# Persistent Communication Connectivity of Multi-agent Systems

Zhiyang Ju
ORCID: 0000-0002-6186-6321

Submitted in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Electrical and Electronic Engineering
THE UNIVERSITY OF MELBOURNE

September 2018

# Abstract

COMMUNICATION connectivity is a common requirement in multi-agent systems. Multi-agent systems typically need to maintain communication connectivity to complete their tasks. Due to the intrinsic complex structure of multi-agent systems, ensuring communication connectivity is typically very challenging. In most of the existing works, communication connectivity is addressed assuming the cooperation of all agents in the system. This is often undesirable,as the communication connectivity is usually not the ultimate goal but rather a necessary feature to complete another essential task. Another gap in the current literature is that this problem is always considered assuming a simple agent dynamic model. However, more complex models are often needed in practical situations, for example, when agents are unmanned aerial vehicles (UAVs). The main objective in this thesis is to consider a problem where a set of agents (clients) do not cooperate in maintaining communication connectivity, while another set of agents (routers) need to achieve this task by cooperating with each other. At the same time, we will consider more complex agent models than is normally done in the literature.

In Chapter 2, we present the communication connectivity control design for multi-agent systems with two clients. These agents are termed as clients, and the others are termed as routers. In such a system, a simple structure of communication relationship is selected to be used for control design. The agent dynamic model considered is a quadrotor model. However, the control design strategy can be applied for other types of models.

In Chapter 3, we address the communication connectivity problem for multi-agent systems with an arbitrary number of clients. Multiple clients make the communication structure and control design more complex. Moreover, the communication structure selected for control design is updated when necessary, which means that the obtained control scheme is time-varying. The agent model is a single integrator model in order to make the presentation simpler. As we already pointed out, our results can be extended for other types of agent dynamic models as long as the motion controllers satisfy some requirements.

Finally, in Chapter 4, we implement communication connectivity control policy designed in Chapter 3 in experiments by using e-puck robots. The system in the experiment is set up with three clients and three routers, where clients are paper boards with markers on them for position tracking while routers are e-puck robots. OptiTrack Flex13 camera system is used to track the positions of agents, and a HP Elitebook 840 laptop running Windows is used to generate control inputs for routers. Two experimental settings were considered: clients move towards each other and clients move away from each other . The experiment is consistent with and illustrates the theoretical results.

# Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Zhiyang Ju

ORCID: 0000-0002-6186-6321, September 2018

# Acknowledgements

I would like to express special appreciation and thanks to my supervisors. First, I would like to thank my main supervisor Prof. Dragan Nešić for accepting me as a PhD student, for guiding my research and encouraging me to continue my study, for your sincere advice and feedback, for all the help in my research and life. I would like to express my gratitude to my co-supervisor Dr. Iman shames. Thank you for all the technical support you gave me throughout the last four years, for all your detailed advice and support in study and life that helped me overcome difficulties.

I would also like to thank my committee chair Prof. Michael Cantoni, for his insight comments during my PhD study, for the advice you give when there were problems.

I would like to thank my undergraduate supervisors, Prof. Huijun Gao, Prof. Shen Yin. I appreciate your help and encouragement that lead me to continue my PhD study after my graduation.

My gratitude goes to the University of Melbourne and China Scholarship Council for providing scholarship and support for my research.

The Control and Signal Processing Lab is a wonderful workplace. I would like to thank the professors and colleges in this lab. Among them, I would like to mention Dr. Wei Wang, Assoc. Prof. Ying Tan, Dr. Saeed Ahmadizadeh, Tianci, Zhanghan, Wenchao, Xiao Xiao, Dr. Lei Lu, Dekang, Hason, Sasan, Boyan, Daniel. I would like to mention some friends in our Electical and Electronic Department, Michael, Han, Lingjuan, Yaqi, Ali, Di, Yu, Lihua, Tian, Yifei, Bowen, Jian, thanks for the great time we spend in the four years.

I would like to thank my friends in Melbourne, Fuqiang, Huashan, Hao, Yaoli, Shiyang, Guoping, Chengcheng, Jian, Songcheng, Jing, Wei, Tianyi, Junchao, Xueying, Chengming for the sincere friendship and happy times.

I would like to thank my parents for all your love, support and encouragement throughout my life. I would like to thank all other family members and friends in China.

Finally, I would like to thank my girlfriend Hongzhan Di. Without your love and support, I would have not made it to this point. Thank you for the happiness and light you bring into my life.

Zhiyang Ju

Melbourne, August 2018

# Preface

This thesis is submitted for the degree of Doctor of Philosophy at the University of Melbourne. The research presented was conducted under the supervision of Prof. Dragan Nešić and Dr. Iman Shames. The work towards this thesis was carried out in collaboration with Prof. Dragan Nešić and Dr. Iman Shames. The percentage of contribution of myself is more than 50%. Some ideas of the work was achieved by discussing with my supervisors Prof. Dragan Nešić and Dr. Iman Shames. My supervisors also contributed to the proofreading, comments and polishing of the thesis.

To the best of my knowledge, this work is generated originally, except where acknowledge and references are made to other work. Any substantially similar thesis has not been or is not being submitted for any other degree, diploma or other qualification at any other university or institution. Part of this work has been presented in the following publication:
Z. Ju, I. Shames, D. Nešić. Ensuring Communication Connectivity in Multi-agent Systems in the Presence of Uncooperative Clients. Decision and Control (CDC), 2016 IEEE 55th Conference on. IEEE, 2016.
This paper is mainly conducted by myself. Iman gave feedback and rewrote some parts. Iman and Dragan gave comments, did proof reading and polished the paper.

This work was supported by China Scholarship Council, Melbourne International Fee Remission Scholarship and Melbourne Abroad Travel Scholarship.

# Contents

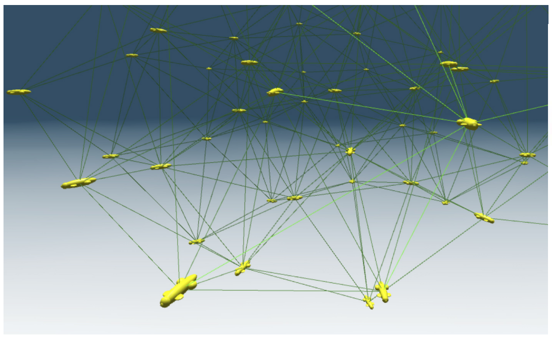
# List of Figures

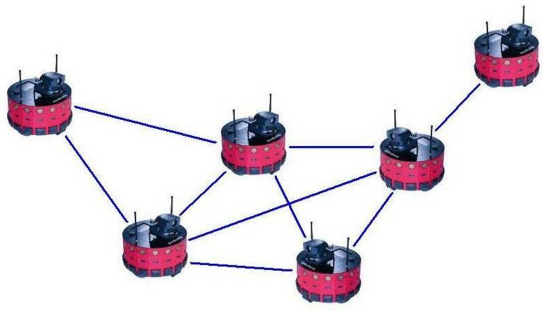
# Chapter 1
# Introduction

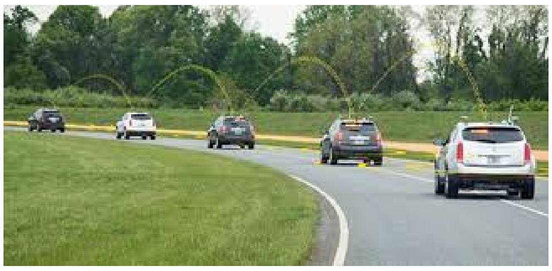
MULTI-AGENT systems have drawn a lot of interest in system and control society and they will continue to attract attention as new missions and applications become relevant. A number of missions require agents to cooperate between each other (e.g. a vehicle platoon) and in order to achieve effective cooperation they need to communicate between each other. Thus, maintaining the communication connectivity is often a necessary first step in ensuring that effective cooperation can be achieved. The focus of our work is to study how to guarantee communication connectivity in multi-agent systems. In this chapter, we will first give a general overview of research in multi-agent systems. After that, we will focus on the literature on communication connectivity of these systems. Finally, we formulate the problem we consider and summarise the main contribution of this thesis.

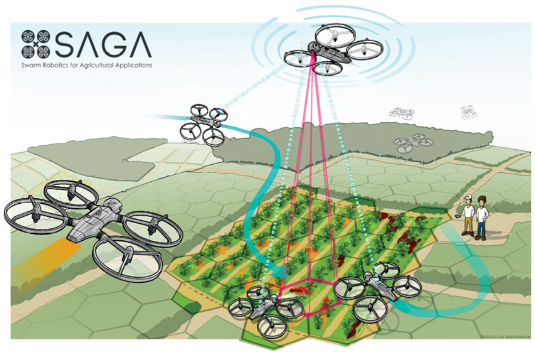
## 1.1 Introduction and Motivation

The multi-agent systems we consider are composed of multiple interacting agents, which have their own computation and movement capabilities. The agents in the system can make their own decisions based on the information they collect from their own sensors, as well as information from other agents they are communicating with. With this requirement on the agents, the systems we consider in this thesis are typically multi-robot systems, such as multiple unmanned underwater/ground/aerial vehicle (i.e. UUV, UGV and UAV) systems. In our setting, interaction is conducted by exchanging information with others through communication links. Some examples of the multi-agent systems are


(a) Multi-agent system with UUVs for Mornitoring Water Quality [7]


(b) Multi-agent system with UGVs as Experiment Platform [49]


(c) Automated Car Platoon [46]



(d) Swarm Robotics for Agricultural Applications [63]

Figure 1.1: Muti-agent system examples

shown in Figure 1.1. These systems are UUVs for water quality monitoring [7], UGVs for experimental platform [49], vehicle platoon [46] and swarm of drones for agricultural applications [63]. Other than these examples, we would present more details of applications of multi-agent systems in the next subsection.

As shown in the examples in Figure 1.1, multi-agent systems contain different types of agents and perform different tasks. However, there are always two main elements in such systems, which are agents themselves and the communication between each other. Thus, we use circles and dashed lines to represent the structure of the multi-agent systems we consider in this thesis in Figure 1.2. In this figure, circles denote the agents, and dashed lines denote that the agents on the two ends can exchange information, i.e. they communicate with each other.

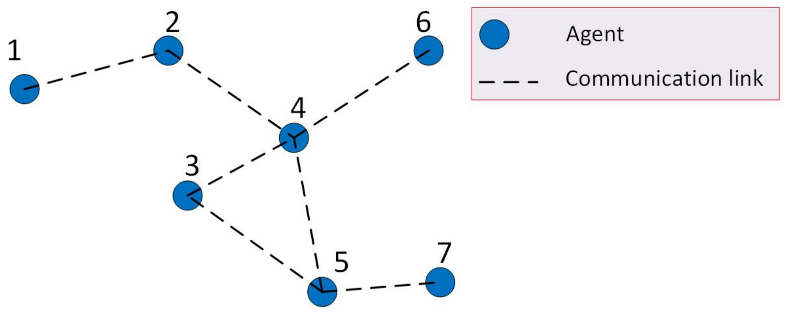


Figure 1.2: Multi-agent system example

### 1.1.1 Applications of Multi-agent Systems

Multi-agent systems have a wide range of applications, such as search and rescue, tracking, surveillance, underwater exploration, and deliveries. Next, we present a brief overview of these applications. Instead of trying to be exhaustive in this overview, we would just give a few representative examples. The underwater robotics and water quality project in EPFL [7] uses an underwater robotic multi-agent system to measure the water quality in lakes and freshwater reservoirs. The underwater robots which act as agents can cooperatively complete the measurement by moving and communicating with each other. Another underwater multi-agent platform for monitoring coral reefs and fisheries is reported in [66]. A multi-agent system is also used to track the long term zebra migration by using wireless sensors [75]. Moreover, multi-agent systems can be used to manage agriculture and forestry effectively [17]. In this project, a fleet of ground and aerial unmanned vehicles are used to form a sensor network to improve crop quality, health and safety for humans and reduce production costs by means of sustainable crop management. The Google Loon project [27] plans to use a group of balloons to provide internet connectivity for remote rural areas. Similar projects using a group of UAVs are conducted by the Connectivity Lab of Facebook [53]. Other well known applications are search and rescue after a natural disaster [50], such as flood, forest fire surveillance [35] and intruders detection [76].

As we can see from the above examples, multi-agent systems have diverse applications in many aspects of our society and life. This attracted a significant research effort in this area. In the next subsection, we will give a brief review of the literature on multi-agent systems.

