to the operator, so this message is broadcasted.

Flow Summary: A data structure maps a source-destination address pair, which is a flow identifier, to its flow's path cost. Every gateway calculates and builds for each flow that gateway is accommodating or will accommodate.

Rerouting Request: It is a command message sent by the operator to all gateways using broadcast. After receiving the request, gateway enters the pending state, sends Suspend Notify, and then executes the rerouting algorithm to reselect the optimal gateway for each flow.

Rerouting Reply: A reply message to Rerouting Reply notifying the completion of the rerouting. Since the topology itself does not change before and after the rerouting, every gateway keeps the route to the operator; thus, this message is forwarded via unicast.

Since HWMP supports reactive mode operation, reply messages to broadcasts can be unicast since the reverse path is already known. Limiting the use of broadcasts to the minimum necessary reduces the network load and completes sequences more quickly.

6.3.3 Expansion of the Protocol Sequence

The protocol sequence is extended as shown in Fig. 6.6 using the above newly defined messages. The specific behavior is as follows.

1. Once the operator issues a gateway relocation command, Relocation Request is broadcast to all gateways.

2. After 1 to GW3 receive the Relocation Request, they change the state from work-

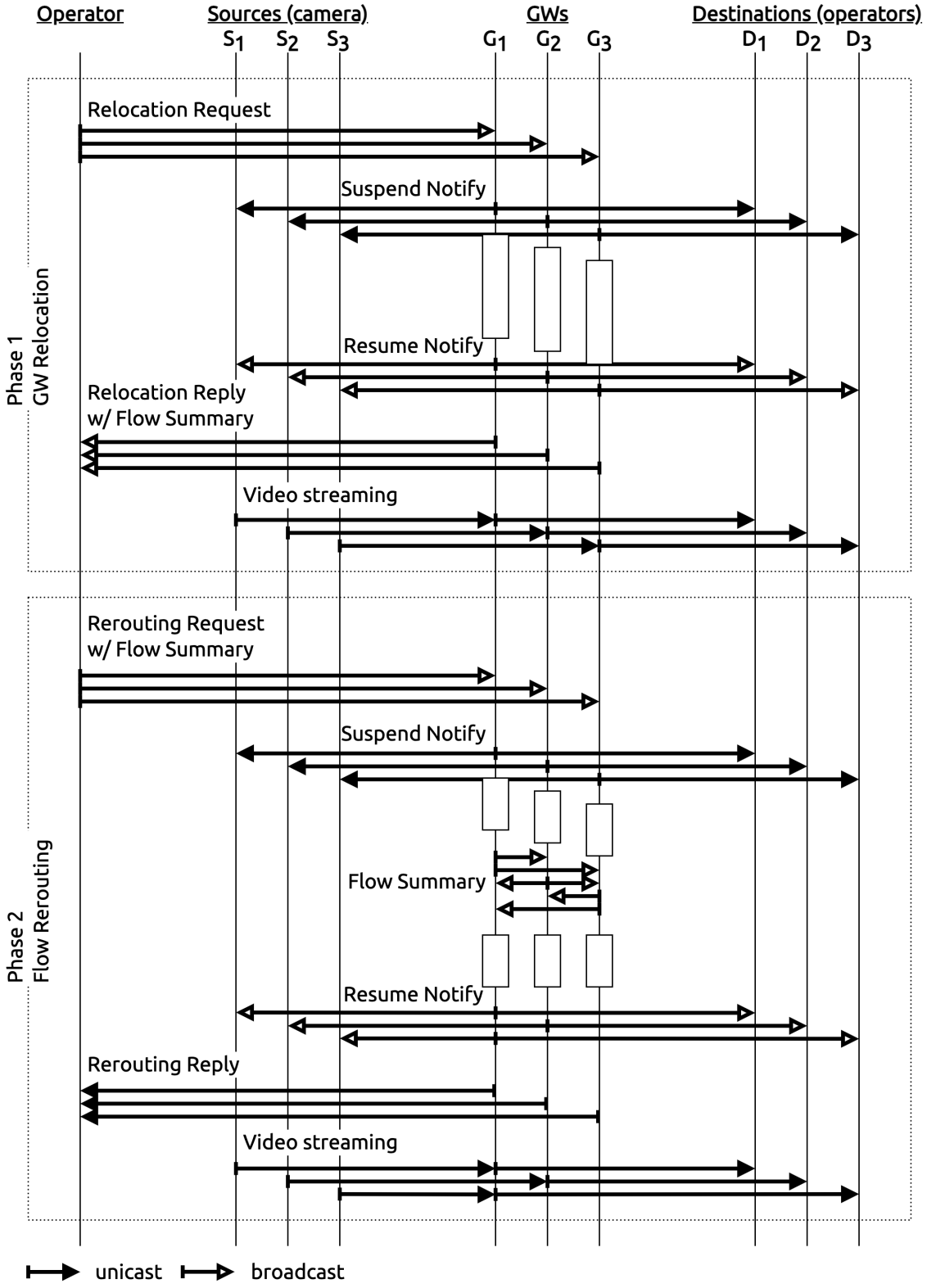


Figure 6.6: Control and data transmission sequence in the proposed protocol.

ing to pending and then send Suspend Notify to both ends of each relaying flow.

3. The source nodes receiving Suspend Notify will stop sending video traffic.

4. Each gateway executes the gateway relocation algorithm independently, calculates its own optimal location for the accommodated flows, and moves there. After moving, gateway sends Resume Notify to both ends of all flows and Relocation Reply to the operator, then transits its status to the working state.

5. The source nodes receiving the Resume Notify resume sending video traffic.

6. The operator receives Relocation Reply and confirms that the relocation was successfully completed. The video streams are recovered sequentially.

s whose Relocation Requests are unreachable for some reason do not transit to the pending state and continue to relay video traffic; however, they do not prevent other gateways from executing relocation. Besides, other exceptions that may occur at each stage of the relocation sequence can be self-healed by defining appropriate workarounds. Therefore, the above procedure provides the partition tolerance required in a real ad hoc network – an essential property in gateway relocation with topology changes.

For the flow rerouting, the operator performs the following procedures next.

7. Broadcasts the Rerouting Request embedding the Flow Summary extracted from the Relocation Reply previously received to all gateways.

8. 1 to GW3 receiving the Rerouting Request shift from the working state to the pending state in the same way as before, and then send the Suspend Notify to suppress video traffic and make the network quiet.

9. Each gateway extracts all flow information in the network from Flow Summary and calculates the path cost for all flows, assuming relaying it. Then, the gateway creates a new Flow Summary containing the calculation results and broadcasts it to other gateways.

10. The gateway that collects the Flow Summary from other gateways executes the flow rerouting algorithm to grasp which gateway should relay which flows. Each gateway sends the Resume Notify to both ends of the newly determined accommodating flows and the Rerouting Reply to the operator. If the gateway has one or more relaying flows, it moves to the working state; otherwise, it moves to the idling state.

11. The source nodes receiving the Resume Notify understand the newly configured path to the destination nodes and resume sending video traffic.

12. The operator receives Rerouting Reply and confirms that the rerouting was successfully completed. The video streams are recovered after that.

Even if there are some gateways whose Rerouting Requests were unreachable accidentally, the rerouting would be completed except for these gateways. In addition, the source nodes that fail to receive the Resume Notify will discard the session after a certain period and start over again by sending another Route Request.

With such a design, *Path Coordinator* can be integrated into real heterogeneous drone swarms as a consequence of the autonomous cooperative behavior of the gateway.

6.4 Preparation for Performance Evaluation

The next section analyzes the performance characteristics of the proposed algorithms using computer simulation with a wide range of parameter settings. This section illustrates the design of the experiments before discussing the results in Sect. 6.5.

6.4.1 Topology Models

It is assumed that only the gateways move geographically and topologically during the relocation process, while all other drones hover and wait in the air. In order to purely assess the performance of the relocation algorithm itself, the evaluation is carried out on

a static network; that is, the initial placements of all drones are reproduced using two graph models, Connected Random Geometric Graph and Manhattan Grid Topology.

If all nodes are constantly moving and the topology changes from moment to moment, the evaluation using an appropriate mobility model is required to consider the temporal continuity. In this case, what the important are mainly the following two points.

- **Availability:** gateway maintains its path reachability from any nodes as much as possible
- **High-precision estimation:** gateway understands the actual network geometry in real-time as precisely as possible.

The former was already dealth in the previous study and proposed a mobility control algorithm to enhance its availability. On the other hand, for the latter, it is entirely a study of topology estimation — for example, the accuracy of network geometry prediction based on geolocation information or the update interval auto-adjusting in link-state routing protocols. Any of them are independent issues from the relocation method itself, so they can be treated separately, and thus the evaluation experiments using static graphs are still worth a lot.

Connected Random Geometric Graph

RGG (Random Geometric Graph) is a graph generated by randomly placing nodes in a field and linking them if their geographic distance (euclidean distance) is less than or equal to n, where n is the communication radius, i.e., the maximum distance between nodes that can communicate at a bitrate above a certain level. This time, the graph consists of a single connected component with no isolated nodes or subclusters – Connected RGG. See the previous chapter for details of the generation method.

Suppose the wildlife in the nature conservation area are randomly scattered, then the camera drones capturing them from the high sky will also be randomly placed

eventually. Besides, since drones need to connect to other drones and join the network for video transmission, the final topology of the drone's network is Connected RGG.

Manhattan Grid Topology

There is a mobility model called Manhattan Mobility Model. As the name suggests, the model is based on the image of Manhattan in New York City, where nodes move along a grid-like street. This time, the node's mobility is ignored, so it can be treated as a simple graph model. In other words, generate a graph by placing nodes in a field along a grid with a geographic distance of n between them and linking the neighboring nodes – Manhattan Grid Topology. Figure 6.7 displays the sample topology. The generated graphs are all planar graphs, which have the advantage of easy theoretical analysis.

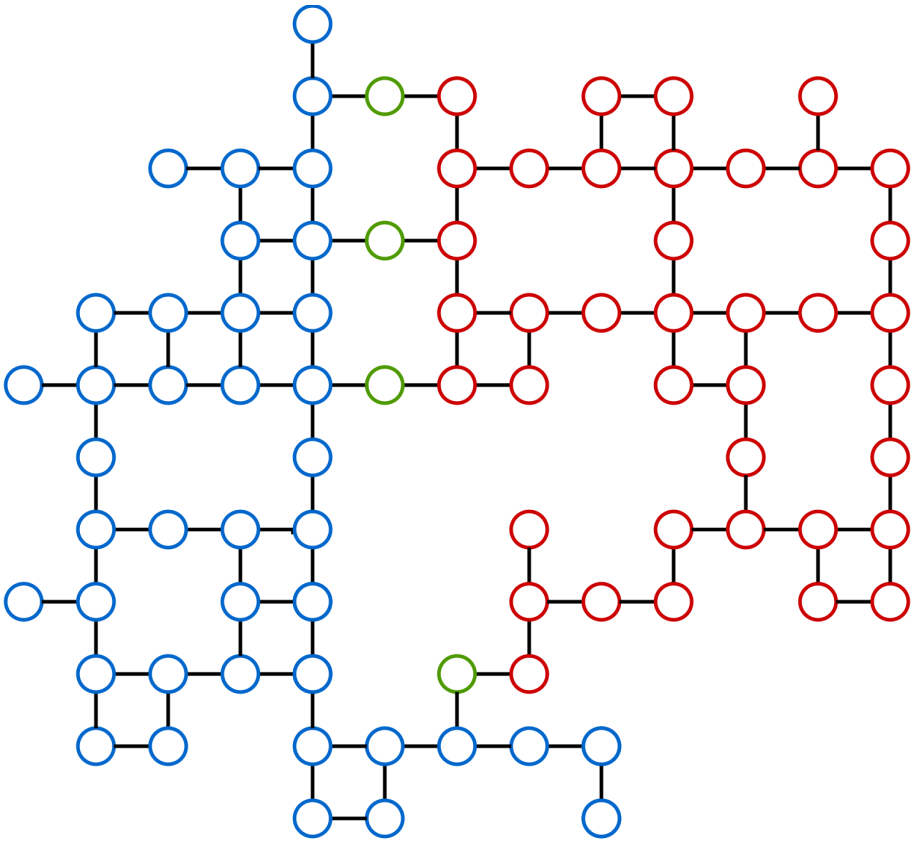


Figure 6.7: Example of Manhattan Grid Topology.

Suppose a drone flies at a low altitude and the LoS is blocked by buildings or other artificial objects, such as for monitoring critical facilities; communication may not be possible even if the geographical distance to the neighbor is within the communication range like the diagonal communication on Manhattan Grid Topology. When the effect of interference from other stations is negligible but obstacles hamper communication, the link quality becomes a binary between the best and the worst – the final topology is isomorphic to Manhattan Grid Topology. Of course, this situation is not included in the scenario assumed this time, but the computer simulation will investigate it for reference to discuss the algorithms' future extensibility.

6.4.2 Gateway Preplacement using Graph Centrality Measures

The proposed algorithm improves communication quality by adaptively relocating gateway locations corresponding to the flows occurring and disappearing dynamically. Here we have one hypothesis.

Hypothesis: Is it possible to determine the initial gateway placement in which gateways no longer need to move for communication occurrence and disappearance in terms of long-term average communication quality?

Following existing studies [97–100], a static gateway placement algorithm using graph centrality measures is designed to test the hypothesis. Algorithm 10 represents the pseudocode. It is a simple best-first search that preferentially bridges graphically essential nodes. By placing a gateway in the center of the graph, i.e., "a location with good accessibility," we expect the subsequently occurred flows to be distributed naturally to realize stable communication quality over a long period.

Tollowing seven centrality measures are adopted:

- **degree:** the higher the score, the higher the degree.
- **closeness:** the higher the score, the smaller the average shortest path to all other nodes.

- **eigenvector:** the higher the score, the higher the neighbors' scores.
- **katz:** The score with constat terms is added, so that the eigenvector centrality's value becomes nonzero.
- **stress:** the higher the score, the more times the node is included on the shortest path connecting the other two nodes.
- **betweenness:** the score normalized the stress centrality by the total number of shortest paths.
- **radiality:** the higher the score, the shorter the distance to reachable neighbors compared to its diameter.

If the computer simulation proves the hypothesis is correct, gateway placement can be done considering only the topology characteristics. In other words, for a nearly static network like drones hovering in the air, there is no need for complex autonomous gateway control mechanisms, and devising the initial gateway placement can ensure sufficient communication quality in the long term for practical use.

Algorithm 10 Preplacement using Graph Centrality Measures

Input: Two culusters $G_R = (V_R, E_R), G_B = (V_B, E_B)$, communication radius R, and number of gateways n_g
Output: Optimal placement P

```
1: function FindOptimalInitialPosition(G_R, G_B, R, n_g)
2:     C_R, C_B ← Centrality(G_R), Centrality(G_B)
3:     V'_R, V'_B ← ReverseSort(V_R, C_R), ReverseSort(V_B, C_B)
4:     P ← {}
5:     while |P| ≤ n_g do
6:         for all u ∈ V'_R, v ∈ V'_B do
7:             if ||u − v|| ≤ 2R then
8:                 P ← (u, v)/2
9:     return P
```

6.5 Performance Evaluation and Discussion

This section discusses the performance evaluation results of the proposed algorithms. The following four computer simulations were conducted to reveal the performance characteristics in detail:

1. Analysis of the path hop reduction effectiveness and gateway relocation cost of the proposed algorithms itself
2. Comparison with the static preplacement method using graph centrality measures
3. Comparison with the brute-force searching
4. Time series analysis of gateway state transitions

The common parameters of each experiment are shown in Table 6.1 – these values like the number of nodes or the communication radius are picked up with the assumption of the drones designed in Chapter 3. Moreover, the effective communication range when Wi-Fi is used outdoors is based on the report of Yuan et al. [32]. Note that only some parts of the graphs are presented in this paper due to the space limitation.

Table 6.1: Common parameters of test case generation.

Number of test cases per parameter set	200
Number of nodes per cluster	20, 30, 40
Number of clusters	2
Number of gateways	2-6
Number of communication flows	6, 8, 10, 12, 14, 16
Number of destinations	2, 3, 4, n
Communication radius	30, 50, 70 [m]

Table 6.2: Parameters for Connected RGG test cases.

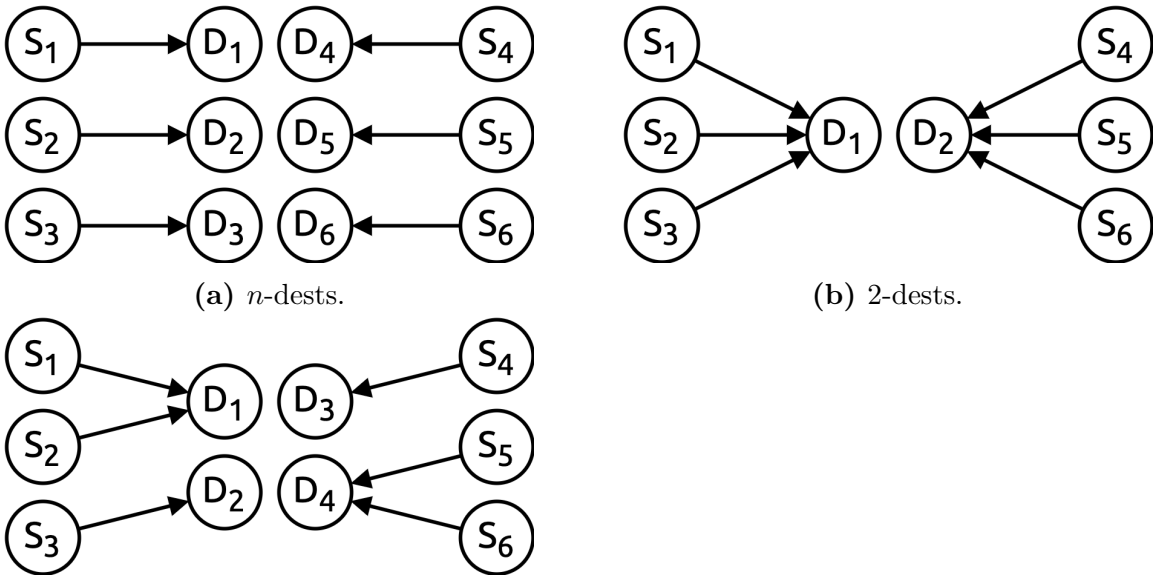
Field size (square field)	300 [m]
Budget of end-to-end hops	5-8 [hops]

Table 6.3: Parameters for Manhattan Grid Topology test cases.

Field size (square field)	1000 [m]
Budget of end-to-end hops	4-6 [hops]

6.5.1 Relocation and Rerouting

First, the computer statistically investigated the effect of path hop reduction for the relocation and rerouting algorithms concerning the number of end-to-end hops for each flow in the three states of the original, after relocation, and after rerouting. The topology models are the two models introduced in the previous section, and the evaluation parameters are shown in Table 6.2 and Table 6.3. 200 evaluations are conducted for one parameter set in this experiment and statistically processed the obtained 200 results. In other words, the computer evaluated a total of 1134000 test cases by generating 200 topologies for each of the 5670 parameter sets with combinations of number of nodes, number of gateways, number of flows, communication radius, etc.



(a) n-dests. **(b)** 2-dests.

(c) 4-dests.

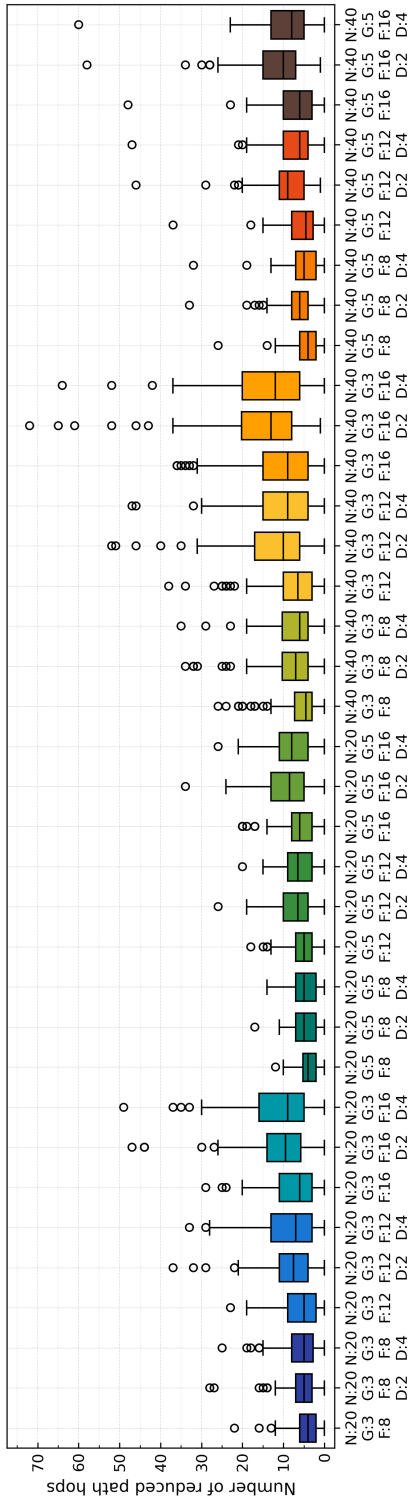
Figure 6.8: Flow generation pattern for test cases.

In addition to the earlier explanation, there is one supplement. In the case of wildlife monitoring, which is the target scenario of this study, the following situations can be considered: ***1)*** There are several campsites in a nature conservation area, and ***2)*** Operators can watch multiple remote videos simultaneously from their campsites by connecting to the HANETs. Therefore, test cases are generated so that the n flow consists of n source nodes (camera drone-side) and m destination nodes (operator-side) and observe the performance characteristics under varied m – called as the m-dests pattern. Note that when the n-dests is mentioned simply without a specific n, there is a one-to-one correspondence between the source and the destination (Fig. 6.8).

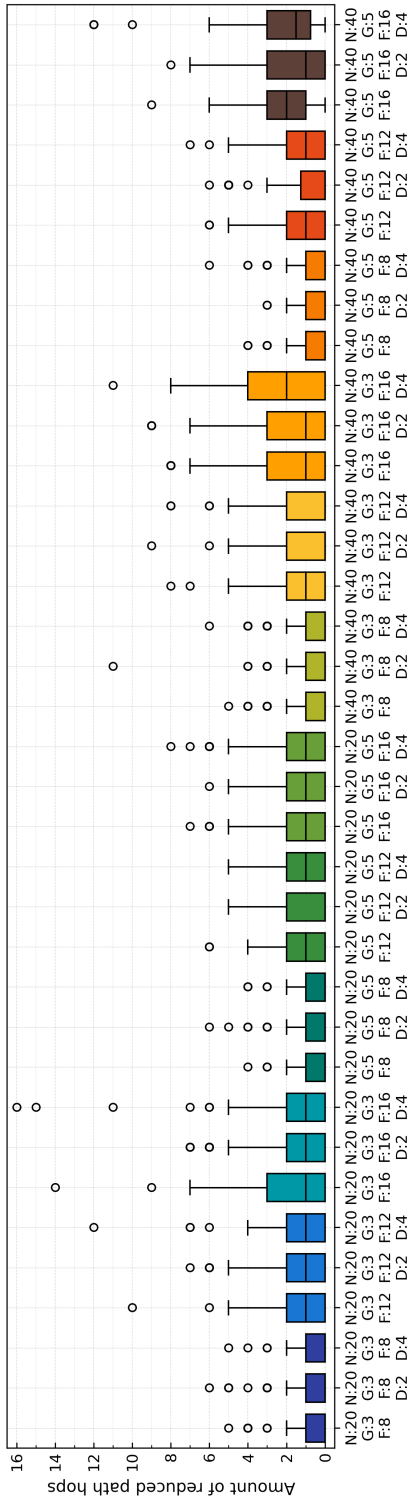
The box plots displayed in Figs. 6.9a-6.9b illustrate the improvement of each flow's end-to-end path-hops by the relocation and rerouting algorithms. The graph model is Connected RGG. The horizontal axis shows the conditions, and the vertical axis shows the total number of reduced hops before and after the algorithm application. A positive

value on the vertical axis indicates an overall improvement in network communication quality. Note that each test case has a different topology, and thus every test case has a different average end-to-end distance for inter-cluster communication. Since the values on the vertical axis are not normalized, it does not make sense to compare the numerical difference among several conditions; this graph aims to check the trend of performance characteristics. The abbreviations of the horizontal labels are defined as follows: N is the number of nodes, G is the number of gateways, F is the number of flows, and D is the number of destinations (n-dests for D unexpressed).

Figure 6.9a proves that the relocation algorithm achieved hop reduction under all evaluation conditions. The reduction effect tends to be more significant for 2-dests and 4-dests than for n-dests flow patterns. Figure 6.9b shows that the rerouting algorithm also reduces the overall number of hops, but on average, it is not as good as the relocation. As can be qualitatively predicted, the larger the number of flows, the greater the improvement effect of rerouting. Note that the maximum value is more prominent than the minimum value in all conditions. For example, in 20 nodes, 3 gateways, 12 flows, and 4-dests environment, if the relocation algorithm is performed 100 times, the average number of reduced hops is 8.8, including 9 cases of 20 or more hops reduced – 20, 20, 21, 23, 24, 25, 28, 29, and 33 hops. Because neither the relocation nor the rerouting will ever make the situation worse, operators are recommended to call these methods as appropriate to reorganize the flows when the network becomes congested.



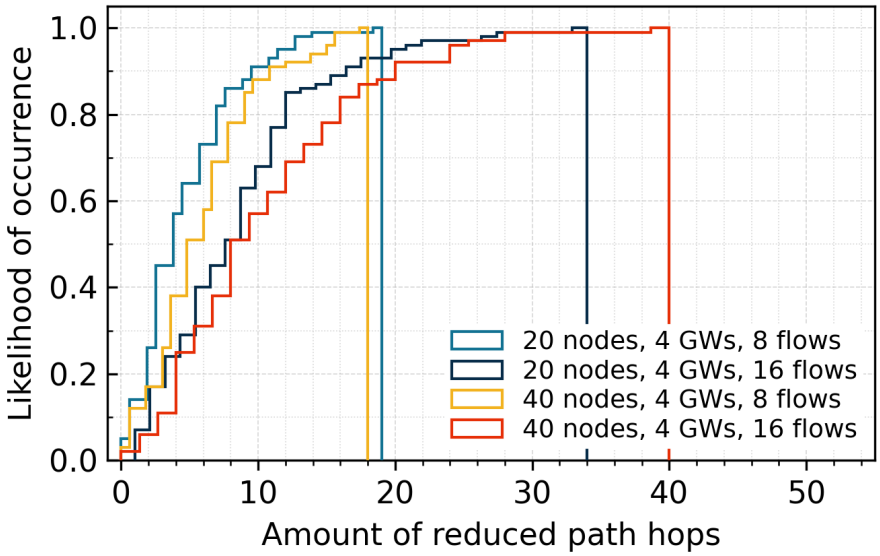
(a) Amount of reduced path hops when the relocation algorithm is applied to the original network.