### 1.1.2 A Brief Review of Research in Multi-agent Systems

Research in networked multi-agent systems increased substantially in the 1990s due to the development of inexpensive and reliable wireless communication systems [40]. In the last several decades, this area has attracted increasing interest in research and applications. Research on multi-agent systems often focused on collaborative missions, such as consensus, distributed optimization, intelligent coordination and distributed estimation [52]. In order for agents to cooperate effectively, it is necessary to maintain a certain level of communication connectivity among them. Generally speaking, communication connectivity means that each pair of agents can exchange information directly or aided by other agents. The application areas ranges from industrial to civilian areas, such as simultaneous localization and mapping, mobile sensor networks and intelligent transportation systems. The main reason for proliferation of multi-agent research is their ability to tackle certain tasks more efficiently than single agents. Moreover, multi-agent systems have advantages over single systems in applications, such as scalability, high adaptivity, and easy maintenance [14].

With cooperative control, multi-agent systems can perform tasks more efficiently, flexibly and robustly. For example, in surveillance, monitoring, rescue and detection tasks, they can cover an area within shorter time and can tolerate failure of some robots without compromising the mission. However, in order to realize these potential advantages, there are many challenges in design and control of multi-agent systems. These challenges are caused by the intrinsic large scale, complex and diverse features of the systems. Next, we will focus on the challenges and advantages brought by application of multi-agent (especially multi-robot) systems.

Specific features of multi-agent systems impose a number of design challenges, such as the communication ability between the robots, battery or fuel constrains and complex structure. A number of different problems were considered in the related literature, such as stability analysis with communication protocols [37,56], communication connectivity during movement [15,19,23,24,38,60], planning with battery constraints [11,47,51], coverage with energy constraints [39, 64, 65], and so on. At the same time, the advantages that stem from cooperation of multiple agents are also studied in different scenarios, for example, cooperative source localization [21, 28, 43, 48, 57, 62], cooperative navigation [16, 20, 22, 30, 34, 55, 58, 67, 68], coverage, monitoring and surveillance with multiple robots [44, 47, 51, 70, 74], and so on.

### 1.1.3 Motivation

Communication connectivity is a prerequisite for cooperation and, hence, it is essential for a number of important problems arising in multi-agent systems. Indeed, cooperation in multi-agent system is an enabling mechanism for improving efficiency in many tasks, such as surveillance of a given area or platooning of driver-less vehicles.

The problem of communication connectivity has received a significant attention in multi-agent (especially multi-robot) systems [42, 59–61, 72, 73]. While a number of issues in communication connectivity has been well understood, there are a number of open questions. One such question is addressed in this thesis: a subset of agents (clients) does not cooperate with the rest of the agents (routers) and the routers are supposed to maintain the communication connectivity under certain restrictions (assumptions) on the clients movement. This scenario is common in practice. For instance, a set of bulldozers (clients) may need to clean a bounded region in a cooperative fashion and a number of drones (routers) is used to maintain communication connectivity for clients. In such a scenario, the clients need to work together to complete their task but they do not restrict their movement for the sake of communication connectivity. On the other hand, the routers work cooperatively to maintain the connectivity based on the clients' movement.

## 1.2 Literature Review

In this section, we present a literature overview of research on the communication connectivity in multi-agent systems. Multi-agent systems have been a prominent research topic for several decades and many problems have been studied in this context, such as consensus, formation control, source localization, cooperative navigation and surveillance. In most cases, an underlying assumption is that there exists a reliable wireless communication connection that can provide information exchange between multiple agents. This requirement is essential in any task that requires cooperation between the agents.

Maintaining a reliable communication connection in a multi-agent system is not a trivial problem. There are multiple factors that can affect the reliability of communication. One is the intrinsic characterization of wireless communication channels, such as shadowing, fading and multi-path propagation. There are also others issues in network layer, such as protocol and routing policy. In control and system field, more attention is dedicated to the design of controllers to maintain communication connectivity. When considering controller design, it is common to assume that if two agents are sufficiently close, then they can communication with each other. Thus, control research has concentrated on maintaining the spatial proximity between agents as a means of maintaining the communication connectivity. In this section, we will overview the control research literature that deals with communication connectivity in multi-agent systems. We classify this literature based on the control methods used to maintain communication connectivity.

### 1.2.1 Preliminaries of Graph

In order to make things clear, we first introduce a graph corresponding to communication relationship of multi-agent systems and present some definitions that will be used throughout this thesis.

An undirected graph $G = (\mathcal{V}, \mathcal{E})$ consists vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. The elements of $\mathcal{E}$ are in the form $\{i, j\}$, for some $i \in \mathcal{V},\ j \in \mathcal{V}$, that is, two elements subsets of $\mathcal{V}$. In a graph corresponding to a multi-agent system, $\mathcal{V}$ is the set of all indices of agents, and $\mathcal{E}$ captures the communication relationship between agents. An edge exists between two agents if they can communicate with each other directly. Such an edge is also termed as a communication link. For example, in the system in Figure 1.2, with the agents being indexed from 1 to 7, $\mathcal{V} = \{1, 2, \dots, 7\}$, $\mathcal{E} = \{\{1,2\}, \{2,4\}, \{3,4\}, \{3,5\}, \{4,5\}, \{4,6\}, \{5,7\}\}$. A vertex $i$ is adjacent to vertex $j$ if $\{i, j\} \in \mathcal{E}$, and $j$ is called a neighbour of $i$. The neighbourhood of vertex $i$ is defined as $\mathcal{N}_i := \{j | \{i, j\} \in \mathcal{E}\}$. A path $p_i$ of length $l$ in $G$, which is $p_i = \{i_1, \dots, i_{l+1}\}$, is a sequence of $l + 1$ distinct vertices $i_1, \dots, i_{l+1}$ such that for

$k = 1, \ldots, l$, the vertices $i_k, i_{k+1}$ are adjacent. Here $i_1, i_{l+1}$ are called end vertices of path $p_i$ [36]. Vertex $i$ is connected to $j$ if there exists at least one path in $G$ whose end vertices are $i$ and $j$. When the vertices of a path are distinct except for its end vertices, the path is called a cycle. The graph $G$ is connected if, for every pair of vertices in $\mathcal{V}$, there is a path with them as its end vertices [36]. A connected graph without cycles is called a tree, denoted by $T$. Leaves of a tree are the vertices which have only one neighbour. A graph $G' = (\mathcal{V}', \mathcal{E}')$ is a subgraph of $G = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. These definitions and more details related to graph can be found in [26,36]. We also introduce "one-hop link" which means that two vertices (agents) connect directly (that is, an edge in $\mathcal{E}$), and "$n$-hop link" which means two agents communicate through a path containing $n+1$ vertices [71].

Moreover, a weighted graph is defined as $G_w = (\mathcal{V}, \mathcal{E}, A)$, where $(\mathcal{V}, \mathcal{E})$ is a graph, and the non-negative matrix $A \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix with $N$ being the number of vertices [13]. The entry $a_{ij}$ of $A$ satisfies: $a_{ij} > 0$ if $\{i, j\} \in \mathcal{E}$, and $a_{ij} = 0$ otherwise. When a weighted graph is corresponding to the communication relationship in a multi-agent system, the value of $a_{ij}$ captures the communication quality of the link $\{i, j\}$. Typically, in control research, $a_{ij}$ decreases with the distance between agents $i$ and $j$. Furthermore, the weighted Laplacian matrix of $G_w$ is defined as $L = diag(A\mathbf{1}) - A$, where $\mathbf{1}$ is a column vector with all ones. Readers may refer to [26] for more details.

### 1.2.2 Early Work without Controller Design

Though the algorithms for maintaining limited range between agents due to visual sensors range were introduced in [8], communication connectivity was not formally considered until [59]. In early work on this topic, only analysis of connectivity was addressed and controller design for maintaining connectivity maintenance was not considered. Some examples can be found in [59] and [60], where connectivity is analysed by using a communication model based on distances between robots. The purpose of the study is finding methods to measure the connectivity and to avoid losing connectivity when designing motion controllers for agents. However, These early papers did not consider control laws of positioning agents on desired positions to maintain the connectivity.

### 1.2.3 Communication Link Connectivity Control

More recently, researchers began to pay attention to the communication link connectivity control problems. Different kinds of approaches were applied to the analysis and controllers design for such problems. These approaches can be divided into two main categories: local connectivity and global connectivity control [54]. These two approaches are defined as follows:

- Local connectivity control problem: when a one-hop link exists at the initial time instant $t = 0$, it will exist for all $t \geq 0$.
- Global connectivity control problem: links can be added or deleted when necessary as long as the graph corresponding to the communication connectivity is connected.

#### Local Connectivity Control

Local connectivity control can be found in [8] and [29, 31, 45]. The approach in [8] did not formally address communication between agents, but it provides a solution for local connectivity while only limited communication range was considered for the communication model. In [45], an un-weighted undirected graph is used to describe the communication connectivity. In this study, the agents with second order dynamics maintain all the one-hop links using a designed controller. That is, maintain the direct links between any two robots connected all the time. Rendezvous and formation control problem are considered with connectivity maintenance control in [31]. In this paper, a weighted undirected graph is used to model the communication connections between agents. The Laplacian matrix $L$ of the corresponding graph $G$ was utilized to determine the connectivity of the graph. This uses the fact the graph is connected if and only if the second smallest eigenvalue $\lambda_2$ of the Laplacian matrix is greater than 0 (see [13] for more details). Based on the value of $\lambda_2$, a control law is designed to guarantee the connectivity by maintaining current links and to add more links to the graph when necessary. The designed controllers maintain the communication connectivity and can also achieve either rendezvous or desired formation. Connectivity is also considered for surveillance and reconnaissance tasks in [29]. Here, the agents perform some searching tasks while maintaining point-to-point communication, that is, one-hop link communication.

The papers above all deal with distributed control design or decentralized control design that each robot can implement individually to perform its own task and maintain connectivity simultaneously. The control laws for these problems are direct because they just need to maintain links between neighbouring two robots. Moreover, the tasks they need to complete are typically not complex, such as consensus, formation. However, the mobility freedom of the agents is extremely restricted because they need to maintain all node-to-node links all the time. Because of this reason, local connectivity control design severely restricts the agents in performing more complex tasks.

**Global Connectivity Control**

Global connectivity control can provide more freedom for the robots to move around to complete more complex tasks. One method to address this problem is called $k$-connectivity maintenance as in [71]. The $k$-connectivity is a property where any two vertices can be connected by a path that includes less than $k+1$ vertices all the time. This method addresses the connectivity of the corresponding graph in continuous time based on the fact that a graph is $k$-connected if and only if all elements of $I + A + \cdots + A^k$ are greater than 0, where $A$ is the weighted adjacency matrix of the graph.

Apart from the $k$-connectivity, another method uses the second smallest eigenvalue $\lambda_2$ of Laplacian matrix $L$ to design connectivity control. Examples of this type of method can be found in [19,54,69,72]. Typically in these papers, first, edge weights $a_{ij}$ are assigned to capture the dependence of $\lambda_2$ on the agents' positions (this is because $L$ is derived from the adjacency matrix $A$). Then, the connectivity control is designed based on the expression of $\lambda_2$ which is a function of agents' positions. Methods in the literature differ from each other in the design of edge weights and controllers. For example, in [72], weights of edges are assigned as $a_{ij}(\mathbf{x}(t)) = \sigma_w(\epsilon - \|x_i(t) - x_j(t)\|)$, where $x_l(t)$ is the position of agent $l$ at time $t$, $\mathbf{x}(t) = [x_1^\intercal(t) \dots x_N^\intercal(t)]^\intercal$,$\epsilon > 0$ is a design parameter, $\sigma_w(y) = \frac{1}{1+e^{-wy}}$, $w > 0$ is a design parameter. With the designed edge weights, a gradient distributed controller is designed to guarantee that $\lambda_2 > 0$. In [69], $a_{ij}$ is assigned as

$$a_{ij}(\mathbf{x}(t)) = \begin{cases} \frac{e^{-\|x_i(t)-x_j(t)\|^2}}{2\delta^2}, \ \|x_i(t) - x_j(t)\| < r \\ 0, \ otherwise \end{cases}.$$

However, the main contribution in this paper is that it finds the derivative of $\lambda_2$ with respect to each agent's position $x_i$. Based on the derivative, a distributed controller is designed to maintain that $\lambda_2 > 0$. A similar idea and method are also addressed in other papers such as [19, 54]. Approaches in these two papers also use the same derivative of $\lambda_2$ as in [69]. Moreover, a potential function of $\lambda_2$ is designed to make the gradient controller robust to the estimation error of $\lambda_2$.

Though the literature above does not explicitly express that existing edges can be deleted and added, distances of some edges can become large as long as the whole graph is in a good connectivity condition (that is, $\lambda_2$ is large enough). This approach using the value of $\lambda_2$ to measure the communication connectivity gives more freedom for agents to move around and perform some pre-assigned tasks.

A slightly different approach to global connectivity control of a multi-agent system is presented in [38]. It gives a rule of adding and deleting links based on agents' positions. In this approach, a set of potential functions are defined as $V(\|x_i - x_j\|) = \frac{1}{\|x_i-x_j\|^2} + \frac{1}{R-\|x_i-x_j\|^2}$, and the gradients of $V$ are used to derive control inputs to maintain connection between neighbouring two agents. Furthermore, a link (an edge) is added to $G$ if the distance between the corresponding two agents is less than $r$. Similarly, a link $\{i, j\}$ can be deleted if its length satisfies $r < \|x_i - x_j\| < R$[1] and the deletion of it will not violate the connectivity of $G$, here $R$ is the communication limit.

Compared with the local connectivity control, the agents in the system using global connectivity control have more freedom to move around without breaking communication connectivity. Therefore, global connectivity control method equips the agents with potential to complete more complex pre-assigned tasks. However, this method still needs all agents to cooperatively maintain communication connectivity, which still affects the agents' ability to complete their own missions.

[1]In this thesis, the norm $\|\cdot\|$ denotes the Euclidean norm.

### 1.2.4 Communication Connectivity Control for Free Mobility

Compared with the approaches stated above, the approach in this thesis gives a subset of agents total freedom to perform their pre-assigned tasks. In such an approach, the agents are split into two groups, clients and routers. Clients have their corresponding pre-assigned tasks while routers are obliged to provide communication connectivity for clients. The feature of this approach is that it permits the clients to perform their own tasks without worrying about the communication between each other. Therefore, this method gives more freedom to clients to perform complex tasks. However, there are few papers addressing this problem because it is difficult to guarantee communication connectivity only by controlling routers. Some related examples can be found [23–25]. In [25], aerial vehicles (routers) were used to provide optimal communication connection for ground vehicles (clients). First, the ground vehicles are treated as stationary and controllers are designed for the aerial vehicles to provide communication connection between the ground vehicles. Then, movement of the ground vehicles is treated as disturbance and it was shown that the controllers designed are still effective for communication connection. In this example, the ground vehicles are much slower than aerial ones. Some general cases can be found in [23] and [24], where clients were not assumed to be much slower than the routers. The solution in [23] is based on reachability analysis with the multi-agent system in discrete time form. The communication connectivity problem in this paper is addressed by finding the routers' positions which maximize the time length during which the communication connectivity is maintained under the movement of clients. A practical way illustrated by experiments is presented in [24], where communication connection is assumed existing all the time. The objective here is to minimize the communication rate discrepancy on each one-hop link. An on-line device is used to measure the communication signal strength, and control laws are designed for routers to achieve the objective based on the measurement.

The approach above can provide free mobility for clients to perform their pre-assigned tasks, but the solutions are still restrictive. The example in [25] did not consider the movement of clients when designing controllers for routers, which restricted the implementation of the method, and communication connectivity was not theoretically proven in [23,24] . The main goal of this thesis is to address these issues in more detail.

## 1.3 Problem Formulation

In this thesis, we design control policies to maintain the communication connectivity in multi-agent systems. Inspired by the literature in Section 1.2, we consider the situation where there are two different groups of agents, clients and routers, in the systems. The control policies are designed for routers to ensure communication connectivity for clients as they are moving around with their pre-assigned control objectives.

A general structure of the multi-agent systems we consider is illustrated in Figure 1.3. Compared with the multi-agent system in Figure 1.2, two types of circles with different

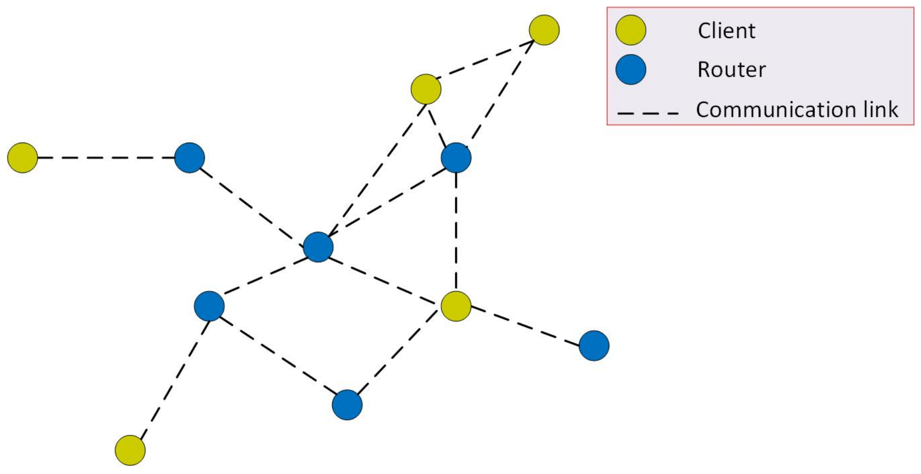


Figure 1.3: Multi-agent system with clients and routers

colours are used to differentiate clients and routers, and dashed lines still denote communication links. In the multi-agent system we consider, the agents are moving robots. Clients move in a time-varying bounded area, and they are assumed to have their own control objectives based on their pre-assigned tasks. For example, their objectives might include monitoring an area of interest, or tracking some moving objectives. Routers, on the other hand, are solely responsible for ensuring communication connectivity of clients, that is, to make sure any two clients can communicate with each other. Ensuring communication connectivity is mathematically modelled by ensuring that a communication path exists between any two clients, as formalized later in this Section.

In the system, we assume that each agent is uniquely indexed by an integer number from $\mathcal{V} = \{1, \ldots, N\}$, where $N$ is the total number of agents in the system. The set of indices of clients is denoted by $\mathcal{C}$, and the set of indices of routers is denoted by $\mathcal{R}$. The number of clients is denoted by $N_c$, and the number of routers is denoted by $N_r$. Note that $\mathcal{V} = \mathcal{C} \cup \mathcal{R}$ and $\mathcal{C} \cap \mathcal{R} = \emptyset$. Let $x_i \in \mathbb{R}^n$ denote the position of agent $i$, where $n = 2$ or 3. The two choices of $n$ are depend on different types of agents. For example, if the agents are ground vehicles, $n$ is set as 2; if the agents are UAVs, $n$ is set as 3. We assume that positions $x_i$ is governed by the following general dynamics for every agent $i$,

$$\begin{aligned} \dot{\xi}_i &= \mathcal{F}_i(\xi_i, u_i) \\ x_i &= \mathcal{H}_i(\xi) \end{aligned} \tag{1.1}$$

where $\xi_i \in \mathbb{R}^{n_i}$ are the states of dynamics of agent $i \in \mathcal{V}$, $\mathcal{F}_i : \mathbb{R}^{n_i} \times \mathbb{R}^{r_i} \to \mathbb{R}^{n_i}$ and $\mathcal{H}_i : \mathbb{R}^{n_i} \to \mathbb{R}$ are differentiable, and $u_i \in \mathcal{U} \subset \mathbb{R}^{r_i}$ with $n_i$ and $r_i$ being positive integers. States $\xi_i$ may include position, velocity, and acceleration of agent $i$. In the problem considered, the control policies of clients $u_i$, $\forall i \in \mathcal{C}$, are unknown and need to be designed to achieve their own control objectives.

In this thesis, we assume the communication between agents only depends on the distances between them. The following assumption characterises the condition under which two agents $i \in \mathcal{V}$ and $j \in \mathcal{V}$ can communicate with each other.

**Assumption 1.1.** *Agent $i \in \mathcal{V}$ and $j \in \mathcal{V}$ can communicate with each other directly at time $t$ if $\|x_i(t) - x_j(t)\| \leq R$, where $R$ is a prescribed positive real number that represents the* maximum communication range *between any pair of agents.*

In some literature, see Section 1.2, models of the communication are typically considered with the spatial proximity between agents. Without loss of generality, we present this communication model to indicate that communication exists between two agents when they are within a certain distance.

The above assumption leads to an undirected graph $G_o(t) = \big(\mathcal{V}, \mathcal{E}_o(t)\big)$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}_o(t)$, where $\{i, j\} \in \mathcal{E}_o(t)$ if $\|x_i(t) - x_j(t)\| \leq R$. With this induced graph $G_o(t)$, the following gives the definition of persistent connectivity of clients.

**Definition 1.1** (Persistently Connected Clients)**.** *We say that clients are* persistently connected *if for any $i \in \mathcal{C}$, $j \in \mathcal{C}$ there exists a path in $G_o(t)$ between them for any $t \geq t_0$.*

In the definition, $t_0$ denotes the initial time instant of the system we consider. Note that, except the two end vertices of the path connecting $i$ and $j$, $i, j \in \mathcal{C}$, any other vertex on the path can either be a router or a client. For example, paths with clients and routers between the two end vertices can be found in Figure 1.3.

Obviously if the clients are allowed to move in an unbounded area, it is not possible to ensure that they remain persistently connected for a given finite number of routers. Thus, the area within which the clients can move is assumed to be bounded.

**Assumption 1.2.** *The clients move within a time-varying bounded area $B(t)$ all the time, that is, $x_i(t) \in B(t)$, $\forall i \in \mathcal{C}$, $\forall t \geq t_0$, where $t_0$ is the initial time instant.*

In broad terms the objective is to design closed loop control policies $u_i(t)$, $\forall i \in \mathcal{R}$ to guarantee that clients are persistently connected. To make the system scalable, we want to design such control policies only by using locally available information. Thus, we make the following assumption on information exchange among agents.

**Assumption 1.3.** *Each agent $i$ can access the information available from its neighbours in $\mathcal{N}_i^o(t)$ and itself at time instant $t$.*

Here, $\mathcal{N}_i^o$ is the neighbourhood of $G$ as defined in subsection 1.2.1, that is, $\mathcal{N}_i^o = \left\{ j | \{i, j\} \in \mathcal{E}_o \right\}$.

In order to guarantee persistent connectivity of clients, we need an assumption on the connectivity property of the multi-agent system at the initial time instant $t_0$.

**Assumption 1.4.** *At the initial time instant $t = t_0$, $G^o(t_0)$ is connected.*

With the dynamic models of agents and assumptions above, we present a general description of our problem throughout this thesis.

**Problem 1.1.** *With the dynamic model of agents in (1.1), $N_c$ clients and $N_r$ routers, under the Assumptions 1.1–1.4, design control policies $u_i(t) \in \mathcal{U}$, $\forall i \in \mathcal{R}$, such that clients are persistently connected.*

Throughout this thesis, we will specify the dynamic models of agents in (1.1) when considering different cases of multi-agent systems with specific dynamic models of agents. Moreover, we will also specify additional for different cases of multi-agent systems. In this thesis, we will address Problem 1.1 in two cases. One case considers the problem in a system with two clients. In such a system, the structure of the corresponding graph is simple. The other case considers the problem in a general system with an arbitrary number of clients. Such a general system has a complex structure of the corresponding graph, which makes the control policy design more complicated.

## 1.4 Contributions

We consider Problem 1.1 throughout the thesis under different assumptions. The control policies we design are proven to solve this problem under some additional assumptions. One main feature of our control policies is that they are satisfied for different types of dynamic model. This means that the solutions can be applied to different types of robots when we consider practical multi-agent systems with agents being robots. For example, these robots can be UAVs, UUVs, UGVs. Moreover, experiments are set up with real robots to test the effect of our control policies, which verify that the policies are effective in implementation.

## 1.5 Outline of the Thesis

In this thesis, we consider guaranteeing clients persistently connected in two types of systems. In Chapter 2, we consider a simple type of multi-agent systems. In such a system, there are only two clients. We design controllers for routers to make sure that the clients are persistent connected according to a simple static graph structure. This work was originally presented in

- Zhiyang Ju, Iman Shames, Dragan Nešić: Ensuring Communication Connectivity in Multi-agent Systems in the Presence of Uncooperative Clients. Decision and Control (CDC), 2016 IEEE 55th Conference on. December 2016.