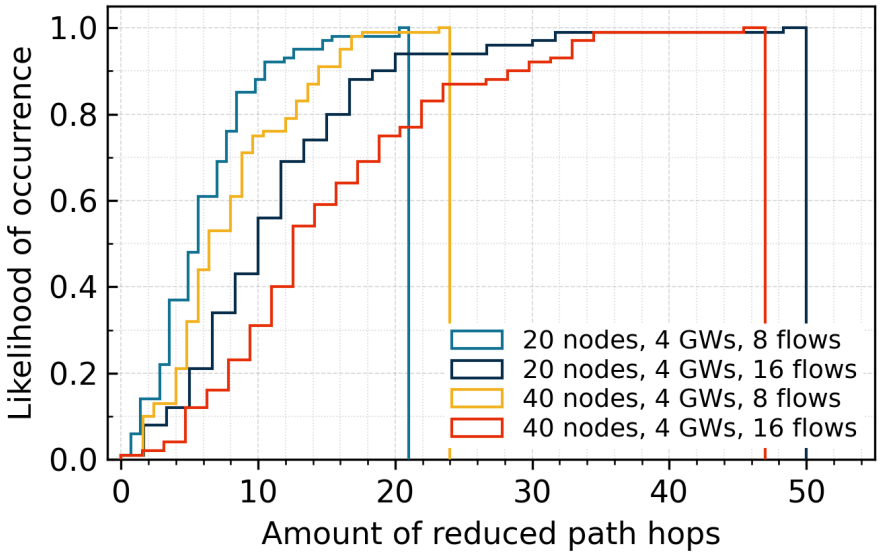
(b) Amount of further reduced path hops when rerouting algorithm is applied to the above relocated network.

Figure 6.9: Relocation and rerouting: Comparison of the total amount of reduced path hops under various conditions.

Figures 6.10-6.13 represent *Path Coordinator*'s reduced path hops and gateway traveling distance in ECDF for Connected RGG and Manhattan Grid Topology. Using *Path Coordinator* means that the rerouting is executed after the relocation – the full performance of the proposal. The horizontal axis is the number of hop reductions, the total and maximum gateway traveling distance, and the vertical axis is the probability of occurrence. The comparison between Fig. 6.10 and Fig. 6.11 confirms that, on average, even in Manhattan Grid Topology, the algorithm achieves an improvement equivalent to that of Connected RGG. In Manhattan Grid Topology, as the number of nodes and flows increases, the upper outliers of reduced hop counts also increase. In other words, when the network is congested, the communication quality is highly degraded compared to Connected RGG due to the topological characteristics, and thus the proposed method works remarkably well. Figures. 6.12-6.13 show that the total traveling distance of the gateways is, on average, between 100m and 150m, and as the shape of the graph indicates, the overall average performance characteristics are independent of topology and flow generation pattern. In the case of Manhattan Grid Topology, we can see a few cases (less than 5%) where the total traveling distance increases significantly. These correspond to the scene of radical performance improvement mentioned earlier.

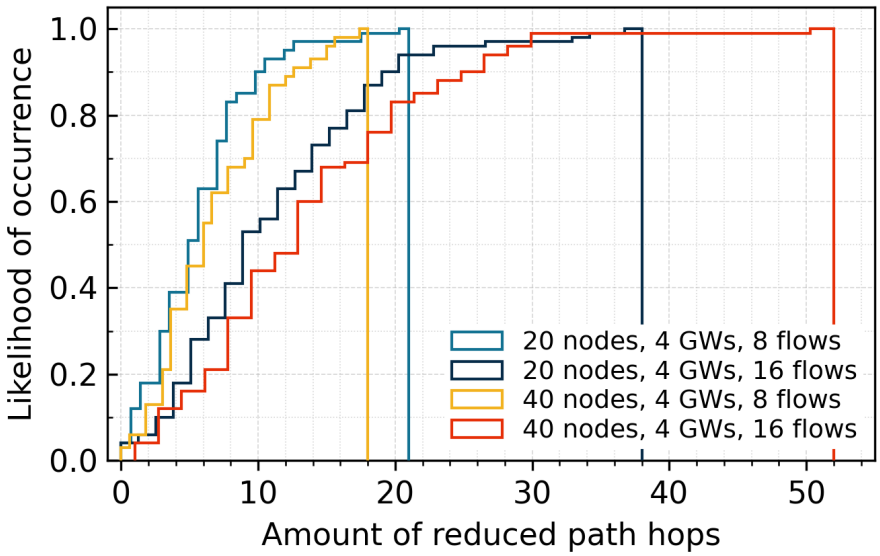


(a) Flow pattern: n-dests.



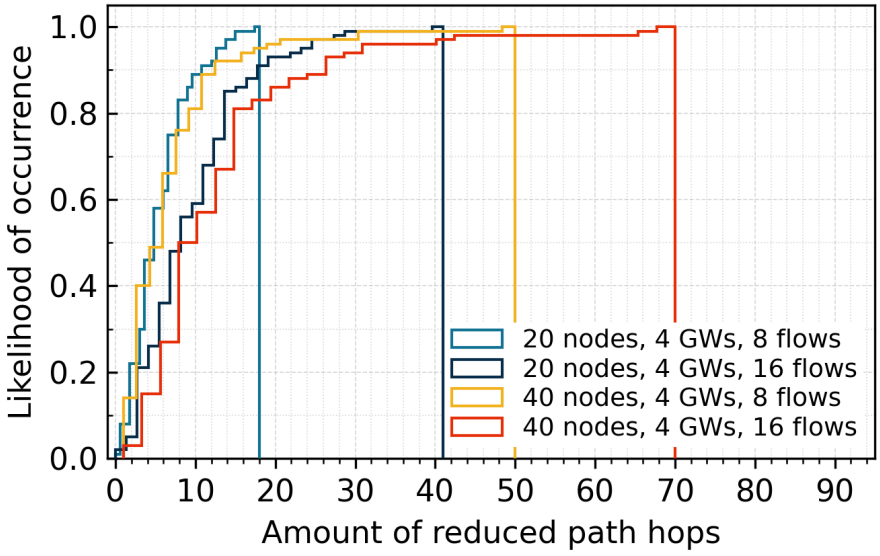
(b) Flow pattern: 2-dests.

Figure 6.10: ECDF: Amount of reduced path hops by *Path Coordinator* under Connected RGG network.



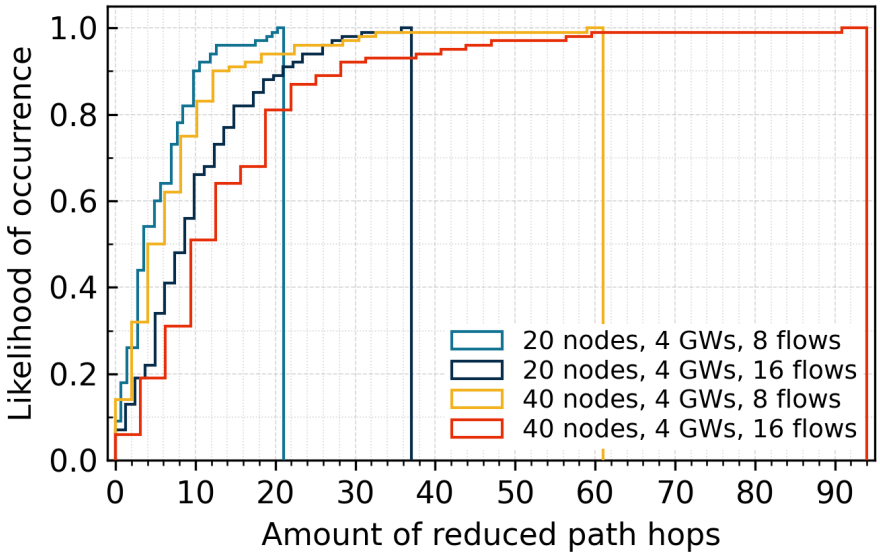
(c) Flow pattern: 4-dests.

Figure 6.10: ECDF: Amount of reduced path hops by *Path Coordinator* under Connected RGG network.

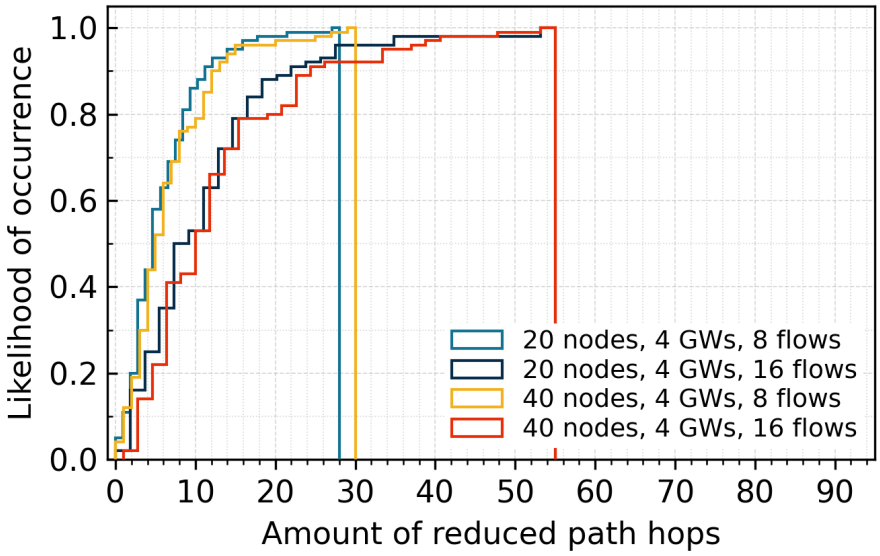


(a) Flow pattern: n-dests.

Figure 6.11: ECDF: Amount of reduced path hops by *Path Coordinator* under Manhattan Grid Topology network.

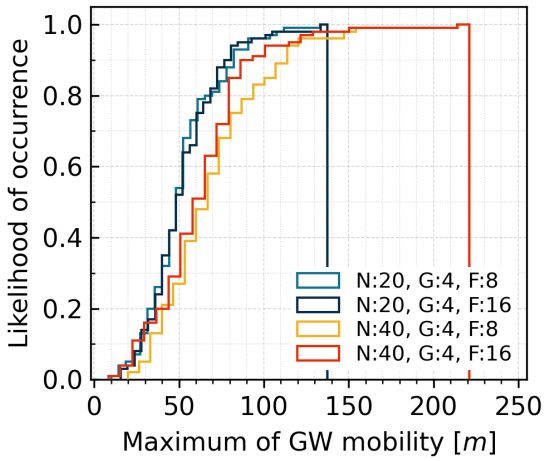


(b) Flow pattern: 2-dests.

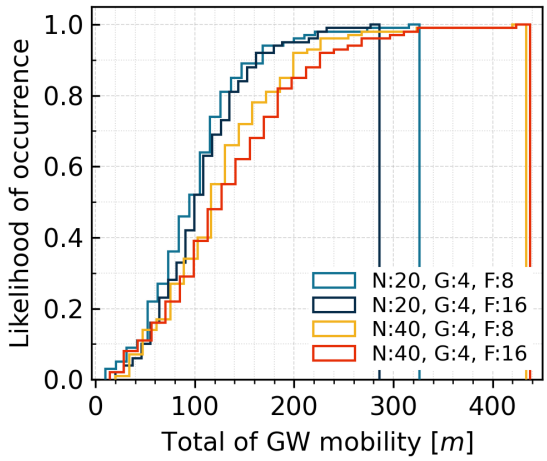


(c) Flow pattern: 4-dests.

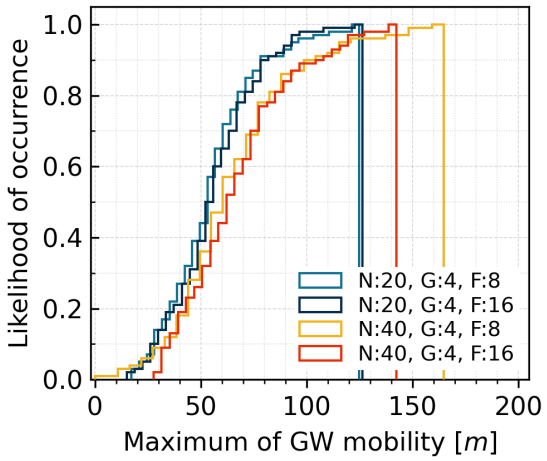
Figure 6.11: ECDF: Amount of reduced path hops by *Path Coordinator* under Manhattan Grid Topology network.



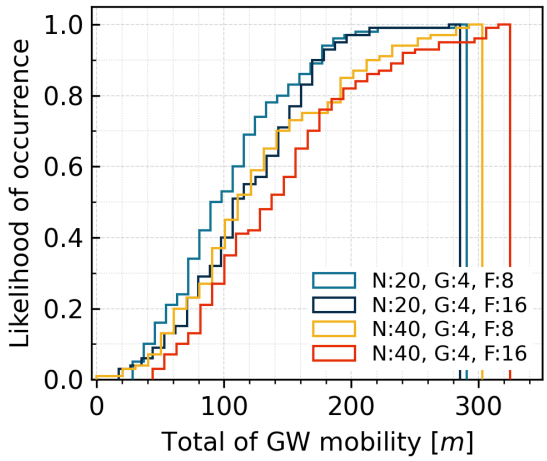
(a) Flow pattern: n-dests.



(b) Flow pattern: n-dests.