In Chapter 3, a general type of system with arbitrary number of clients is considered. The structure of the corresponding graph is more complex and time-varying in such a system. In order to guarantee persistent connectivity of clients, the control policy for routers is designed by considering updating the graph structure when necessary. In Chapter 4, we implement our control policy in experiments. E-puck robots are used to act as routers to implement our policy on. We confirmed out theoretical results in experiments. Finally, conclusions and future work are given in Chapter 5.

# Chapter 2

# Persistent Connectivity with Two Clients

*The problem of maintaining and guaranteeing communication connectivity between a pair of clients via controlling a number of routers is considered. It is assumed that agents satisfy quadrotor dynamics. A set of controllers are proposed and it is shown that these controllers solve the problem exponentially fast under a set of mild assumptions. The simulation results illustrate the effectiveness of the proposed controllers.*

## 2.1 Introduction

PERSISTENT connectivity of two clients is considered in this chapter, which is a simple case of Problem 1.1 with only two clients existing in the system. An example to illustrate such a system is shown in Figure 2.1. Though simple, this kind of multi-

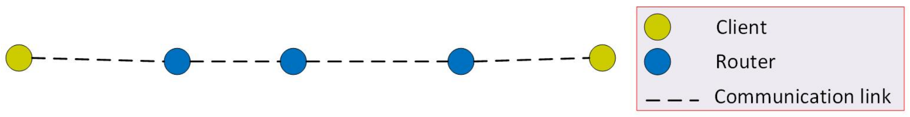


Figure 2.1: A multi-agent system example with two clients

agent systems is commonly encountered in practice. For example, in scenarios such as searching and rescue after flood or forest fire, one client is a moving robot quipped with expensive sensors for gathering information, and the other client is a mobile base for receiving useful information. In such an application, the two clients are usually far away from each other such that the communication between them needs to be aided by routers in the system. Moreover, the approach in this chapter can still be applied to guarantee persistent connectivity of more clients if the systems are in a star structure as illustrated in Figure 2.2 with five clients and eight routers. Note that in such systems, one client

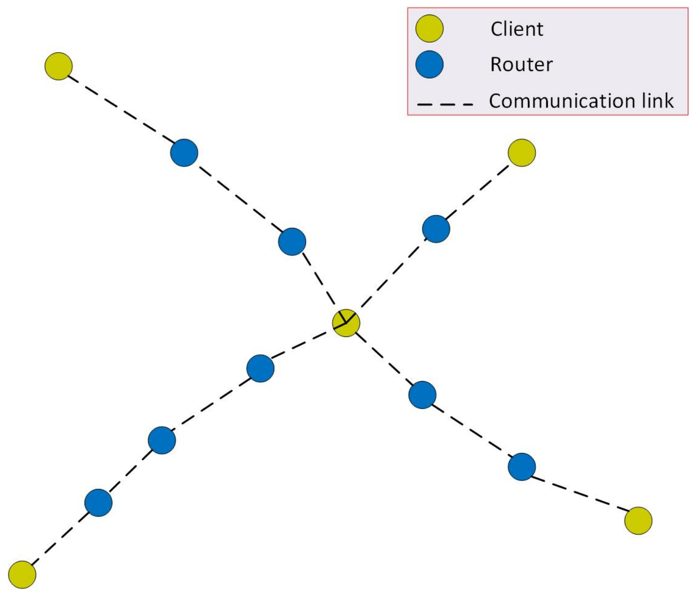


Figure 2.2: A multi-agent system example with five clients

acts as the base which connects all the other clients via separate paths. Therefore, this approach can be extended to be an alternative way of solving Problem 1.1 with additional assumptions.

In the system we consider in this chapter, the clients and routers are quadrotors, which are commonly used in multi-agent systems. We note that the clients models here are assumed to be quadrotors for the sake of the simplicity of exposition, however, they can be assumed to have any dynamics as long as their positions, velocities, and accelerations can be measured (or estimated) by the routers. First, we choose a path from the induced graph, which contains all the agents with the end vertices being the two clients. Then, we design controllers for routers based on the selected path. Furthermore, we analytically establish that the proposed controllers solve the persistent connectivity problem exponentially fast under a set of suitable assumptions.

This chapter is organized as follows. In Section 2.2, the problem we consider is formulated. The controllers and the main results are given in Section 2.3. Simulation results are presented in Section 2.4 to verify the applicability of the proposed controllers and validity of the theoretical results. In the end, concluding remarks and future directions are

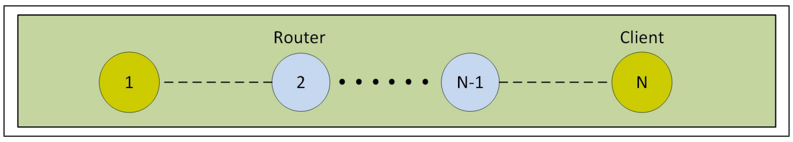


Figure 2.3: An example of the multi-agent system with two clients

presented.

## 2.2 Problem Statement

We adopt the feedback linearised model of quadrotor in [9] throughout this chapter. Consider $N$ quadrotors in $\mathbb{R}^3$ where the feedback linearised model of each quadrotor $i \in \mathcal{V} = \{1, \ldots, N\}$ is given by

$$\dot{x}_i = v_i, \tag{2.1}$$
$$\dot{v}_i = -g\bar{c}v_i + \frac{1}{m}F_i, \tag{2.2}$$
$$\dot{F}_i = -\beta_q F_i + u_i, \tag{2.3}$$

where $x_i \in \mathbb{R}^3$ denotes the position of quadrotor $i$, $v_i \in \mathbb{R}^3$ denotes its velocity, $g$ is the gravity constant, $\bar{c}$ is a constant related to drag force during flight, $m$ is the mass of the quadrotor, $F_i \in \mathbb{R}^3$ is a variable related to lift generated by rotors and attitudes of the quadrotor $i$, $\beta_q$ is a constant, and $u_i$ is the control input.

The assumption on the maximum communication range of the quadrotors is as stated in Assumption 1.1, where agents $i$ and $j$ are quadrotors in the specific multi-agent system in this chapter. We recall that this assumption leads to the induced graph $G_o = \{\mathcal{V}, \mathcal{E}_o\}$, and $\mathcal{V}$ is partitioned into two subsets $\mathcal{C}$ and $\mathcal{R}$.

In broad terms the main objective is to ensure that there is always a path between any pair $i, j \in \mathcal{C}$ via controlling the positions of the routers. In the rest of this chapter, we assume that $\mathcal{C} = \{1, N\}$ and $\mathcal{R} = \{2, \ldots, N-1\}$. The system with two clients is illustrated in Figure 2.3. We assume that at time $t = 0$, $\mathcal{N}_i^o$ satisfies the following assumption.

**Assumption 2.1.** *Let $\overline{\mathcal{M}}_i = \{i-1, i+1\}$, for $i = 2, \ldots, N$. Then at the initial time instant $t = 0$, $\overline{\mathcal{M}}_i \subset \mathcal{N}_i^o$, for $i = 2, \ldots, N-1$.*

Note that, in this chapter, we assume the initial time instant $t_0 = 0$.

The objective in this work now is to find $u_i$ for all $i \in \mathcal{R}$ such that the two clients are persistently connected aided by the routers. In order to achieve the objective, we have the following assumptions on the state measurement and information exchange between agents, which specifies Assumption 1.3.

**Assumption 2.2.** *Each agent i measures its own state, i.e.* $\left[x_i^{\mathsf{T}}, v_i^{\mathsf{T}}, F_i^{\mathsf{T}}\right]^{\mathsf{T}}$*, and receives* $\left[x_j^{\mathsf{T}}, v_j^{\mathsf{T}}, F_j^{\mathsf{T}}\right]^{\mathsf{T}}$ *from* $j \in \overline{\mathcal{M}}_i$*.*

*Remark:* Note that if $v_j$ and $F_j$, $j \in \overline{\mathcal{M}}_i$, are not directly available then $i$ may estimate them by measuring $x_j$ and receiving $u_j$ instead. As stated earlier the main problem of interest is to ensure a connected path from client 1 to client $N$ with $N-2$ routers. For this to be feasible we need the following assumption on the time-varying bounded area $B(t)$ the clients move within.

**Assumption 2.3.** *Let* $\sup_t \|x_1(t) - x_N(t)\| \leq \bar{\delta}$*, for some bounded real value* $\bar{\delta}$*. Then* $N - 1 \geq \dfrac{\bar{\delta}}{R - \bar{\Delta}}$*, where* $\bar{\Delta}$ *is a design parameter.*

This assumptions gives specific requirement on the bounded area $B(t)$ as in Assumption 1.2, and we will provide explicit bounds on $\bar{\Delta}$ later in this chapter. The following assumption provides a limit on the maximum speed of the clients.

**Assumption 2.4.** *Let* $v_{im} = \sup_t \|v_i(t)\|$*,* $i \in \mathcal{C}$*, then* $\max_{i \in \mathcal{C}}\{v_{im}\} \leq v_{cm}$*, where* $v_{cm}$ *is some bounded positive real value.*

In order to guarantee connectivity for all time $t \geq 0$, the initial positions of the agents should satisfy the following assumption, which is a stronger version of Assumption 1.4.

**Assumption 2.5.** *If N is an even number, then it can be written as* $N = 2n$ *for some integer n. It is assumed that at time* $t = 0$ *the following statements are true:*

$$\|x_n(0) - x_{n+1}(0)\| \leq \frac{1}{3}\|x_{n-1}(0) - x_{n+2}(0)\| \tag{2.4a}$$

$$\begin{aligned}\|x_j(0) - x_{2n+1-j}(0)\| \leq \frac{1}{2}(\|x_{j-1}(0) \\ - x_{2n+2-j}(0)\| + \|x_{j+1}(0) - x_{2n-j}(0)\|)\end{aligned} \tag{2.4b}$$

$$\|x_1(0) - x_2(0)\| \leq \|x_2(0) - x_3(0)\| + \frac{1}{\beta}v_{cm} \tag{2.4c}$$

$$\|x_k(0) - x_{k+1}(0)\| \leq \frac{1}{2}(\|x_{k-1}(0) - x_k(0)\|$$

$$+ \|x_{k+1}(0) - x_{k+2}(0)\|) \tag{2.4d}$$

$$\|x_{N-1}(0) - x_N(0)\| \leq \|x_{N-2}(0) - x_{N-3}(0)\| + \frac{1}{\beta}v_{cm} \tag{2.4e}$$

*where $k = 2, \ldots, n-1, n+1, \ldots, N-2$, $j = 2, \ldots, n-1$, and $\beta$ is a positive real number which will be defined later. If $N$ is an odd number, then it can be written as $N = 2n-1$ for some integer $n$. It is assumed that at time $t = 0$ the following statements are true:*

$$\|x_{n-1}(0) - x_{n+1}(0)\| \leq \frac{1}{2}\|x_{n-2}(0) - x_{n+2}(0)\|, \tag{2.5a}$$

$$\|x_j(0) - x_{2n-j}(0)\| \leq \frac{1}{2}(\|x_{j-1}(0)$$

$$- x_{2n+1-j}(0)\| + \|x_{j+1}(0) - x_{2n-1-j}(0)\|), \tag{2.5b}$$

$$\|x_1(0) - x_2(0)\| \leq \|x_2(0) - x_3(0)\| + \frac{1}{\beta}v_{cm}, \tag{2.5c}$$

$$\|x_k(0) - x_{k+1}(0)\| \leq \frac{1}{2}(\|x_{k-1}(0) - x_k(0)\|$$

$$+ \|x_{k+1}(0) - x_{k+2}(0)\|), \tag{2.5d}$$

$$\|x_{n-1}(0) - x_n(0)\| \leq \frac{1}{3}\|x_{n-1}(0) - x_{n+1}(0)\|$$

$$+ \frac{1}{3}\|x_{n-2}(0) - x_{n-1}(0)\|, \tag{2.5e}$$

$$\|x_n(0) - x_{n+1}(0)\| \leq \frac{1}{3}\|x_{n-1}(0) - x_{n+1}(0)\|$$

$$+ \frac{1}{3}\|x_{n+1}(0) - x_{n+2}(0)\| \tag{2.5f}$$

$$\|x_{N-1}(0) - x_N(0)\| \leq \|x_{N-2}(0) - x_{N-1}(0)\| + \frac{1}{\beta}v_{cm} \tag{2.5g}$$

*where $k = 2, \ldots, n-1, n+1, \ldots, N-1$, $j = 2, \ldots, n-1$, and the same as in the previous case $\beta$ is a positive real number.*

An illustration depicting the distances considered in Assumption 2.5 can be found in Fig. 2.4.