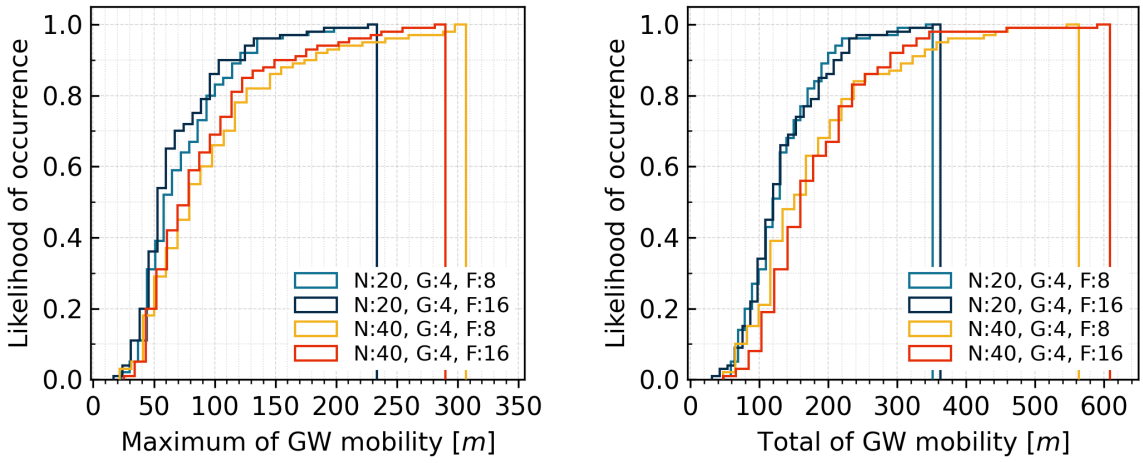


(c) Flow pattern: 4-dests.



(d) Flow pattern: 4-dests.

Figure 6.12: ECDF: Amount of gateway movement occurred by *Path Coordinator* under Connected RGG network.



(a) Flow pattern: n-dests.

(b) Flow pattern: n-dests.

(c) Flow pattern: 4-dests.

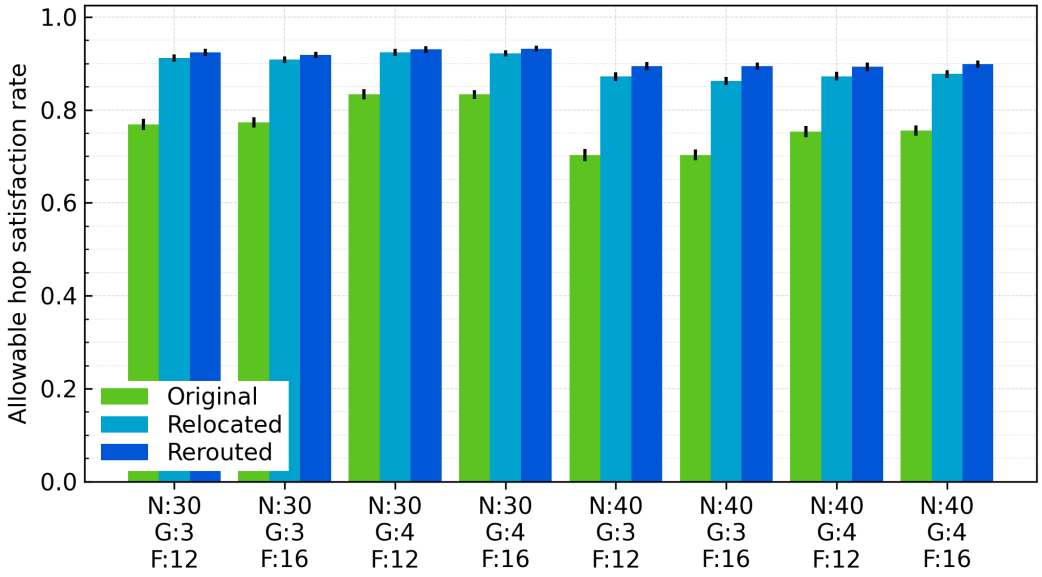
(d) Flow pattern: 4-dests.

Figure 6.13: ECDF: Amount of gateway movement occurred by *Path Coordinator* under Manhattan Grid Topology network.

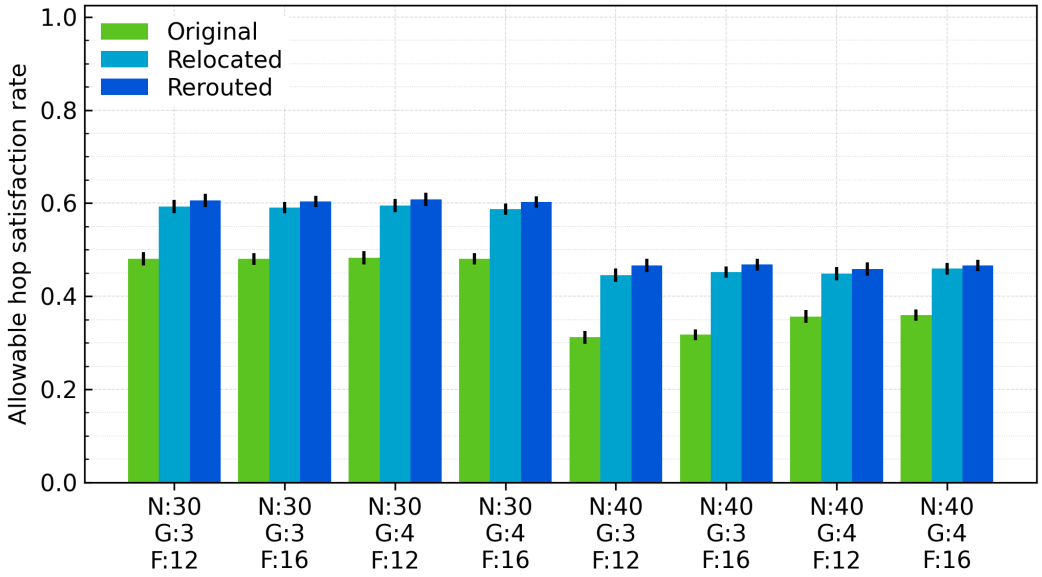
In the discussion of Figs. 6.9a-6.9b, the operator should call *Path Coordinator* as needed when the congestion occurs, and the cost can be estimated as follows. For example, in 20 nodes, 4 gateways, 8 flows, and 4-dests environment, the average amount of total gateway's traveling distance is 107.0 meters, which is 26.7 meters per gateway. The maximum traveling distance per gateway is 55.8 meters, and the worst is 124.8 meters. Therefore, in terms of the assumed drone hardware specs, the average relocation time is 11.2 seconds, and the worst is 25.0 seconds. If this level of downtime is acceptable, it is better to run the relocation actively. As a side note, it is known that flight consumes less power than hovering if the aircraft's aerodynamic design is optimized for forwarding flight. Therefore, the only disadvantage of the relocation is the gateway downtime, and there is no need to worry about reducing the remaining flight time due to the gateway's massive movement.

As described in Sect. 6.2, the relocation algorithm can improve and maintain the allowable hop satisfaction rate in best-effort – Fig. 6.14 presents the result. In the parameter range of Fig. 6.14, the proposed method improves the satisfaction rate by a minimum of 9.7%, a maximum of 19.1% for Connected RGG, and a minimum of 10.1%, a maximum of 15.4% for Manhattan Grid Topology. These results, which show the enhanced satisfaction rate by the proposed algorithm, confirm that the original purpose of this study, i.e., overall improvement in video quality, can be achieved.

Figure. 6.15 shows the number of idle gateways before and after the algorithms run. The relocation algorithm only moves gateways and does not update flows' accommodating gateways, so the number of idle gateways does not change. The graph shows that the rerouting algorithm significantly improves the number of idle gateways in both Connected RGG and Manhattan Grid Topology – it succeeded in concentrating flows to specific gateways and creating idle gateways. The larger the number of idle gateways, the more advantageous it is to create shortcuts to newly generated flows with *Path Optimizer*. It was the purpose of designing the rerouting algorithm, and the experimental results confirmed that it was realized as expected.

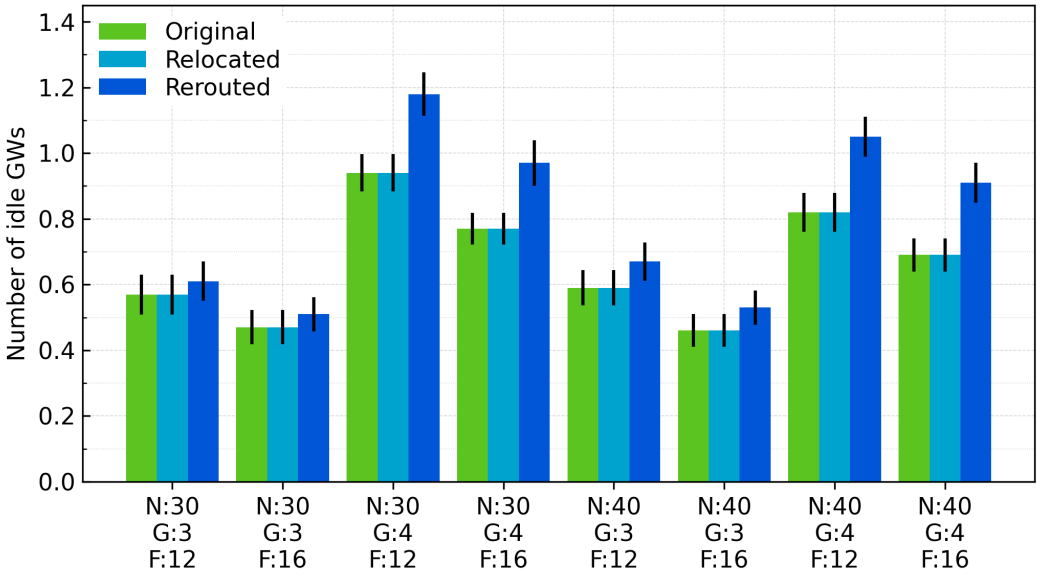


(a) Graph: Connected RGG.

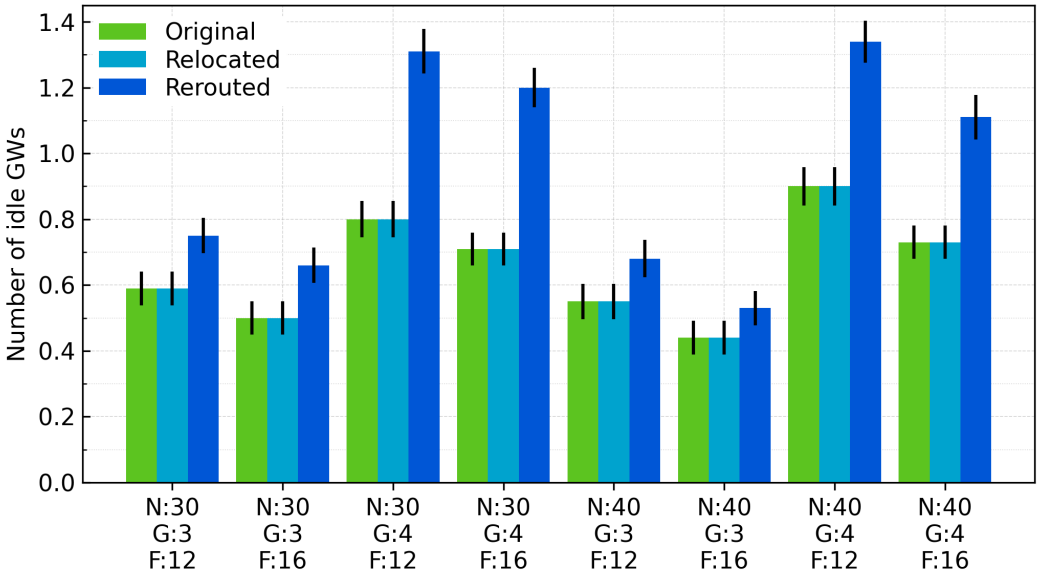


(b) Graph: Manhattan Grid Topology.

Figure 6.14: Allowable hop satisfaction rate: Improvement of the satisfaction rate under several conditions.



(a) Graph: Connected RGG.



(b) Graph: Manhattan Grid Topology.

Figure 6.15: Idle gateways: Improvement of the number of idling gateways under several conditions.

6.5.2 Strategic Preplacement vs. Adaptive Relocation

Second, the computer tests the hypothesis raised in the previous section – Is it possible to maintain the high communication quality over a long period by devising the initial gateway placement?

The parameter settings for this experiment are the same as in the previous experiment, using Table 6.2 and Table 6.3. However, whereas the previous experiment generated 200 topologies for each parameter set and examined the performance distribution, this experiment generated 200 flows in a single topology. It is like the following: ***1)*** One topology is generated for each of the 1890 possible combinations of parameters such as the number of nodes, number of gateways, and communication radius. Note that the graph is partitioned into two independent connected components at this point. ***2)*** Algorithm 10 determines all s' placement statically and connects the two networks into one. ***3)*** Prepare 200 flow patterns for each topology, and get a total of 37800 test cases. The flow patterns are obtained by randomly picking up 18 sources and their corresponding 18 destinations in the network. ***4)*** Perform the dynamic gateway relocation by the proposed method for each test case.

This experiment does not aim to investigate how good Algorithm 10 itself is. The purpose is to see how much difference in quality is between the strategic static and adaptive dynamic placement. It is necessary to make sure that the evaluation is not limited to the cases where all flows happen to occur at convenient sections for the initial gateway locations; hence, 200 flow patterns are generated for one topology.

The box plot in Fig. 6.16 presents the result. The graph model is Connected RGG, the horizontal axis is the test case condition, and the vertical axis is the number of total reduced path hops before and after the proposed algorithm application. Each box consists of 200 samples. As mentioned above, it is meaningless to compare the numerical values of Fig. 6.16 themselves. The purpose of the experiment is not to compare the performance score among centrality measures.