*Remark:* Assumption 2.5 is easily satisfied for small $\bar{\delta}$ as defined in Assumption 2.3

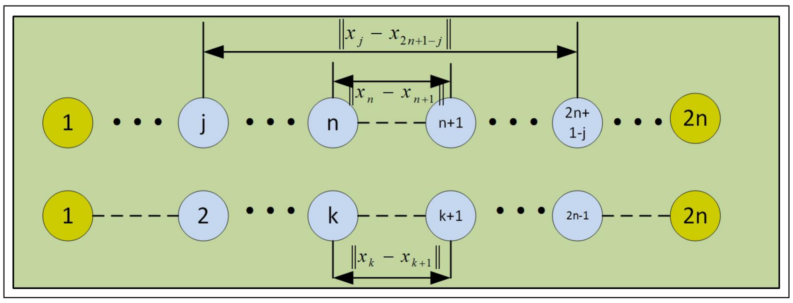


Figure 2.4: Initial conditions for $x_i$ when $N = 2n$

by making $\|x_i(0) - x_{i+1}(0)\| = \|x_{i+1}(0) - x_{i+2}(0)\|$, $i = 1, \dots, N-2$.

Then the problem can be formalized as the following.

**Problem 2.1.** *Consider $N$ agents in set $\mathcal{V}$ where the dynamics of each agent $i$ is governed by* (2.1). *Under Assumptions 1.1 and 2.1–2.5, find $u_i$ for all $i \in \mathcal{R}$ such that there exists a path between agent* 1 *and $N$ for all $t \geq 0$, that is the two clients are persistently connected.*

*One way to solve this problem is to guarantee following conditions are satisfied $\forall t \geq 0$.*

$$\|x_i(t) - x_{i+1}(t)\| \leq R, \tag{2.6}$$

*where $i = 1, \dots, N-1$.*

*Remark:* The way of solving Problem 2.1 is equivalent to designing controllers such that the trajectories remain in the non-compact set that characterizes the desired states, i.e. the states that result in the existence of a path connecting 1 and $N$, if Assumptions 2.1–2.5 and 1.1 hold.

## 2.3 Controller Design and Analysis for Quadrotor Model

In this section, we first give the designed controller and give the results of persistent connectivity of clients in Theorem 2.1. In order to prove the theorem, we then give two lemmas, one is on the relationship between an auxiliary system and the original system, the other one is about the connectivity results of the auxiliary system. Finally, the proof of Theorem 2.1 is given based on the two lemmas.

For a multi-agent system with $N$ quadrotors whose dynamics are as in (2.1), we design $u_r = [u_2^\intercal, \dots, u_{N-1}^\intercal]^\intercal$ as follows

$$\begin{aligned} u_i =& \beta_q F_i + m[(-\beta_r + g\bar{c} - 2\beta)a_i + \beta(a_{i-1} + a_{i+1}) \\ & - \beta_s(2v_i - v_{i-1} - v_{i+1})] - \beta_F e_F^i, \end{aligned}$$

where $i = 2, \dots, N-1$, $a_i = \dot{v}_i = -g\bar{c}v_i + \frac{1}{m}F_i$, $\beta_q$, $\beta_r$, $\beta_s$, $\beta_F$ are positive real constants, $\beta = \frac{\beta_s}{\beta_r}$, $e_F = \left[e_F^{2\intercal}, \dots, e_F^{N-1\intercal}\right]^\intercal$, where

$$\begin{aligned} e_F^i =& F_i - m\Big[ - \beta_r v_i - \beta_s(x_i - x_{i-1}) - \beta_s(x_i - x_{i+1}) \\ & + g\bar{c}v_i - 2\beta(v_i - \frac{v_{i-1} + v_{i+1}}{2})\Big] \end{aligned}$$

with $i = 2, \dots, N-1$.

By using these control inputs, we have following results regarding the positions of quadrotors.

**Theorem 2.1.** *Suppose Assumptions 1.1 and 2.1 – 2.5 hold. Using controllers $u_r$ in (2.7), the solutions $x_i(t)$ of (2.1) satisfy the conditions in (2.6), $\forall t \geq 0$ where $\bar{\Delta} \geq \frac{2\beta}{N} v_{cm} + 2K_0$ for some positve $K_0$.*

The proof of this theorem is obtained via applying two lemmas that will be introduced next. The first lemma involves establishing that the solution $x_i$, $i \in \mathcal{R}$, of multiple quadrotor system (2.1) for an appropriately chosen set of control laws will converge to the solution of Problem 2.1 for an auxiliary system comprised of $N$ single-integrators. Then, in the second lemma the connectivity of the clients in the system of single-integrators is demonstrated. The proof of Theorem 2.1 is then achieved via invoking these two lemmas.

An auxiliary system with $N$ agents with single-integrator dynamics is introduced as follows

$$\dot{\bar{x}}_i = \bar{v}_i, \tag{2.7}$$

where $i = 1, \dots, N$, $\bar{x}_i$ is the position of agent $i$, and $\bar{v}_i$ is its control input. Similar to the assumptions on (2.1) we assume the following for (2.7).

**Assumption 2.6.** *The control inputs $\bar{v}_i$ and states $\bar{x}_i$ in system (2.7) satisfy Assumptions 1.1 and 2.1–2.5 where the agents dynamics are governed by* (2.7) *instead of (2.1).*

For this system, we design $\bar{v}_i,\ i \in \mathcal{R}$ as

$$\bar{v}_i = -\beta(\bar{x}_i - \bar{x}_{i-1}) - \beta(\bar{x}_i - \bar{x}_{i+1}), \tag{2.8}$$

where $i = 2, \dots, N-1$, $\beta$ is the same as in (2.7).

Additionally, it is assumed that the following assumption holds.

**Assumption 2.7.** *For all $t \geq 0$, $\bar{x}_i(t) = x_i(t)$, $\bar{v}_i(t) = v_i(t)$, $i \in \mathcal{C}$.*

*Remark:* This assumption ensures that the clients in the auxiliary system of single-integrators have the same position and velocity of their counterparts in the original system.

We denote

$$e_x = \left[e_x^{2\intercal}, \dots, e_x^{N-1\intercal}\right]^\intercal, \tag{2.9}$$

where,

$$e_x^i = x_i - \bar{x}_i, \tag{2.10}$$

where $i = 2, \dots, N-1$. Now we have the following lemma,

**Lemma 2.1.** *Consider systems in (2.1) and (2.7), suppose Assumptions 1.1 and 2.1 – 2.7 hold. Using controllers in (2.7) and (2.8), the $e_x(t)$ in (2.9) will converge to* 0*. Additionally, there exists some $K_0 > 0$, $\gamma > 0$, such that $e_x(t)$ satisfies $\|e_x(t)\| \leq K_0 e^{-\gamma t}, \forall t \geq 0$.*

*Proof.* See Section 2.5. □

In the following Lemma 2.2, a result of connectivity of clients from the controller (2.8) designed for system (2.7) will be stated.

**Lemma 2.2.** *Suppose Assumption 2.6, 2.7 hold for system (2.7), using controller in (2.8), the solutions of states $\bar{x}_i(t)$ will always satisfy the following conditions $\forall t \geq 0$.*

$$\|\bar{x}_i(t) - \bar{x}_{i+1}(t)\| \leq R - \bar{\Delta}_1, \tag{2.11}$$

*where* $\bar{\Delta}_1 = \bar{\Delta} - \dfrac{2\beta}{N}\|v_{cm}\|$, $i = 1, \dots, N-1$[1].

*Proof.* See Section 2.5. $\square$

After having the results in the two lemmas, we give the proof of Theorem 2.1 as follows.

*Proof.* We already proved in Lemma 2.2 that the controller designed for single-integrator system (2.7) will guarantee states $\bar{x}_i(t)$ satisfy the requirements in (2.11). From Lemma 2.1, the solution $x_i(t)$, $i \in \mathcal{R}$ of (2.1) with controller (2.7) will exponentially converge to $\bar{x}_i$, $i \in \mathcal{R}$. Here we will link these two lemmas to prove the results in this theorem.

From (2.10), we have, $x_i(t) = \bar{x}_i(t) + e_x^i(t)$, where $i = 2, \dots, N-1$. Based on these equations and Assumption 2.7, we have

$$\begin{aligned}
\|x_1 - x_2\| =& \|\bar{x}_1 - \bar{x}_2 - e_x^2\| \\
\le& \|\bar{x}_1 - \bar{x}_2\| + \|e_x^2\|, && \text{(2.12a)} \\
\|x_i - x_{i+1}\| =& \|\bar{x}_i + e_x^i - \bar{x}_{i+1} - e_x^{i+1}\| \\
\le& \|\bar{x}_i - \bar{x}_{i+1}\| + \|e_x^i\| + \|e_x^{i+1}\|, && \text{(2.12b)} \\
\|x_{N-1} - x_N\| =& \|\bar{x}_{N-1} + e_x^{N-1} - \bar{x}_N\| \\
\le& \|\bar{x}_N - \bar{x}_1\| + \|e_x^{N-1}\|, && \text{(2.12c)}
\end{aligned}$$

where $i = 2, \dots, N-1$.

From Lemmas 2.1, for $i = 2, \dots, N-1$, we have

$$\|e_x^i\| \le \|e_x\| \le K_0 e^{-\lambda t}. \tag{2.13}$$

Based on Lemma 2.2 and (2.12), (2.13), for $i = 2, \dots, N-1$,we have

$$\|x_i - x_{i+1}\| \le R - \bar{\Delta}_1 + 2K_0 e^{-\lambda t}, \tag{2.14}$$

[1]Note that $\bar{\Delta}_1 > 0$ because from the statement of Theorem 2.1 it is known that $\bar{\Delta} > \dfrac{2\beta}{N}\|v_{cm}\|$.

By choosing

$$\bar{\Delta} \geq \frac{2\beta}{N} v_{cm} + 2K_0 e^{-\lambda t}, \tag{2.15}$$

based on (2.14), $x_i$ satisfy the conditions in (2.6). $\square$

## 2.4 Simulation Results

In the simulation, we choose parameters from [32] in order to have a result close to the practical quadrotor system. These parameters are $g = 9.8, c = 0.01, m = 1.04, \beta_q = 0.1$. For the parameters of controller designed, we set $\beta = 2.5, \beta_r = 5, \beta_F = 5$. $R$ is set as $R = 20$ which satisfies Assumption 1.1. In the simulation, Assumption 2.2 holds for we will use these information in the simulation.

The clients are governed by the following inputs:

$$u_i = \alpha \begin{bmatrix} x_i \\ v_i \\ F_i \end{bmatrix} + K\left( T \begin{bmatrix} x_i \\ v_i \\ F_i \end{bmatrix} - \begin{bmatrix} x_d^i \\ v_d^i \\ a_d^i \end{bmatrix} \right) + J_d^i,$$

where $i \in \mathcal{C}$, $\alpha = [0_{3\times 3}, 0.0098 I_3, 0.198 I_3]$, $K = [-30I_3, -31I_3, -10I_3]$,

$$T = \begin{bmatrix} 1.04I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 1.04I_3 & 0_{3\times3} \\ 0_{3\times3} & -0.1019I_3 & I_3 \end{bmatrix}, x_d^1 = \begin{bmatrix} 10 + 10\sin(\frac{1}{5}t - \frac{\pi}{2}) \\ 10\cos(\frac{1}{5}t - \frac{\pi}{2}) \\ 5 \end{bmatrix},$$

$$x_d^N = \begin{bmatrix} 53 + 20\sin(\frac{1}{10}t - \frac{\pi}{2}) \\ 20\cos(\frac{1}{10}t - \frac{\pi}{2}) \\ 5 \end{bmatrix}$$

, and $v_d^i$, $a_d^i$, $J_d^i$ are the first, second, and third derivatives of $x_d^i$. Using these $u_i$, $i \in \mathcal{C}$, and set the initial conditions as $x_1(0) = 0$, $x_N(0) = [33, 0, 0]^\intercal$, $v_i(0) = 0, F_i(0) = 0$, $i \in \mathcal{C}$. We obtain $\sup_t \|x_1(t) - x_N(t)\| = 70$, $\|v_i(t)\| \leq 2.5$, $\forall t \geq 0, \forall i \in \mathcal{C}$.