If the hypothesis in Sect. 6.4 is correct, the proposed method's hop count reduction effect should become smaller than in the previous experiment (Fig. 6.9a). The average

path hops for each communication should remain small without dynamic relocation due to the intelligent preplacement algorithm. However, the actual results reveal that the performance of the proposed method maintains high scores, not much different from that of the results, although it depends to some extent on the topology. It suggests that even if we devise the initial placement using the graph centrality measures, it is not as good as the dynamic relocation. In short, the hypothesis was rejected.

Of course, there is a possibility that there exists a gateway placement optimization method that performs better than Algorithm 10; still, this investigation is not the main topic of this paper, so the author leave it to other researchers. It is true that the proposal is superior to the naive preplacement, which supports the effectiveness of the dynamic relocation approach.

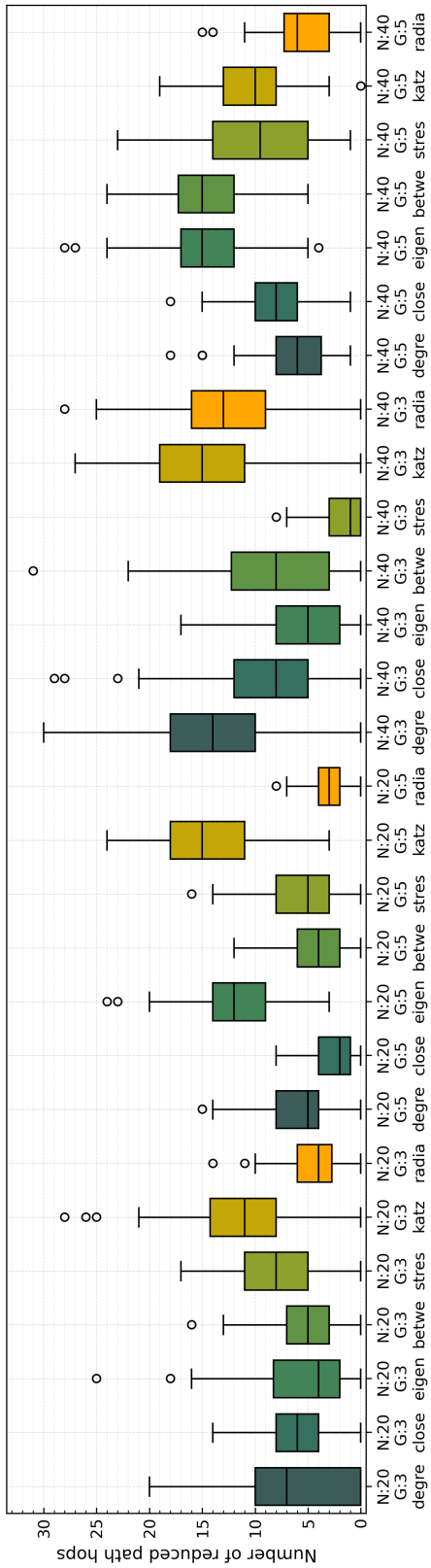


Figure 6.16: Preplacement performance: Reduced hops when *Path Coordinator* is applied in the preplaced network under various conditions.

6.5.3 Proposed vs. Brute-force

Next, the computer compares the performance of the proposed algorithm with the upper limit found by brute-force search. The brute-force search verifies all possible locations for gateway placement – there are possibly ${}_{n_1 n_2}\mathrm{C}_m$ placement patterns at most for the number of nodes of each cluster n_1, n_2, and the number of gateways m. For example, when the network size is 20 nodes and 3 gateways, the number of evaluations is at most 1.06×10^7; when 40 nodes and 5 gateways, then 8.68×10^{13} evaluations. It is an unrealistic computational cost for modern computers. Therefore, the paths along which the flows pass are focused and excluded from the evaluation target of those impossible gateway placement candidates to reduce the computation cost; in a nutshell, the pruning strategy was adopted. The pruning strategy is effective only when the number of flows is tiny, and its computational cost depends on the flows' paths and is hard to generalize. The evaluation range of this experiment is shown in Table 6.4.

Table 6.4: Parameters for the brute-force searching.

Graph model	Connected RGG
Number of nodes per cluster	20, 30
Number of gateways	3, 4
Number of communication flows	6
Number of destinations	4, n-dests
Budget of end-to-end hops	5 [hops]

Figure 6.17a shows the distribution of the path hop improvement under each condition by brute-force search, and Fig. 6.17b shows the distribution of the performance difference with the proposed method under the same conditions. The horizontal axis is the test case condition, and the vertical axis is the total number of reduced path hops, where each box consists of 200 samples. In all conditions, the performance difference is

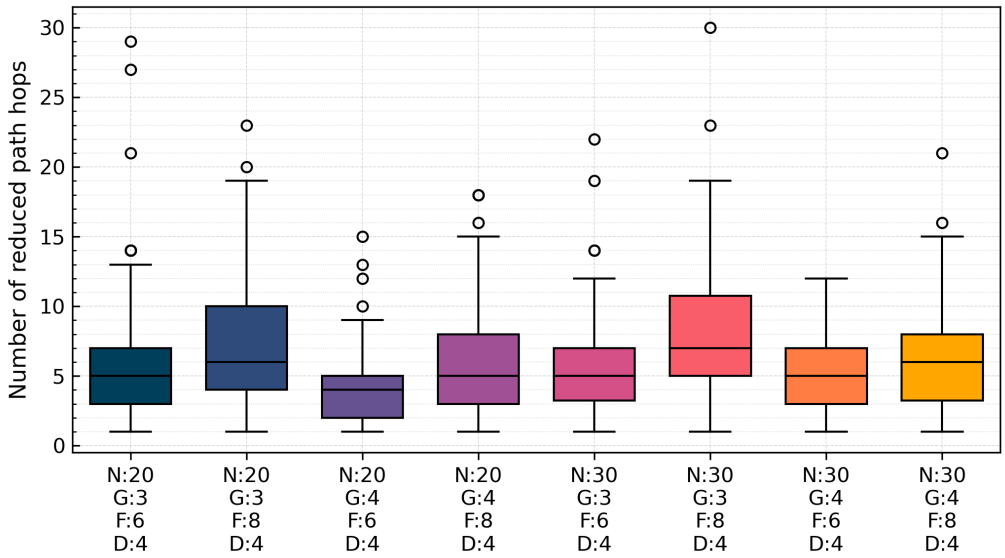
within two hops in 75% of the samples, and the average number of reduced hops is at most one hop.

Figure 6.17c indicates the flow's average end-to-end hop distance. Although the relocation using brute-force search is significantly smaller than that of the proposed method, the difference is only about 0.2 hops, which is negligible in practice.

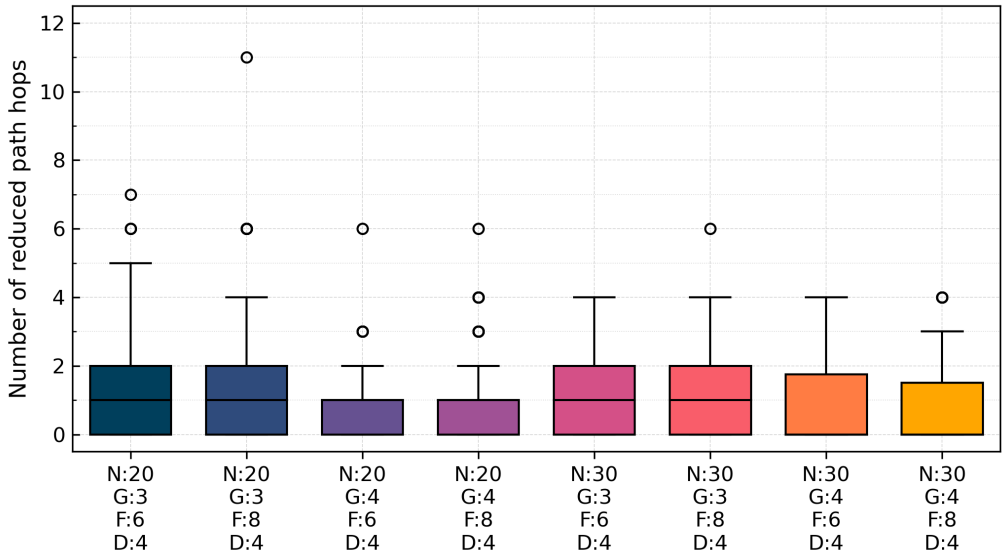
Figure 6.17d shows the average gateway traveling distance due to the relocation. In the proposed method, each gateway searches for an optimal placement only for flows passing through it, thus reducing the distance from 5 to 15 meters compared to the brute-force.

Figure 6.17e represents the number of idle gateways after the relocation. We can say that the idle gateway generating performance of the proposed method is comparable to that of the brute-force. Note that the placement found by brute-force is to maximize the total number of reduced hops, not to maximize the number of idle gateways.

Relocation using brute-force search is only possible a high-performance computer at the ground station, such as an operator's PC, can ***1)*** withstand the enormous computational cost, ***2)*** realize the bird's-eye view of the network situation, grasping topology, nodes' geolocations, and paths of all video sessions, ***3)*** and order any drone to move to any location. The overall performance of the proposed method is comparable to that of the brute-force. Furthermore, the computational cost is light enough to be handled by a low-power processor, and it works in an autonomous decentralized environment. The superiority of the proposed method is obvious.

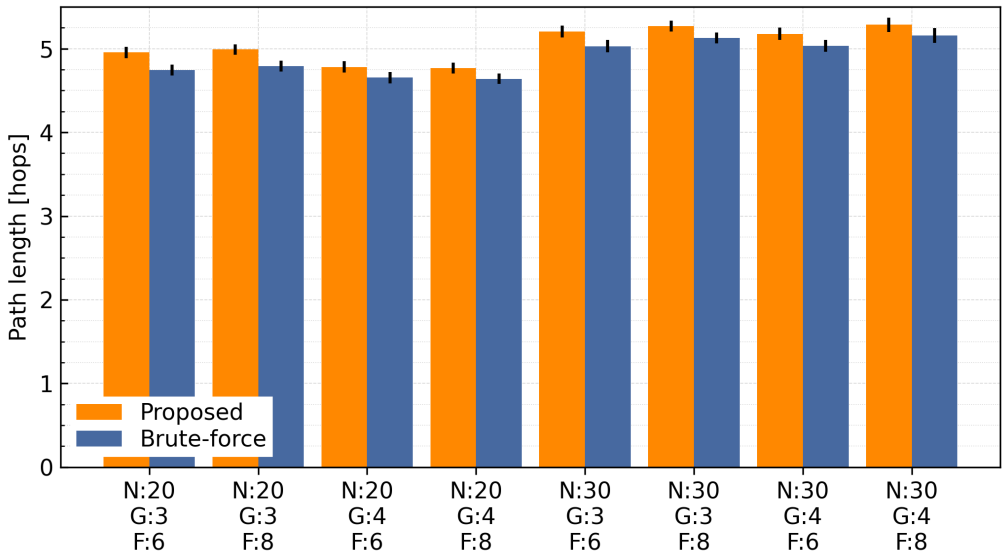


(a) Amount of reduced hops by the brute-force searching.

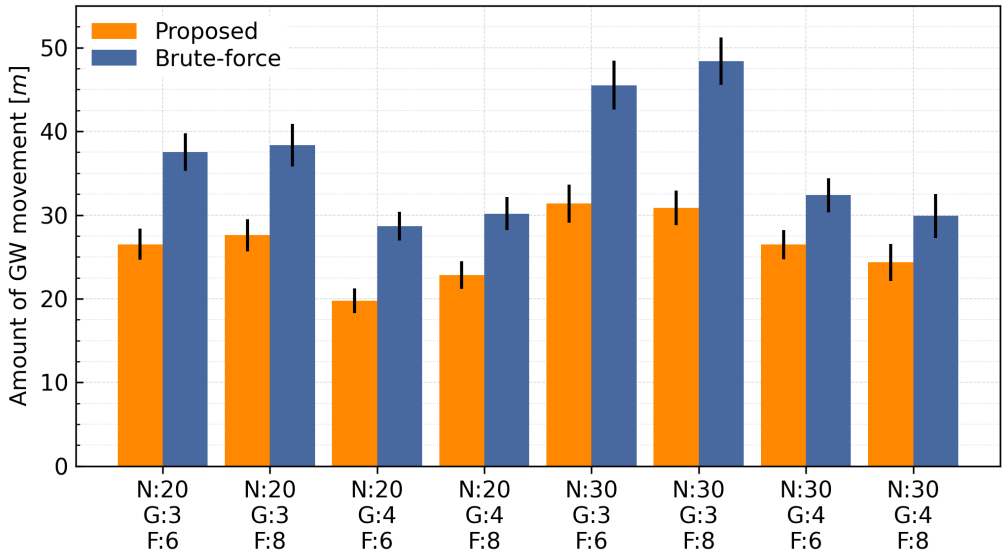


(b) Difference between the upper limit and *Path Coordinator*.

Figure 6.17: Performance upper limit: Limit of the path hop reduction and the comparison with *Path Coordinator*.

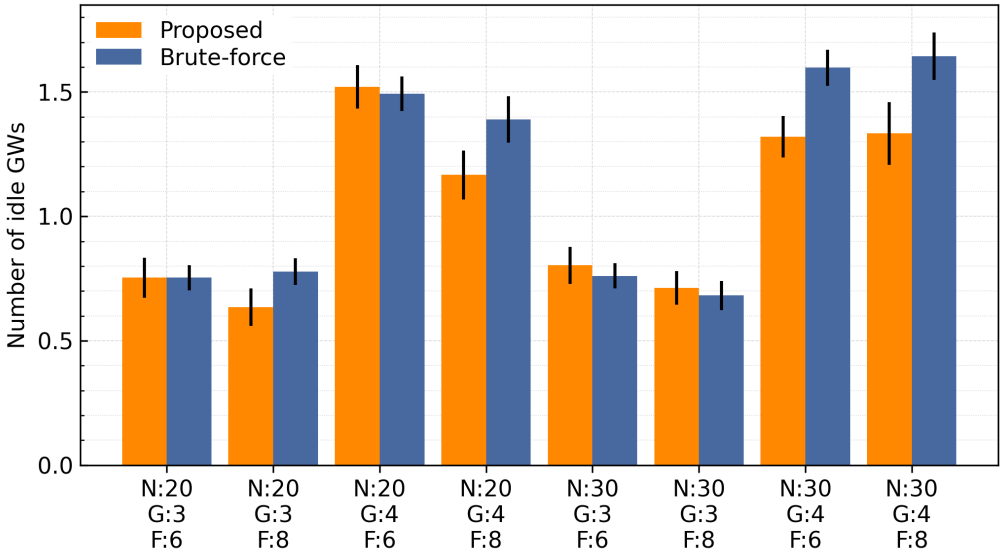


(c) Average E2E path hops of each flow.



(d) Average gateway traveling distance.

Figure 6.17: Performance upper limit: Limit of the path hop reduction and the comparison with *Path Coordinator*.



(e) Average number of the idling gateways.

Figure 6.17: Performance upper limit: Limit of the path hop reduction and the comparison with *Path Coordinator*.

6.5.4 Timeline Analysis

Finally, the time trend of the number of idle gateways is observed. In the three previous experiments, a snapshot of n flows was taken as a single test case with no temporal continuity. Here the transition of the number of idle gateways is visualized when gateways are relocated and flows are rerouted at some intervals while dynamically generating and disappearing flows. The evaluation conditions are shown in Table 6.5.

Table 6.5: Parameters for the timeline analysis.

Graph model	Connected RGG
Number of nodes per cluster	30
Number of gateways	5
Number of communication flows	40
Number of destinations	n-dests
Average session arrival interval	80, 120, 160 [sec]
Session sustaining period	240-720 [sec]
Relocation interval	200, 400, 600, 800 [sec]
Evaluation period	3600 [sec]

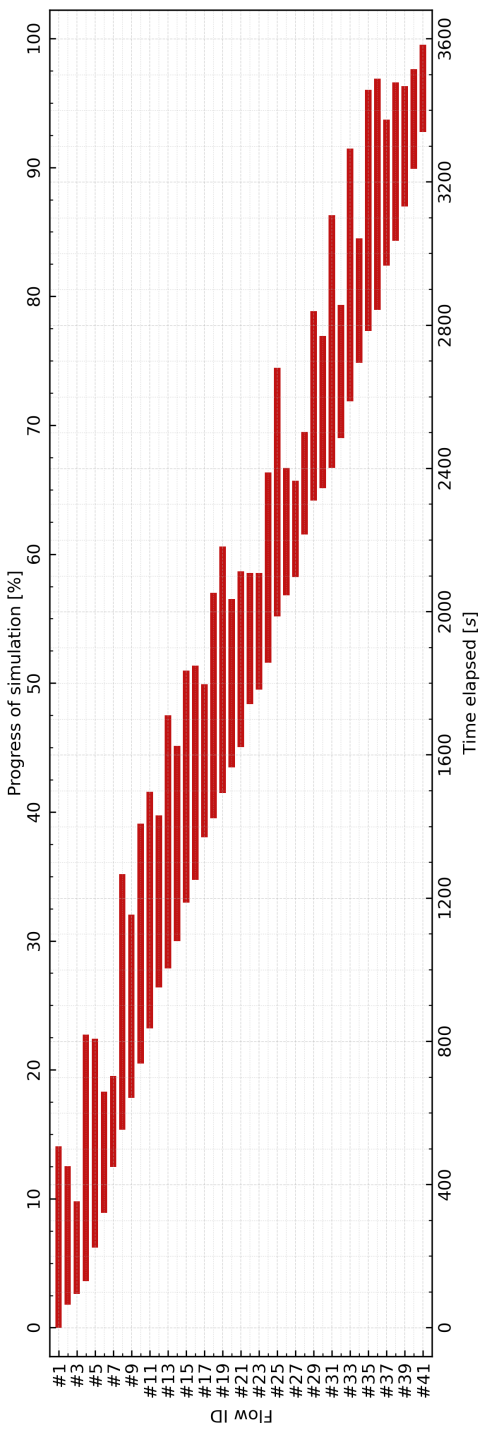
Figure 6.18a shows the timeline of one of the flow generation patterns. The horizontal axis is time-lapse, and the vertical axis is the flow ID. New flows are generated with an exponential distribution of 80 seconds on average, and their durations are determined by a uniform distribution of 240 to 720 seconds. A total of 41 flows were generated in 3600 seconds.

Figure 6.18b illustrates the result. The left vertical axis represents the number of idle gateways, and the right vertical axis represents the number of flows in the network at that time. The computer compared the case where the gateways are not moved from the initial placement and dynamically relocated every 200 seconds. Note that gateway movement and flow rerouting times are omitted from the graph for readability.

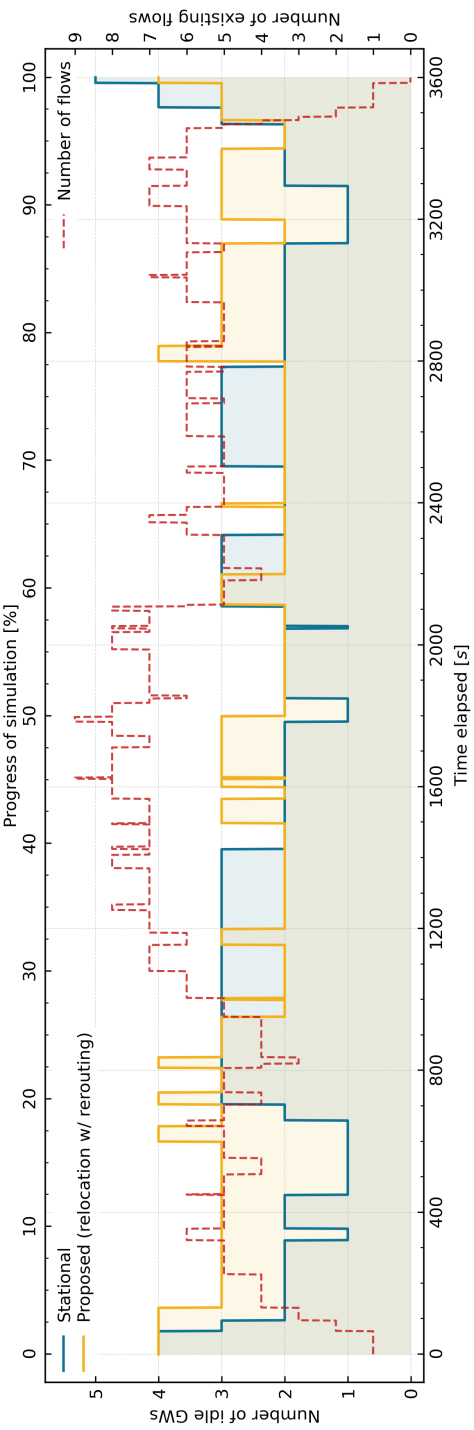
In the case of fixed gateway placement, the number of idle gateways decreases correspondingly with the increase in the number of flows. The graph area, i.e., the sum of the products of the number of idle gateways and its maintained time, is 8322 for the fixed placement and 9519 for the proposed method. Dividing the difference by the evaluation time of 3600 seconds, the average difference in the number of idle gateways is only 0.3325. However, it is worth mentioning that the proposed method provided more than two idle gateways for 3600 seconds at any time, which means more stable gateway availability than the fixed placement.

The first strategy of the proposed method is to direct the gateway to the optimal location to reduce the total number of path hops in the network or increase the allowable hop satisfaction rate as much as possible. If and only if multiple relocation patterns are found, the second strategy is to consolidate flows to some gateways so that the number of idle gateways can quickly increase. Although securing the idle gateways is only a secondary effect, the results are clearly different compared to the case where no action is taken.

All four experiments confirmed that the proposed algorithms in this paper fully demonstrate the expected performance and have sufficient capabilities to complement the previously proposed algorithm (Chapter 5).



(a) Flow generation pattern.



(b) Change in the number of idling gateways with *Path Coordinator* and fixed at the initial placement.

Figure 6.18: Timeline: Dynamically occurring and disappearing flows and corresponding changes in the number of idling gateways.

6.6 Conclusion

The author have continuously studied the communication quality improvement in the real-time video transmission using heterogeneous drone swarms, especially from the aspect of gateway mobility control. In this work, as an extension of the previous work, a simultaneous gateway relocation algorithm was proposed when the video traffic in the network increases more than the number of gateways. This powerful optimization method enables the transmission of as many remote videos as needed over the optimal paths, regardless of the number of gateways.

The proposed algorithm, *Path Coordinator*, is composed of two sub-algorithms – gateway relocation algorithm and flow rerouting algorithm. The relocation algorithm performs physical network optimization, updating the topology by geographic gateway movement; on the other hand, the rerouting algorithm performs logical optimization, reselecting gateways to accommodate each flow. Furthermore, these two algorithms can run independently, leaving the operator free to deal with the network congestion situation.

A conceptual protocol design extending the one proposed previously showed that the algorithm can be implemented in real heterogeneous drone swarms – an autonomous distributed environment – and has tolerance to the network partition. Various computer simulations were carried out to analyze the algorithms' performance characteristics and revealed the following points:

- The relocation algorithm reduced the number of path hops under all parameter conditions. Besides, in all conditions, a reduced hop count of more than twice the average was recorded at about 5% of all cases. The relocation time is also less than 30 seconds due to the optimal gateway traveling strategy.

- The rerouting algorithm achieved further path hop reduction from the post- relocated network for all conditions. In addition, it successfully increased the number of idle gateways than doing nothing.

- Overall, *Path Coordinator*, composed of the above two algorithms, improved the allowable hop satisfaction rate by up to 119.1% and achieved the original goal of improving the video quality. This performance is comparable to the performance upper limits obtained with the brute-force search.

To conclude, the algorithm has good performance to meet the initial objective, and the results were satisfactory.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This paper studied the reduction of communication delay in HANETs (Heterogeneous Ad Hoc Networks) through optimal mobility control of gateways.

First, a preliminary experiment was carried out in Chapter 4. An ad hoc network consisting of terminals with low-power processors was built, and the ratio of processing delay to end-to-end delay was evaluated in an actual experiment. The result confirmed that in the case of CPU-intensive transfer processes such as protocol conversion, the processing delay is up to 140% larger than the delay caused by the wireless link. The number of relay hops has a dominant effect on end-to-end delay in low-power ad hoc networks – a reduction in the number of path hops can reduce communication delay and the end-to-end delay can be expressed by a linear model.

Next, an autonomous gateway mobility control algorithm was designed and evaluated in Chapter 5. It is for establishing stabilized and low-delay communication among heterogeneous clusters, assuming that only gateways are controllable and relocatable to ensure the flexible operationality of drone swarms. The main algorithm is composed of two independent sub algorithms – *Link Stabilizer* and *Path Optimizer*. *Link Stabilizer* maintains the neighbor links and consists of two schemes: the neighbor clustering based on relative velocities and the gateway velocity calculation using a kinetic model. *Path*

Optimizer creates a shortcut to reduce the end-to-end delay for newly established communication by relocating the gateway dynamically. A conceptual protocol design was also presented to implement this algorithm into real-world drone swarms in a distributed manner. Computer simulation revealed that *Link Stabilizer* improved the connection stability for all three mobility models even under the high node mobility, and *Path Optimizer* reduced the communication delay by the optimal shortcut formation under any conditions of the experiments and its performance is comparable to the performance upper limit obtained by the brute-force searching.

Finally, a complementation algorithm for *Path Optimizer* was proposed in Chapter 6. *Path Optimizer* is a method to minimize the number of end-to-end path-hops by autonomously relocating gateways to create a shortcut in the network for each communication request. However, *Path Optimizer* has limitations in improving communication quality when more video sessions than the number of gateways are requested simultaneously. This makes *Path Coordinator* necessary – a new algorithm achieving a uniform reduction in end-to-end hops and maximizing the allowable hop satisfaction rate regardless of the number of sessions by introducing the cooperative and synchronous relocation of all gateways. *Path Coordinator* consists of two phases: first, physical optimization is performed by geographically relocating all gateways (relocation phase), and then logical optimization is achieved by modifying the relaying gateways of each video flow (rerouting phase). Computer simulations reveal that *Path Coordinator* adapts to various environments and demonstrate excellent performance which is comparable to the upper limits possible with brute-force search.

7.2 Future Works and Suggestions

As described in Chapter 3, some conditions were set to simplify the problem. These simplifications need to be resolved in further studies.

1. Interconnect two clusters.

2. Only the number of path hops affects the end-to-end communication delay and omits the consideration of dynamic link status like RSSI, packet loss ratio, etc.

3. The drone flies at enough high altitude to consider that all communications exist within line-of-sight distance. The free space propagation model is adopted to describe the radio characteristics, and no interference occurs.

4. In addition to the above, all drones fly at the same altitude. No three-dimensional behavior occurs, and the computer simulation is performed in a two-dimensional field.