Furthermore, set $\bar{\delta} = 70$, $v_{cm} = 2.5$ with which Assumption 2.4 is satisfied, $N = 5$. We set the initial values of clients as $x_4(0) = \begin{bmatrix} \frac{99}{4} & 0 & 5 \end{bmatrix}^\intercal, x_3(0) = \begin{bmatrix} \frac{33}{2} & 0 & 5 \end{bmatrix}^\intercal, x_2(0) = \begin{bmatrix} \frac{33}{4} & 0 & 5 \end{bmatrix}^\intercal$ and $v_i(0) = 0, F_i(0) = 0, i \in \mathcal{R}$, to satisfy the conditions in Assumption 2.5.

From the initial conditions, we have $K_0 = 0$ in (2.19). And we set $\bar{\Delta} = 2.5$ as in (2.15). We can see that these choices satisfy Assumption 2.3. And in the simulation we label the quadrotors with the number that satisfy the relationships in Assumption 2.1. With these settings, all the Assumptions 1.1–2.5 of this multiple quadrotor system are all satisfied.

The simulation results of distances between neighbouring two quadrotors and the norms of $v_i$ are in Fig. 2.5 and Fig. 2.6.

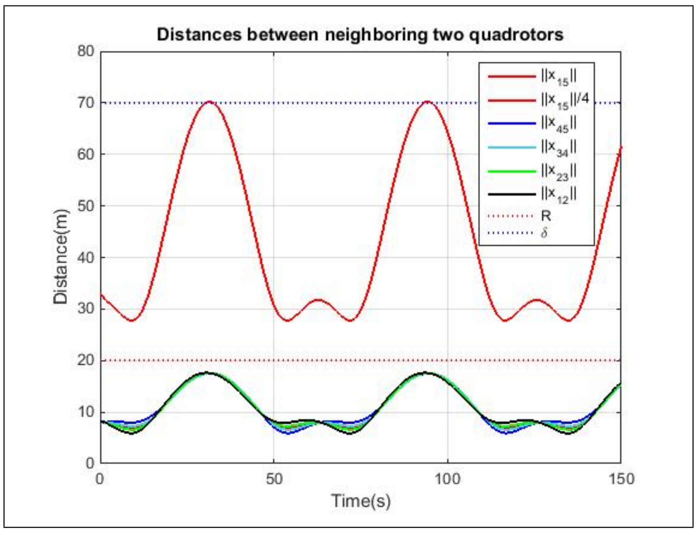


Figure 2.5: Distances between two neighbouring quadrotors

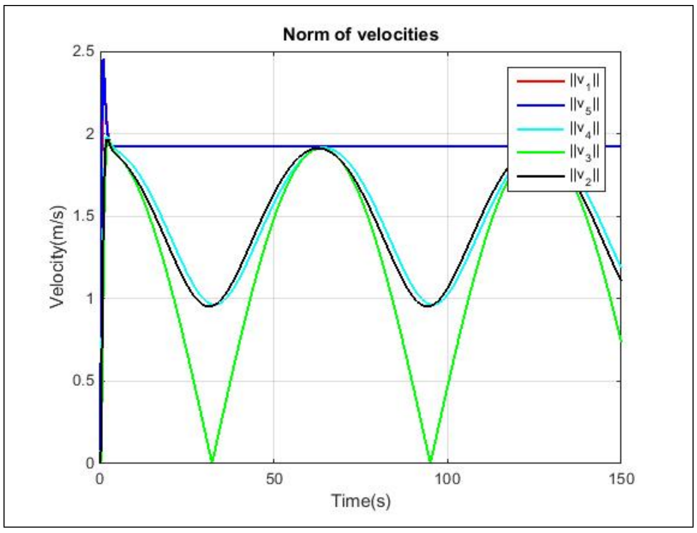


Figure 2.6: Norms of velocities

From the results, we can see that the distance between any two neighbouring quadrotors is less than $R = 20$, $\forall t \geq 0$. This simulation verifies that the controller we designed guarantees the persistent connectivity of clients in the multiple quadrotor system, that is, satisfy the conditions in (2.6). Additionally, in Fig. 2.6, we can see the magnitude of velocities of clients can compare to the magnitude of routers' velocities, which implies that our control law can be applied for maintaining connectivity for comparable fast clients.

## 2.5 Proofs of Main Results

### 2.5.1 Proof of Lemma 2.1

*Proof.* By using the control input $u_r$ as in (2.7), we can calculate the derivatives of $e_F$ in (2.7) as

$$\dot{e}_F = -\beta_F e_F. \tag{2.16}$$

Now the dynamics of the system (2.1) becomes

$$\begin{aligned}
\dot{x}_i =& v_i, \\
\dot{v}_i =& -\beta_r v_i - \beta_s(x_i - x_{i-1}) - \beta_s(x_i - x_{i+1}) \\
& - 2\beta(v_i - \frac{v_{i-1} + v_{i+1}}{2}) + \frac{1}{m} e_F^i, \\
\dot{e}_F =& -\beta_F e_F,
\end{aligned}$$

where $i = 2, \ldots, N-1$.

Furthermore, we define $e_v$ as $e_v = \left[e_v^{2\mathsf{T}}, \ldots, e_v^{N-1\mathsf{T}}\right]^{\mathsf{T}}$, where, $e_v^i = v_i + \beta(x_i - x_{i-1}) + \beta(x_i - x_{i+1})$, where $i = 2, \ldots, N-1$. By taking derivate of $e_v$ and with the system (2.1), we have the dynamics of $e_v$ as, $\dot{e}_v^i = -\beta_r e_v^i + \frac{1}{m} e_F^i$, that is,

$$\dot{e}_v = -\beta_r e_v + \frac{1}{m} e_F \tag{2.17}$$

Then the dynamics of the system can be rewritten as

$$\dot{x}_i = -\beta(x_i - x_{i-1}) - \beta(x_i - x_{i+1}) + e_v^i,$$

$$\begin{aligned} \dot{v} &= -\beta_r e_v + \frac{1}{m} e_F, \\ \dot{e}_F &= -\beta_F e_F, \end{aligned}$$

where $i = 2, \dots, N-1$.

According to Assumption 2.5 and dynamics in the above equations and (2.7), we can taking derivatives of $e_x$ in (2.9) and get the dynamics of it as, $\dot{e_x} = Ae_x + e_v$, where $A = \begin{bmatrix} -2\beta & \beta & \dots & \dots & 0 \\ \beta & -2\beta & \beta & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \beta & -2\beta \end{bmatrix}$.

Then the dynamics of $e_x, e_v$ and $e_F$ can be written as

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_v \\ \dot{e}_F \end{bmatrix} = \begin{bmatrix} A & I_{N-2} & 0_{N-2} \\ 0_{N-2} & -\beta_r I_{N-2} & \frac{1}{m} I_{N-2} e_F \\ 0_{N-2} & 0_{N-2} & -\beta_F I_{N-2} \end{bmatrix} \begin{bmatrix} e_x \\ e_v \\ e_F \end{bmatrix}. \tag{2.18}$$

We can see that $A$ is a tridiagonal matrix and it is also Toeplitz, the eigenvalues of $A$ are

$$\lambda_i = -2\beta + 2\beta cos(\frac{k\pi}{N-1}), for\ k = 1, 2, \dots, N-2,$$

which are all negative. So $A$ is Hurwitz. Furthermore, $-\beta_r I_{N-2}$ and $-\beta_F I_{N-2}$ are all Hurwitz, then the system (2.18) is exponentially stable at the origin. So $e_x(t)$ will converge to 0.

By denoting $x_e = \left[e_x^\top, e_v^\top, e_F^\top\right]^\top$, we can also conclude that, there exists a $\lambda > 0$, such that $\|x_e(t)\| \leq \|x_e(0)\| e^{-\lambda t}$, $\forall t \geq 0$. By letting

$$K_0 = \|x_e(0)\|, \tag{2.19}$$

we have, $\|e_x(t)\| \leq \|x_e(t)\| \leq K_0 e^{-\lambda t}$. □

### 2.5.2 Proof of Lemma 2.2

*Proof.* The proof will be slightly different between $N$ is even number and odd number. We start the proof from analysing the system with $N$ is even number, that is $N = 2n$, where $n = 2, 3, \ldots,$. The proof will be similar for $N = 2n - 1$, where $n = 2, 3, \ldots,$, so we will just give the results directly.

In the first step of this proof, we check that the distances $\|\bar{x}_i - \bar{x}_{2n+1-i}\|$, where $i = 2, \ldots, N-1$ are less than or equal to some upper bounds which are functions of $\bar{x}_i, i \in \mathcal{V}$. Based on these inequalities, we can find a constant upper bound on $\|\bar{x}_n - \bar{x}_{n+1}\|$. In the second step, we will check that the distances $\|\bar{x}_i - \bar{x}_{i+1}\|$ where $i = 1, \ldots, N-1$ are less than or equal to some upper bounds which are functions of $\bar{x}_i, i \in \mathcal{V}$. By using these inequalities and the upper bound on $\|\bar{x}_n - \bar{x}_{n+1}\|$ in the first step, we can get constant upper bounds on $\|\bar{x}_i - \bar{x}_{i+1}\|$ where $i = 1, \ldots, N-1$, which will satisfy the requirements in (2.11).

For each pair of $\bar{x}_i$ and $\bar{x}_{2n+1-i}$, we construct a Lyapunov function $V_i = \frac{1}{2}\|\bar{x}_i - \bar{x}_{2n+1-i}\|^2$, where $i = 3, \ldots, n+1$. By this choice of $i$, we will start the analysis by considering the left-hand part agents in the system as in Fig. 2.3.

The derivatives of $V_i$ are as follows

$$\begin{aligned}
\dot{V}_n =&(\bar{x}_{n+1} - \bar{x}_{n+2})^\intercal[-3\beta(\bar{x}_n - \bar{x}_{n+1}) \\
&+ \beta(\bar{x}_{n-1} - \bar{x}_{n+2})] \\
\leq& -3\beta\|\bar{x}_n - \bar{x}_{n+1}\|(\|\bar{x}_n - \bar{x}_{n+1}\| \\
&- \frac{1}{3}\|\bar{x}_{n-1} - \bar{x}_{n+2}\|), \\
\dot{V}_i =&(\bar{x}_i - \bar{x}_{2n+1-i})^\intercal[-2\beta(\bar{x}_i - \bar{x}_{2n+1-i}) \\
&+ \beta(\bar{x}_{i-1} - \bar{x}_{2n+2-i}) + \beta(\bar{x}_{i+1} - \bar{x}_{2n-i})] \\
\leq& -2\beta\|\bar{x}_i - \bar{x}_{2n+1-i}\|(\|\bar{x}_i - \bar{x}_{2n+1-i}\| \\
&- \frac{1}{2}\|\bar{x}_{i-1} - \bar{x}_{2n+2-i}\| - \frac{1}{2}\|\bar{x}_{i+1} - \bar{x}_{2n-i}\|),
\end{aligned}$$

where $i = n-1, \ldots, 2$. A sufficient condition under which the derivatives of $V_i$ are less than 0 are as follows

$$\begin{aligned}
\|\bar{x}_n - \bar{x}_{n+1}\| >& \frac{1}{3}\|\bar{x}_{n-1} - \bar{x}_{n+2}\|, \\
\|\bar{x}_i - \bar{x}_{2n+1-i}\| >& \frac{1}{2}(\|\bar{x}_{i-1} - \bar{x}_{2n+2-i}\| \\
&+ \|\bar{x}_{i+1} - \bar{x}_{2n-i}\|),
\end{aligned}$$

where $i = n-1, \ldots, 2$. From this condition, we can see the solution of the single-integrator system will converge to states that satisfy the following conditions,

$$\begin{aligned}
\|\bar{x}_n - \bar{x}_{n+1}\| \leq& \frac{1}{3}\|\bar{x}_{n-1} - \bar{x}_{n+2}\|, \\
\|\bar{x}_i - \bar{x}_{2n+1-i}\| \leq& \frac{1}{2}(\|\bar{x}_{i-1} - \bar{x}_{2n+2-i}\| \\
&+ \|\bar{x}_{i+1} - \bar{x}_{2n-i}\|),
\end{aligned}$$

Additionally, from the requirements on the initial states in (2.4) in Assumption 2.1, we can conclude that $\|\bar{x}_i(t) - \bar{x}_{2n+1-i}(t)\|$ will satisfy the former inequalities $\forall t \geq 0$.