5. The drone's flight time is only a probabilistic consideration and is not treated numerically.

Especially for the next step, the dynamic characteristics of the wireless link must be carefully considered. The followings are some suggestions from the author.

7.2.1 Building Enhanced Transmission Delay Model

The measurement experiments in Chapter 4 are primarily intended to determine the percentage of processing delay in the end-to-end delay. They do not provide a detailed analysis of the delay in the wireless link, i.e., the transmission propagation delay. In this experiment, the delay generated at the wireless link was slight because the SNR was sufficiently large, and the radio wave quality was stable.

However, in practice, there may be situations where link delay is dominant concerning end-to-end delay. It will be necessary to verify how the decrease in received power and SNR with the increase in radio propagation distance affect the communication delay and how it compares with theoretical values, such as MCS, in actual experiments. It is also necessary to model the relationship between inter-node distance and communication delay based on the numerical data obtained and to use this information for algorithm design.

7.2.2 Addressing Dynamic Link Quality

As discussed in Chapter 3, the dynamic characteristics of the wireless link are not considered algorithmically this time. The objective is to ensure end-to-end reachability, and the end-to-end path is not scored at a granularity finer than the number of path hops.

Wireless links are inherently different from wired links and should be treated with a firm distinction of characteristics in the future. Specifically, this means the introduction of link scoring using RSSI and other evaluation measures. It may be possible to build a highly reliable and stable logical path by adopting multipath routing to deal with unreliable and unstable wireless links.

7.2.3 Radio Interference Consideration

Related to the previous point, the simulator should be implemented with the consideration of radio interference. Even if the line of sight is clear between neighbors, reflections from the ground cannot be avoided. Therefore, at least a two-wave model must be employed instead of a free-space model. Moreover, ns3 is questionable in its ability to reproduce the behavior of the physical layer, and other simulators may need to be used.

7.2.4 Throughput-based QoS Evaluation

This study focused only on end-to-end reachability and adopted the number of route hops as the delay unit, but the study must be developed to treat this as a timescale. Specifically, the following should be considered:

- Calculation of SNR; setting of theoretical bit rate based on MCS.
- Airtime consumption when using the same frequency channel.
- Reproduction of video traffic. It depends on what is assumed in the encoding.

It will be necessary to consider the available bandwidth on the algorithm side. The encoding characteristics will also affect the gateway's mobility control or rerouting strategy.

Bibliography

[1] B. Rashid and M.H. Rehmani,"Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications*, vol.60, pp.192–219, 2016.

[2] T. Qiu, N. Chen, K. Li, D. Qiao, and Z. Fu, "Heterogeneous ad hoc networks: Architectures, advances and challenges," *Ad Hoc Networks*, vol.55, pp.143–152, 2017.

[3] A. Al-Saadi, R. Setchi, Y. Hicks, and S.M. Allen, "Routing protocol for heterogeneous wireless mesh networks," *IEEE Transactions on Vehicular Technology*, vol.65, no.12, pp.9773–9786, Dec 2016.

[4] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous internet of things build our future: a survey," *IEEE Communications Surveys Tutorials*, vol.20, no.3, pp.2011–2027, thirdquarter 2018.

[5] A. Prorok, M.A. Hsieh, and V. Kumar, "Formalizing the impact of diversity on performance in a heterogeneous swarm of robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp.5364–5371.

[6] P. Popovski, K.F. Trillingsgaard, O. Simeone, and G. Durisi, "5G wireless network slicing for EMBB, URLLC, and MMTC: A communication-theoretic view," *IEEE Access*, vol.6, pp.55765–55779, 2018.

[7] R. Arnold, J. Jablonski, B. Abruzzo, and E. Mezzacappa, "Heterogeneous UAV multi-role swarming behaviors for search and rescue," in *2020 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. IEEE, 2020, pp.122–128.

[8] J. A. Alvarez Aldana, S. Maag, and F. Zaïdi, "MANETs interoperability: Current trends and open research," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp.481–487.

[9] H. Hassan, P. Trwoga, and I. Kale, "If-MANET: Interoperable framework for mobile ad hoc networks," in *Computer Networks: 22nd International Conference*. 2015, pp.54–68.

[10] S. Fujiwara, T. Ohta, and Y. Kakuda, "An inter-domain routing for heterogeneous mobile ad hoc networks using packet conversion and address sharing," in *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp.349–355.

[11] B. Zhou, Z. Cao, and M. Gerla, "Cluster-based inter-domain routing (CIDR) protocol for MANETs," in *2009 Sixth International Conference on Wireless On-Demand Network Systems and Services*. IEEE, 2009, pp. 19–26.

[12] S. Lee, S.H.Y. Wong, C. Chau, K. Lee, J. Crowcroft, and M. Gerla, "InterMR: Inter-manet routing in heterogeneous manets," in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*. IEEE, 2010, pp.372–381.

[13] W. Gao, J. Nguyen, Y. Wu, W. G. Hatcher, and W. Yu, "A bloom filter-based dual-layer routing scheme in large-scale mobile networks," *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp.1–9.

[14] H. Safa, M. Karam, and B. Moussa, "PHAODV: Power aware heterogeneous routing protocol for MANETs," *Journal of Network and Computer Applications*, vol.46, pp.60–71, 2014.

[15] P. V. Ghate and H. K. Pati, "Collaborative distributed communication in heterogeneous environments: A comprehensive survey," *Journal of Network and Computer Applications*, vol.61, pp.1–20, 2016.

[16] C.K. Chau, J. Crowcroft, K.W. Lee, and S.H. Wong, "Inter-domain routing for mobile ad hoc networks," in *3rd International Workshop on Mobility in the Evolving Internet Architecture, MobiArch '08*. ACM, 2008, pp.61–66.

[17] A. Sharma, P. Vanjani, N. Paliwal, C. M. W. Basnayaka, D. N. K. Jayakody, H.-C. Wang, and P. Muthuchidambaranathan, "Communication and networking technologies for UAVs: A survey," *Journal of Network and Computer Applications*, vol.168, p.102739, Oct. 2020.

[18] N. Sharma, M. Magarini, D. N. K. Jayakody, V. Sharma, and J. Li, "On-demand ultra-dense cloud drone networks: Opportunities, challenges and benefits," *IEEE Communications Magazine*, vol.56, no.8, pp.85–91, Aug. 2018.

[19] W. Chen, J. Liu, H. Guo, and N. Kato, "Toward robust and intelligent drone swarm: Challenges and future directions," *IEEE Network*, vol.34, no.4, pp.278–283, Jul. 2020.

[20] R. Doriya, S. Mishra, and S. Gupta, "A brief survey and analysis of multi-robot communication and coordination," in *International Conference on Computing, Communication & Automation*. IEEE, 2015, pp.1014–1021.

[21] X. Zhang, B. Xian, B. Zhao, and Y. Zhang, "Autonomous flight control of a nano quadrotor helicopter in a GPS-denied environment using on-board vision," *IEEE Transaction on Industrial Electronics*, vol.62, no.10, pp.6392–6403, Oct. 2015.

[22] S. Chung, A.A. Paranjape, P. Dames, S. Shen, and V.Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol.34, no.4, pp.837–855, Aug 2018.

[23] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles – a survey," *Journal of Industrial Information Integration*, vol.16, p.100–106, Dec. 2019.

[24] M. Campion, P. Ranganathan, and S. Faruque, "A review and future directions of uav swarm communication architectures," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2018, pp.0903–0908.

[25] W. Shi, H. Zhou, J. Li, W. Xu, N. Zhang, and X. Shen, "Drone assisted vehicular networks: Architecture, challenges and opportunities," *IEEE Network*, vol.32, no.3, pp.130–137, May 2018.

[26] O. Bautista, K. Akkaya, and A. S. Uluagac, "Customized novel routing metrics for wireless mesh-based swarm-of-drones applications," *Internet of Things*, vol.11, p.100265, 2020.

[27] G. Kim, I. Mahmud, and Y. Cho, "Self-recovery scheme using neighbor information for multi-drone ad hoc networks," in *2017 23rd Asia-Pacific Conference on Communications (APCC)*. IEEE, 2017, pp.1–5.

[28] O. Shrit, S. Martin, K. Alagha, and G. Pujolle, "A new approach to realize drone swarm using ad-hoc network," in *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. IEEE, 2017, pp.1–5.

[29] O. Bouachir, M. Aloqaily, F. Garcia, N. Larrieu, and T. Gayraud, "Testbed of QoS ad-hoc network designed for cooperative multi-drone tasks," in *17th ACM International Symposium on Mobility Management and Wireless Access*. ACM, 2019, pp.89–95.

[30] J. Rands and Q. Han, "Regression-based network monitoring in swarm robotic systems," *16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp.1–10.

[31] C.S.V. Gutiérrez, L.U.S. Juan, I.Z. Ugarte, and V.M. Vilches, "Real-time Linux communications: An evaluation of the linux communication stack for real-time robotic applications," in *arXiv preprint*. arXiv:1808.10821, 2018.

[32] Z. Yuan, J. Jin, L. Sun, K. Chin, and G. Muntean, "Ultra-reliable IoT communications with UAVs: A swarm use case," *IEEE Communications Magazine*, vol.56, no.12, pp.90–96, Dec. 2018.

[33] M. Kadivar, M. E. Shiri, and M. Dehghan, "Distributed topology control algorithm based on one- and two-hop neighbors' information for ad hoc networks," *Computer Communications*, vol.32, no.2, pp.368–375, Feb. 2009.

[34] R. Wattenhofer and A. Zollinger, "XTC: A practical topology control algorithm for ad-hoc networks," in *18th International Parallel and Distributed Processing Symposium*. 2004, pp.216–223.

[35] Z. Mi and Y. Yang, "Topology control and coverage enhancement of dynamic networks based on the controllable movement of mobile agents," in *2011 IEEE International Conference on Communications (ICC)*. IEEE, 2021, pp.1–5.

[36] L. Kong, L. Ye, F. Wu, M. Tao, G. Chen, and A. V. Vasilakos, "Autonomous relay for millimeter-wave wireless communications," *IEEE Journal on Selected Areas in Communications*, vol.35, no.9, pp.2127–2136, Sep. 2017.

[37] P. Zhan, K. Yu, and A. L. Swindlehurst, "Wireless relay communications with unmanned aerial vehicles: Performance and optimization," *IEEE Transaction on Aerospace and Electronic Systems*, vol.47, no. 3, pp.2068–2085, Jul. 2011.

[38] L. Chen, Y. Huang, F. Xie, Y. Gao, L. Chu, H. He, Y. Li, F. Liang, and Y. Yuan, "Mobile relay in LTE-advanced systems," *IEEE Communications Magazine*, vol.51, no.11, pp.144–151, Nov. 2013.

[39] W. Wang, V. Srinivasan, and Kee-Chaing Chua, "Extending the lifetime of wireless sensor networks through mobile relays," *IEEE/ACM Transaction on Networking*, vol.16, no.5, pp.1108–1120, Oct. 2008.

[40] K. Li, W. Ni, X. Wang, R. P. Liu, S. S. Kanhere, and S. Jha, "Energy-efficient cooperative relaying for unmanned aerial vehicles," *IEEE Transaction on Mobile Computing*, vol.15, no.6, pp.1377–1386, Jun. 2016.

[41] F. Mezghani, P. Kortoci, N. Mitton, and M. Di Francesco, "A multi-tier communication scheme for drone-assisted disaster recovery scenarios," in *2019 IEEE 30th Annual International Symposium on Personal*. IEEE, 2019, pp.1–7.

[42] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Vehicular Technology Magazine*, vol.12, no.3, pp.73–82, Sep. 2017.

[43] 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 Journal*, vol.6, no.2, pp.3727–3742, Apr. 2019.

[44] S. Sekander, H. Tabassum, and E. Hossain, "Multi-tier drone architecture for 5G/B5G cellular networks: Challenges, trends, and prospects," *IEEE Communications Magazine*, vol.56, no.3, pp.96–103, Mar. 2018.

[45] N. Saputro, K. Akkaya, R. Algin, and S. Uluagac, "Drone-assisted multi-purpose roadside units for intelligent transportation systems," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp.1–5.

[46] L. Chen and W.B. Heinzelman, "QoS-aware routing based on bandwidth estimation for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol.23, no.3, pp.561–572, March 2005.

[47] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *IEEE Global Telecommunications Conference 1995 (GLOBECOM)*. IEEE, 1995, vol.3, pp.2129–2133.

[48] Q. Sun and H. Langendörfer, "Efficient multicast routing for delay-sensitive applications," in *Second Workshop on Protocols for Multimedia Systems (PROMS)*. IEEE, 1995, pp.452–458.

[49] S. Vutukury and J.J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 1999, Association for Computing Machineryj p.227–238,

[50] A. Skordylis and N. Trigoni, "Delay-bounded routing in vehicular ad-hoc networks," in *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp.341–350, 2008.

[51] P. Nanda and A. Simmonds, "A scalable architecture supporting qos guarantees using traffic engineering and policy based routing in the internet," *International Journal of Communications, Network and System Sciences*, pp.583–591, 2009.

[52] M.A. Gawas, K. Modi, P. Hurkat, and L.J. Gudino, "QoS based multipath routing in manet: A cross layer approach," in *International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2017, pp.1806–1812.

[53] S. Othmen, A. Belghith, F. Zarai, M.S. Obaidat, and L. Kamoun, "Power and delay-aware multi-path routing protocol for ad hoc networks," in *International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, 2014, pp.1–6.

[54] Y. Yan, L. Ci, R. Zhang, and Z. Wang, "Load balancing routing algorithm among multiple gateways in MANET with internet connectivity," in *16th International Conference on Advanced Communication Technology*. IEEE, 2014, pp.388–392.