Furthermore, by some calculation, we can rewrite the former inequalities as $\|\bar{x}_i - \bar{x}_{2n+1-i}\| \leq \frac{2n-2i+1}{2n-2i+3}\|\bar{x}_{i-1} - \bar{x}_{2n+2-i}\|$, According to Assumption 2.3, we have $\|\bar{x}_1(t) - \bar{x}_N(t)\| \leq (N-1)(R-\bar{\Delta})$, $\forall t \geq 0$. Substitute this into the former inequalities, we have, $\|\bar{x}_i - \bar{x}_{2n+1-i}\| \leq (2n-2i+1)(R-\bar{\Delta})$. Especially for $\|\bar{x}_n - \bar{x}_{n+1}\|$, we have,

$$\|\bar{x}_n - \bar{x}_{n+1}\| \leq (R-\bar{\Delta}). \tag{2.20}$$

In the second step of the proof, we construct another group of Lyapunov functions for each pair of $\bar{x}_i, \bar{x}_{i+1}$, where $i = 1, \ldots, n-1$, as $V_{0i} = \frac{1}{2}\|\bar{x}_i - \bar{x}_{i+1}\|^2$, this is still for the left-hand part of the agents in the system as in Fig. 2.3. According to Assumptions 2.5 and 2.2, the derivatives of $V_{0i}$ are as follows

$$\begin{aligned}
\dot{V}_{01} =& (\bar{x}_1 - \bar{x}_2)^\intercal[-\beta(\bar{x}_2 - \bar{x}_1) + \beta(\bar{x}_2 - \bar{x}_3) + v_1] \\
\leq& -\beta\|\bar{x}_1 - \bar{x}_2\|\left(\|\bar{x}_1 - \bar{x}_2\| - \|\bar{x}_2 - \bar{x}_3\| - v_{1m}\right), \\
\dot{V}_{0i} =& (\bar{x}_i - \bar{x}_{i+1})^\intercal[-2\beta(\bar{x}_i - \bar{x}_{i+1}) + \beta(\bar{x}_{i-1} - \bar{x}_i)]
\end{aligned}$$

$$
\begin{aligned}
&+\beta(\bar{x}_{i+1}-\bar{x}_{i+2})] \\
\leq &-2\beta\|\bar{x}_i-\bar{x}_{i+1}\|(\|\bar{x}_i-\bar{x}_{i+1}\|-\|\bar{x}_{i-1}-\bar{x}_i\| \\
&-\|\bar{x}_{i+1}-\bar{x}_{i+2}\|),
\end{aligned}
$$

where $i=2,\dots,n-1$.

A sufficient condition under which these derivatives of $V_{0i}$ are less than 0 is

$$
\begin{aligned}
\|\bar{x}_1-\bar{x}_2\| >&\|\bar{x}_2-\bar{x}_3\|+\frac{1}{\beta}v_{1m}, \\
\|\bar{x}_i-\bar{x}_{i+1}\| >&\frac{1}{2}(\|\bar{x}_{i-1}-\bar{x}_i\|+\|\bar{x}_{i+1}-\bar{x}_{i+2}\|),
\end{aligned}
$$

where $v_{1m}$ is the maximal value of $\|v_1(t)\|$ as in Assumption 2.2.

Similar as above analysis for $V_i$, the states $\bar{x}_i$, $i\in\mathcal{V}$ will converge to states such that the following inequalities will always be satisfied.

$$
\begin{aligned}
\|\bar{x}_1-\bar{x}_2\| \leq&\|\bar{x}_2-\bar{x}_3\|+\frac{1}{\beta}v_{1m}, \\
\|\bar{x}_i-\bar{x}_{i+1}\| \leq&\frac{1}{2}(\|\bar{x}_{i-1}-\bar{x}_i\|+\|\bar{x}_{i+1}-\bar{x}_{i+2}\|),
\end{aligned}
$$

where $i=2,\dots,n-1$.

Additionally, according to (2.4) in Assumption 2.4, we conclude that these inequalities are always satisfied $\forall t\geq 0$.

Furthermore, by some calculation, we can rewrite the above inequalities as

$$
\begin{aligned}
\|\bar{x}_1-\bar{x}_2\| \leq&\|\bar{x}_2-\bar{x}_3\|+\frac{1}{\beta}v_{1m}, \\
\|\bar{x}_i-\bar{x}_{i+1}\| \leq&\|\bar{x}_{i+1}-\bar{x}_{i+2}\|+\frac{1}{\beta}v_{1m}.
\end{aligned}
$$

From former analysis, we have $\|\bar{x}_n-\bar{x}_{n+1}\|\leq(R-\bar{\Delta})$ as in (2.20). By substituting this inequality in the above inequalities, we finally have

$$
\|\bar{x}_i-\bar{x}_{i+1}\|\leq(R-\bar{\Delta})+\frac{n-i}{\beta}v_{1m},
$$

where $1 \leq i \leq n-1$.

Similar analysis can be applied for the other half side containing agent 1 as in Fig. 2.3, so we also have

$$\|\bar{x}_i - \bar{x}_{i+1}\| \leq (R - \bar{\Delta}) + \frac{i-n}{\beta} v_{2m},$$

where $n+1 \leq i \leq 2n-1$, $v_{1m}$ is the maximal value of $\|v_1(t)\|$ as in Assumption 2.2.

Analysis on system with $2n$ agents can also be applied similarly on the system consisting of $2n-1$ agents, where $n = 2, 3, \ldots$. By using the similar analysing method and under the conditions of initial states in (2.5) in Assumption 2.4, we finally have the following results will always hold $\forall t \geq 0$.

$$\|\bar{x}_i - \bar{x}_{i+1}\| \leq (R - \bar{\Delta}) + \frac{2n-2i-1}{2\beta} v_{1m},$$

where $1 \leq i \leq n-1$, $\|\bar{x}_i - \bar{x}_{i+1}\| \leq (R - \bar{\Delta}) + \dfrac{2i-2n+1}{2\beta} v_{1m}$, where $n \leq i \leq 2n-1$. We denote $d_{imax}, d_{Nmax}$ as,

$$d_{imax} = \max_i \{\sup_t \|\bar{x}_i(t) - \bar{x}_{i+1}(t)\|\},$$

where $i = 1, \ldots, N-1$, and we denote $d_{max}$ as, $d_{max} = \max\{d_{imax}\}$. From Assumption 2.4, we have, $v_{1m} \leq v_{cm}, v_{2m} \leq v_{cm}$. Then, from the results of upper bound on the distance between neighbouring two agents on systems with odd and even number of agents, we have, $d_{max} \leq (R - \bar{\Delta}) + \dfrac{N}{2\beta} v_{cm}$. By denoting, $\bar{\Delta}_1 = \bar{\Delta} - \dfrac{N}{2\beta} v_{cm}$, we have $d_{max} \leq (R - \bar{\Delta}_1)$. From the definition of $d_{max}$, we conclude that the states $\bar{x}_i, i \in \mathcal{V}$ satisfy the conditions in (2.11) $\forall t \geq 0$. $\square$

## 2.6 Conclusions

In this chapter, we consider a practical multi-agent system and design controllers for routers to guarantee persistent connectivity of the two clients in the system. The main concern of this connectivity design is to guarantee that the clients can move freely to perform their pre-assigned tasks, which is a typical situation in practical environments. The controllers designed for routers are proved to ensure persistent connectivity of clients assisted by an auxiliary single-integrator system and the simulation results verified the efficiency of the designed controllers. However, in some cases, not all the routers are needed to provide connectivity if a subset of them is sufficiently to handle it , this is also an interesting aspect to consider.

# Chapter 3

# Persistent Connectivity with Multiple Clients

*Persistent connectivity of clients is addressed in a general case of multi-agent systems with an arbitrary number of clients. The control policy in this chapter consists of periodically computing desired positions and then steering the routers to those desired positions. First, desired positions of routers are determined by an optimization problem which minimizes the length of the longest edge of a tree corresponding to the communication relationship between agents. Then, the routers are steered to those desired positions by motion control. When the optimization problem is infeasible for communication connectivity, new routers are added to change the structure of the communication graph (tree). A quadrotor model of agents is used in the simulation with the control policies for routers. A comparison of two optimization algorithms for solving the optimization problem is also presented.*

## 3.1 Introduction and Background

IN this chapter, we consider communication connectivity in the general case with an arbitrary number of clients. More clients make the communication structure between agents more complex and the control policy design more complicated.

In the communication connectivity literature, such as the papers reviewed in Chapter 1, there is a common assumption (implicit or explicit) that the system can be maintained connected by maintaining the initial communication topology. This assumption is reasonable if the main goal of all agents is to maintain the communication connectivity. However, in practical situations, a subset of agents may have a more important task to perform and by if they cooperated in maintaining the communication connectivity this would severely limit their ability to perform their main task. For example, they may need to reach some locations for surveillance, measure something or help search for something or somebody. If these agents are restricted by the communication connectivity control, they may not be able to complete their main tasks. When considering these situations, it may not be feasible to keep the original communication topology of the graph for all time and the communication graph may need to be occasionally modified. For instance, this can be done by re-positioning the routers, adding or deleting edges of the graph or even adding more routers in the systems. Another aspect in the literature that may be improved is the dynamic models of the agents, which are usually robots. The models are always considered as single integrator or double integrator models. However, the models of actual robots are always more complex, for instance, models of UAVs are given by [9].

Instead of assuming connectivity can be maintained for all time by using the same communication graph structure, our control policy changes the graph structure when needed to guarantee connectivity. The control policy keeps switching between optimization and then control. If the optimization is infeasible, the policy resorts to an update of the communication graph structure (adding more routers, for instance). When the optimization is again feasible, the policy starts again switching between optimization and control. During the optimization and control, first, an optimization algorithm calculates the desired positions of the routers by minimizing the length of the longest edge of a tree selected from the communication graph. Then, the routers are steered to their desired positions by motion control. Moreover, our control policy is also suitable for general dynamic models of agents, which makes it widely applicable. With these properties, the approach in this chapter is more suitable than those in the literature for practical situations.

The rest of the chapter is organized as follows. We firstly give some preliminaries in Section 3.2. In Section 3.3, we give a description of the multi-agent system and the problem considered. A procedure of solving the connectivity problem is presented in Section 3.4. This section includes two methods of solving an optimization problem to get desired positions of routers, update of graphs, and requirements on motion controller design for routers. The result that persistent connectivity of clients is guaranteed by our control policy is also given to complete this section. Simulation results are presented in Section 3.5, and conclusions are presented in the last section.

## 3.2 Preliminaries

Some basic definitions regarding the sub-gradients of convex functions are presented next. For more information the readers may refer to [10].

**Definition 3.1.** *Vector $g$ is a sub-gradient of a convex function $f : \mathcal{D} \to \mathbb{R}$ at $x \in \mathcal{D}$ if*

$$f(y) \geq f(x) + g^T(y - x), \quad \forall y \in \mathcal{D}.$$

**Definition 3.2.** *The sub-differential $\partial f(x)$ of $f$ at $x$ is the set of all sub-gradients:*

$$\partial f(x) = \{g | f(y) \geq f(x) + g^T(y - x), \quad \forall y \in \mathcal{D}\}.$$

Let $f(x) = \max\{f_1(x), \ldots, f_m(x)\}$, we know that $f$ is convex if all $f_i$ are convex. Moreover, assume each $f_i$ is differentiable. At a point $\bar{x}$ where $f(\bar{x}) = f_j(\bar{x})$ for some $j \in \{1, \ldots, m\}$, $\nabla f_j(\bar{x}) \in \partial f(\bar{x})$. This is because

$$\begin{aligned} f(\bar{x}) + \nabla f_j^T(\bar{x})(y - \bar{x}) &= f_j(\bar{x}) + \nabla f_j^T(\bar{x})(y - \bar{x}) \\ &\leq f_j(y) \leq \max\{f_1(y), \ldots, f_m(y)\} = f(y) \end{aligned}$$

If there exist more than one $f_i(x)$ with maximal value, we use $M(x)$ to denote the set of indices of functions $f_i(x)$ with the maximal values. The set $M(x)$ is defined as $M(x) = \Big\{ i \Big| i \in \arg\max\{f_1(x), \ldots, f_m(x)\} \Big\}$. Then a set of sub-gradients of $f(x)$ at $x$ is

$$S_f(x) = \Big\{ s_f \Big| s_f = \sum_{i \in M(x)} \alpha_i \nabla f_i(x),\ \alpha_i \geq 0 \textit{ and } \sum_{i \in M(x)} \alpha_i = 1 \Big\}. \tag{3.1}$$

This is because $\forall\, s_f \in S_f(x)$,

$$\begin{aligned} f(x) + s_f^T(y-x) &= \sum_{i\in M(x)} \alpha_i f_i(x) + \sum_{i\in M(x)} \alpha_i \nabla f_i(x)(y-x) \\ &\le \sum_{i\in M(x)} \alpha_i f_i(y) \le \sum_{i\in M(x)} \alpha_i \max\left\{f_1(y),\dots,f_m(y)\right\} = f(y). \end{aligned}$$

It is obvious that $S_f(x) \subset \partial f(x)$, so we can choose any element of $S_f(x)$ as the subgradient of $f(x)$ at $x$.