[55] A. Belhoul, Y.A. Şekercioğlu, and N. Mani, "Mobility-aware RSVP: A framework for improving the performance of multimedia services over wireless IP-based mobile networks," *Computer Communications*, vol.32, no.4, pp.569–582, 2009.

[56] S. Zaidi, S. Bitam, and A. Mellouk, "Hybrid error recovery protocol for video streaming in vehicle ad hoc networks," *Vehicular Communications*, vol.12, pp.110–126, 2018.

[57] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (UASs) the future of wildlife monitoring? A review of accomplishments and challenges: a review of UASs in wildlife monitoring," *mammal review*, vol.45, no.4, pp.239–252, oct. 2015.

[58] C. Burke, M. Rashman, S. Wich, A. Symons, C. Theron, and S. Longmore, "Optimizing observing strategies for monitoring animals using drone-mounted thermal infrared cameras," *International Journal of Remote Sensing*, vol.40, no.2, pp.439–467, Jan. 2019.

[59] J.-A. Vayssade, R. Arquet, and M. Bonneau, "Automatic activity tracking of goats using drone camera," *Computers and Electronics in Agriculture*, vol.162, pp.767–772, Jul. 2019.

[60] Y. Sato, S. Ozawa, Y. Terasaka, K. Minemoto, S. Tamura, K. Shingu, M. Nemoto, and T. Torii, "Remote detection of radioactive hotspot using a Compton camera mounted on a moving multi-copter drone above a contaminated area in

Fukushima," *Journal of Nuclear Science and Technology*, vol.57, no.6, pp.734–744, Jun. 2020.

[61] B. Yu, X. Dong, Z. Shi, and Y. Zhong, "Formation control for quadrotor swarm systems: Algorithms and experiments," in *32nd Chinese Control Conference (CCC)*. 2013, pp.7099–7104.

[62] H.-J. Kim and H.-S. Ahn, "Realization of swarm formation flying and optimal trajectory generation for multi-drone performance show," *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2016, pp.850–855.

[63] H. Li, J. Peng, W. Liu, K. Gao, and Z. Huang, "A novel communication-aware formation control strategy for dynamical multi-agent systems," *Journal of the Franklin Institute*, vol.352, no.9, pp.3701–3715, Sep. 2015.

[64] W. Youssef and M. Younis, "Intelligent gateways placement for reduced data latency in wireless sensor networks," in *2007 IEEE International Conference on Communications*. IEEE, 2007, pp.3805–3814.

[65] Y. Cao, Y. Shi, J. Liu and N. Kato, "Optimal satellite gateway placement in space-ground integrated network for latency minimization with reliability guarantee," *IEEE Wireless Communications Letters*, vol.7, no.2, pp.174–177, April 2018.

[66] Tang, Jian, Bin Hao, and Arunabha Sen, "Relay node placement in large scale wireless sensor networks," *Computer Communications*, vol.29, no.4, pp.490–501, Feb. 2006.

[67] B. Aoun, R. Boutaba, Y. Iraqi and G. Kenward, "Gateway placement optimization in wireless mesh networks with QoS constraints," *IEEE Journal on Selected Areas in Communications*, vol.24, no.11, pp.2127–2136, Nov. 2006.

[68] Y. Miao, S. Vural, Z. Sun, and N. Wang, "A unified solution for gateway and in-network traffic load balancing in multihop data collection scenarios," *IEEE Systems Journal*, vol.10, no.3, pp.1251–1262, 2015.

[69] D. Kim et al., "Minimum data-latency-bound$k$-sink placement problem in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol.19, no.5, pp.1344–1353, Oct. 2011.

[70] G. e. m. Zhioua, H. Labiod, N. Tabbane, and S. Tabbane, "Fqgws: A gateway selection algorithm in a hybrid clustered VANET LTE-advanced network: Complexity and performances," in *2014 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, pp.413–417, Feb 2014.

[71] S. W. Card, A. Wiethuechter, R. Moskowitz, and A. Gurtov, "Drone remote identification protocol (DRIP) requirements and terminology," *Internet Engineering Task Force*, Request for Comments RFC 9153, Feb. 2022.

[72] M. A. Gawas, K. Modi, P. Hurkat, and L. J. Gudino, "QoS based multipath routing in MANET: A cross layer approach," in *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2017, pp. 1806–1812.

[73] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an IP backbone network," *IEEE Journal on Selected Areas in Communications*, vol.21, no.6, pp.908–921, 2003.

[74] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Dx: Latency-based congestion control for datacenters," *IEEE/ACM Transactions on Networking*, vol.25, no.1, pp.335–348, 2016.

[75] D. Sanghi, A. K. Agrawala, O. Gudmundsson, and B. N. Jain, "Experimental assessment of end-to-end behavior on internet," in *IEEE INFOCOM'93 The Conference on Computer Communications*. IEEE, 1993, pp.867–874.

[76] L. Angrisani, G. Ventre, L. Peluso, and A. Tedesco, "Measurement of processing and queuing delays introduced by an open-source router in a single-hop network," *IEEE Transaction on Instrumentation and Measurement*, vol.55, no.4, pp.1065–1076, 2006.

[77] A. Beifuß, D. Raumer, P. Emmerich, T. M. Runge, F. Wohlfart, B. E. Wolfinger, and G. Carle, "A study of networking software induced latency," in *2015 International Conference and Workshops on Networked Systems (NetSys)*. IEEE, 2015, pp.1–8.

[78] K. M. Salehin, R. Rojas-Cessa, C.-b. Lin, Z. Dong, and T. Kijkanjanarat, "Scheme to measure packet processing time of a remote host through estimation of end-link capacity," *IEEE Transactions on Computers*, vol.64, no.1, pp.205–218, 2013.

[79] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *IEEE Global Telecommunications Conference 2004 (GLOBECOM)*. IEEE, 2004, pp.1629–1634.

[80] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an IP backbone network," *IEEE Journal on Selected Areas in Communications*, vol.21, no.6, pp.908–921, 2003.

[81] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot, "Bridging router performance and queuing theory," *ACM SIGMETRICS Performance Evaluation Review*, vol.32, no.1, pp.355–366, 2004.

[82] L. Angrisani, G. Ventre, L. Peluso, and A. Tedesco, "Measurement of processing and queuing delays introduced by an open-source router in a single-hop network," *IEEE Transactions on Instrumentation and Measurement*, vol.55, no.4, pp.1065–1076, 2006.

[83] G. Almes, S. Kalidindi, M. J. Zekauskas, and A. Morton, "A one-way delay metric for ip performance metrics (IPPM)," *Internet Engineering Task Force*, Request for Comments RFC 7679, Jan. 2016.

[84] L. Caldas-Calle, J. Jara, M. Huerta, and P. Gallegos, "QoS evaluation of VPN in a Raspberry Pi devices over wireless network," *2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS)*, IEEE, 2017, pp.125–128.

[85] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2014, vol.3, pp.1629–1634.

[86] S. Yashkov, "Processor-sharing queues: some progress in analysis," *Queueing Systems*, vol.2, no.1, pp.1–17, 1987.

[87] Y. Daldoul, D.-E. Meddour, and A. Ksentini, "IEEE 802.11 n/ac data rates under power constraints," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp.1–6.

[88] B. An and S. Papavassiliou, "A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks," *International Journal of Network Management*, vol.11, no.6, pp.387–395, 2001.

[89] M. Aida, C. Takano and M. Murata, "Oscillation model for describing network dynamics caused by asymmetric node interaction," *IEICE Transactions on Communications*, vol.E101-B, no.1, pp.123–136, 2018.

[90] Hybrid Wireless Mesh Protocol, "HWMP Protocol specification," *The Working Group for WLAN Standards of the Institute of Electrical and Electronics Engineers*, Nov. 2006. [Online]. Available: `https://mentor.ieee.org/802.11/public/06/11-06-1778-01-000s-hwmp-specification.doc`

[91] D.B. Johnson and D.A. Maltz, "Dynamic source Routing in ad hoc wireless networks," *Mobile Computing*, vol.353, pp.153-181, 1996.

[92] G. Jayakumar and G. Ganapathi, "Reference point group mobility and random waypoint models in performance evaluation of MANET routing protocols," *Journal of Computer Networks and Communications*, vol.2008, pp.13:1–13:10, 2008.

[93] W. Wang, X. Guan, B. Wang, and Y. Wang, "A novel mobility model based on semi-random circular movement in mobile ad hoc networks," *Information Sciences*, vol.180, no.3, pp.399–413, 2010.

[94] O. Bouachir, A. Abrassart, F. Garcia, and N. Larrieu, "A mobility model for UAV ad hoc network," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp.383–388.

[95] J. Dall and M. Christensen, "Random geometric graphs," *Physical Review E*, vol.66, p.016121, Jul. 2002.

[96] R. Bauer, G. D'Angelo, D. Delling, A. Schumm, and D. Wagner, "The shortcut problem – complexity and algorithms," *Journal of Graph Algorithms and Applications*, vol.16, no.2, pp.447–481, 2012.

[97] T. Godquin, M. Barbier, C. Gaber, J.-L. Grimault, and J.-M. L. Bars, "Placement optimization of IoT security solutions for edge computing based on graph theory," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp.1–7.

[98] I. Lera, C. Guerrero, and C. Juiz, "Comparing centrality indices for network usage optimization of data placement policies in fog devices," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2018, pp.115–122.

[99] A. Jain and B. V. R. Reddy, "Node centrality in wireless sensor networks: Importance, applications and advances," in *2013 3rd IEEE International Advance Computing Conference (IACC)*. IEEE, 2013, pp.127–131.

[100] A. Syarif, A. Abouaissa, L. Idoumghar, P. Lorenz, R. Schott, and G. S. Staples, "New path centrality based on operator calculus approach for wireless sensor network deployment," *IEEE Transaction on Emerging Topics in Computing*, vol.7, no.1, pp.162–173, Jan. 2019.

Achievements

Awards and Honors

[**Z-1**] 情報ネットワーク研究会研究賞, “力学モデルに基づく HANETs における GW 移動制御の性能解析”, 電子情報通信学会, 2020 年度.

Publications forming part of the dissertation

A. Journal Papers

[**A-1**] T. Miya, K. Ohshima, Y. Kitaguchi, and K. Yamaoka, “Autonomous Gateway Mobility Control for Heterogeneous Drone Swarms: Link Stabilizer and Path Optimizer,” *IEICE Transaction on Communications*, Vol.E105-B, No.4, pp.432–448, Apr. 2022.

[**A-2**] T. Miya, K. Ohshima, Y. Kitaguchi, and K. Yamaoka, “Adaptive GW Relocation and Strategic Flow Rerouting for Heterogeneous Drone Swarms,” *IEICE Transaction on Communications*, Vol.E106-B, No.4, Apr. 2023. (to be appeared)

B. International Conferences (Peer-review)

[**B-1**] T. Miya, K. Ohshima, Y. Kitaguchi, and K. Yamaoka, “Experimental Analysis of Communication Relaying Delay in Low-Energy Ad-hoc Networks,” *2021 IEEE 18th Annual Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, USA, pp.1–2, Jan. 2021.

C. Japanese Domestic Conference (No Peer-review)

[**C-1**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “力学モデルに基づく HANETs における GW 移動制御の性能解析”, 電子情報通信学会技術研究報告, 一般社団法人 電子情報通信学会, Vol. 120, No. 293, pp. 13–18, Dec. 2020.

[**C-2**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “低電力アドホックネットワークにおける通信中継負荷の分析”, 電子情報通信学会技術研究報告, 一般社団法人 電子情報通信学会, Vol. 119, No. 461, pp. 109–114, Mar. 2020.

[**C-3**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “力学モデルによる HANETs トポロジコントロールの一考察”, 電子情報通信学会総合大会講演論文集, 一般社団法人 電子情報通信学会, Vol. 2021, No. B–7–21, Mar. 2021.

[**C-4**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “HANETs における GW 自律移動制御による低遅延通信の実現”, 電子情報通信学会技術研究報告, 一般社団法人 電子情報通信学会, Vol. 120, No. 414, pp. 67–72, Mar. 2021.

Publications relevant to the dissertation but not forming part of it

b. International Conferences (Peer-review)

[**b-1**] T. Miya, K. Ohshima, Y. Kitaguchi, K. Yamaoka, “The Upper Limit of Flow Accommodation under Allowable Delay Constraint in HANETs,” *The 2019 16th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, USA, pp.1–2, Jan. 2019.

c. Japanese Domestic Conference (No Peer-review)

[**c-1**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “自律クラスタ群における許容遅延に基づく GW 最適配置の一考察”, 電子情報通信学会ソサイエティ大会講演論文集, 一般社団法人 電子情報通信学会 Sept. 2018.

[**c-2**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “異種アドホックネットワーク間相互接続環境における最適経路探索手法”, 電子情報通信学会技術研究報告, 一般社団

法人 電子情報通信学会, Vol. 117, No. 460, pp. 219–224, Mar. 2018.

[**c-3**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “ゲートウェイ累積遅延と許容遅延を考慮したフロー収容数の最大化”, 電子情報通信学会総合大会講演論文集, 一般社団法人 電子情報通信学会, Vol. 2018, No. B–7–10, Mar. 2018.

[**c-4**] 宮 太地, 大島 浩太, 北口 善明, 山岡 克式, “リンク切断を考慮した HANETs における許容遅延制約下の GW 最適配置”, 電子情報通信学会技術研究報告, 一般社団法人 電子情報通信学会, Vol. 118, No. 466, pp. 331–336, Mar. 2019.