We use the notation $P_C(x) : \mathbb{R}^n \to C$ to denote a projection of $x$ on set $C \subset \mathbb{R}^n$, such that $\|x - P_C(x)\| = \inf\left\{\|x-\tilde{x}\| | \tilde{x} \in C\right\}$. Notation $|\cdot|$ denotes the number of elements in a set, $v^\mathsf{T}$ denotes transpose of a vector $v$ and $\lceil\cdot\rceil$ denotes the ceiling function.

## 3.3 Problem Setup

A multi-agent system with an arbitrary number of clients is considered in this chapter. The structure of such a general multi-agent system is shown in Figure 1.3. Different from the simple communication structure in systems with two clients, we note that the clients might also be used to aid communication if it does not conflict their own control objectives. The main problem in this chapter is still designing suitable motion control policies for the routers to guarantee that the clients are persistently connected. In the sequel, we will formally state the problem description along with necessary assumptions.

For simplicity, in this chapter, let $x_i \in \mathbb{R}^2$, the position of agent $i \in \mathcal{V}$, be governed by

$$\dot{x}_i = u_i,\ \forall i \in \mathcal{V}, \tag{3.2}$$

where $u_i \in \mathcal{U}_i \subset \mathbb{R}^2$ is the control input of agent $i$. Here, $\mathcal{U}_i$ is defined as

$$\mathcal{U}_i = \begin{cases} \left\{ \left[u_i^1\ u_i^2\right]^\mathsf{T} \middle| \left\| \left[u_i^1, u_i^2\right]^\mathsf{T} \right\| \le v_r \right\}, & \text{if } i \in \mathcal{R} \\ \left\{ \left[u_i^1\ u_i^2\right]^\mathsf{T} \middle| \left\| \left[u_i^1, u_i^2\right]^\mathsf{T} \right\| \le v_c \right\}, & \text{if } i \in \mathcal{C}. \end{cases} \tag{3.3}$$

In the problem considered, the control policies of clients $u_i,\ \forall i \in \mathcal{C}$, need to be designed to achieve their objectives. Later in this chapter we will relax this assumption for the dynamic model of agents because our control policy is not dependent on specific dynamic models. The policy only requires that the speeds of clients are bounded and the motion controllers of the routers satisfy some requirements presented in Subsection 3.4.4.

We have the following standing assumption on the speed of each agent $i \in \mathcal{V}$.

**Assumption 3.1.** *The upper bounds $v_r$ and $v_c$ of speed of agents satisfy*

$$v_r > \bar{v}_r > \max\left\{\frac{(\Delta + t_x)v_c + e_r}{t_m}, \frac{3e_r - t_d v_c}{t_m}\right\},$$

*where $\Delta$, $t_x$, $e_r$, $t_m$ are some positive constants, which will be defined.*

This assumption ensures that routers have the potential to move sufficiently faster than clients to maintain any existing connections with them.

The condition that characterises the communication among agents is the same as in Assumption 1.1. The following assumption specifies the bounded area $B(t)$ in which the clients operate.

**Assumption 3.2.** *The clients move within a time-varying bounded area $B(t)$ all the time, that is, $x_i(t) \in B(t)$, $\forall i \in \mathcal{C}$, $\forall t \geq t_0$, where $t_0$ is the initial time instant, $B(t)$ is defined as $B(t) = \{x \mid \|x - x_c(t)\| \leq \frac{D}{2}\}$ with $D$ being a positive constant and $x_c(t) = \frac{\sum_{i \in \mathcal{C}} x_i(t)}{|\mathcal{C}|}$.*

The objective is to design distributed closed loop control policies $u_i(t)$, $\forall i \in \mathcal{R}$ to guarantee that clients are persistently connected. Assumption 1.3 still holds here to characterise the information exchange between neighbouring two agents.

In this chapter, we need a stronger connectivity assumption at the initial time instant than Assumption 1.4 as follows. First, we define a slightly different graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\{i, j\} \in \mathcal{E}(t)$ if

$$\|x_i(t) - x_j(t)\| \leq R - \delta_G, \tag{3.4}$$

where $\delta_G$ is a positive constant, which we will revisit later in the chapter. Here, $\delta_G$ always satisfies

$$\delta_G < R. \tag{3.5}$$

The graph $G(t)$ defined here is for the purpose of designing control policies, while the previous graph $G_o(t)$ determines the communication relationship among agents. Regarding the initial positions of agents, we have the following assumption.

**Assumption 3.3.** *At the initial time instant $t = t_0$, $G(t_0)$ is connected.*

In an area $B(t)$, the routers cannot maintain persistent connectivity for clients if the number of routers is not large enough. Therefore, we present an assumption on the number of routers in the system.

**Assumption 3.4.** *The number of routers $N_r$ satisfies*

$$N_r \geq 2N_c \left\lceil \frac{\frac{D}{2} + R - 2e_r}{R - \delta - 2e_r} + 1 \right\rceil + 1, \tag{3.6}$$

*where $N_c$ is the number of clients, $R$ is the maximum communication range, $D$ is the constant used to define $B(t)$, and $\delta$[1] is a positive constant.*

This assumption on the number of routers depends on the design of control policies proposed in this chapter. It guarantees that there are enough routers to ensure persistent connectivity of clients under the proposed methodologies, which is shown in Section 3.4. Next, we introduce an assumption on the relationship between $R$ and $\delta$.

**Assumption 3.5.** *In the system we consider, $R$ and $\delta$ satisfy*

$$R \geq 4\delta. \tag{3.7}$$

The main focus of this chapter is to provide a solution to the following problem,

**Problem 3.1.** *Under Assumptions 3.1–3.5, 1.1 and 1.3, design control policies $u_i(t) \in \mathcal{U}$, $\forall i \in \mathcal{R}$, so that all clients are persistently connected.*

The details of the solution of Problem 3.1 are included in Section 3.4.

[1]This $\delta$ is a design constant whose value is stated in Subsection 3.4.5.

## 3.4 Problem Solution

The main contribution of this chapter is a solution to Problem 3.1, which is described in Algorithm 1. The algorithm always works with a subset of "active routers" that maintain communication connectivity and "inactive routers" that are not used for communication connectivity but are made available if the need arises (e.g. clients move too far from each other and the current number of active routers is no longer sufficient to maintain the communication connectivity). The algorithm is implemented in a distributed fashion and it switches between different modes of operation as illustrated in Figure 3.1. Dur-

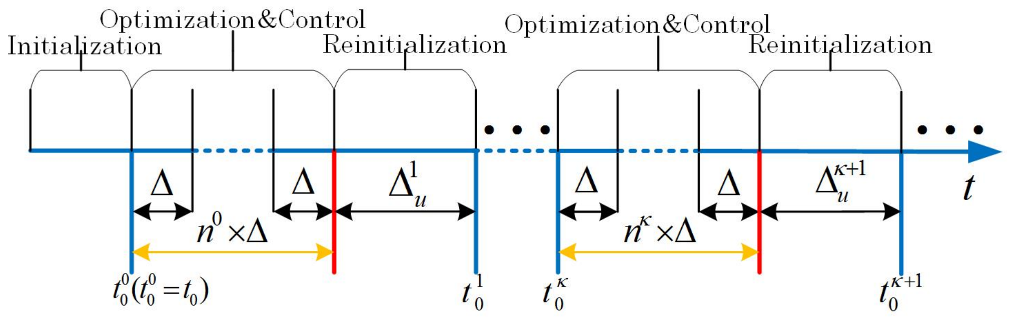


Figure 3.1: Modes of operations in Algorithm 1

ing the initialization stage, the algorithm uses the initial positions of clients to select active routers and their positions that can establish communication connectivity for clients. Once the initialization is completed, the algorithm switches to optimization and control stage. During the optimization stage, positions of clients are measured at equidistant sampling times and then used to determine new desired positions of the active routers via an optimization algorithm. Meanwhile, the desired positions of inactive routers are determined in a different fashion to make sure they are always connected to some active routers. In the control stage, the desired positions of both active and inactive routers are used in their corresponding control algorithms to steer all routers to their desired positions. Algorithm 1 alternates between optimization and control stages as long as the corresponding optimization problem is feasible[2]. If at some stage, the clients have moved to positions where the optimization problem is infeasible, the algorithm is reinitialized. Reinitialization determines a new set of active routers and their positions so that commu-

[2]The definition of feasibility is in Subsection 3.4.1.

nication connectivity of clients can be maintained. Once the reinitialization produces a new set of active routers with their positions, the algorithm switches to the optimization and control stage which was described above. The algorithm is reinitialized every time the infeasibility occurs, whereas optimization and control stage is continued for as long as the optimization problem is feasible. We describe below all stages of Algorithm 1 in more detail, see Figure 3.1.

In the initialization stage (see lines $1-8$ in Algorithm 1) as in Figure 3.1, the active routers are selected using the graph $G$ defined in Section 3.3. Firstly (line 1, Algorithm 1) a router is appointed to $x_c(t)$ as the base. The index of the base is denoted by $b$. Then we use $\mathcal{R}_b$ to denote the set of routers except the base $b$, that is $\mathcal{R}_b := \mathcal{R} \setminus \{b\}$. This appointment of the base will make sure the number of routers in Assumption 3.4 is enough to provide communication connectivity for clients, which is explained in Subsection 3.4.3. The variable $\kappa$ is used as an index that enumerates the initialization and reinitialization stages. The superscript $\kappa$ in variables, such as $G^\kappa$, denotes the variables determined in the initialization or reinitialization stage. The initial value $\kappa = 0$ denotes the initialization stage (line 2, Algorithm 1), and $\kappa$ will be increased by 1 every time a reinitialization stage is encountered in the iterations. Then $G(t)$ at current time is assigned to $G^\kappa$ (line 3, Algorithm 1), where $\kappa = 0$. Once $G^\kappa$ is generated, a sub-graph in form of a tree[3] (denoted as $T^\kappa = (\mathcal{V}_{T^\kappa}, \mathcal{E}_{T^\kappa})$) is selected from $G^\kappa$ (line 4, Algorithm 1). This tree is generated so that its set of vertices includes the set of clients and the number of routers (including $b$) on the tree is less than or equal to $N_c \left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} + 1 \right\rceil + 1$[4], that is, $\mathcal{C} \subset \mathcal{V}_{T^\kappa}$ and $\left|\mathcal{V}_T \cap \mathcal{R}\right| \leq N_c \left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} + 1 \right\rceil + 1$. Another requirement of the tree is that there is no edge between two clients[5]. Rules for choosing the tree depend on a user defined criterion. For example, the tree can be chosen so that it contains a minimum number of routers. Using the selected tree $T^\kappa$, we define the set of active routers as $S_A^\kappa = \mathcal{R}_b \cap \mathcal{V}_{T^\kappa}$, and the set of inactive routers as $S_I^\kappa = \mathcal{R}_b \setminus S_A^\kappa$. In order to connect the inactive routers to the tree $T^\kappa$, we

[3]A tree is chosen as the sub-graph because it is simple to check the connectivity of a tree. In fact, the solution in this chapter can be generalized to cover other types of sub-graphs.

[4]Such a tree with such a vertex number requirement can be constructed at the initialization stage by using a method similar to the one presented in Algorithm 8.

[5]This requirement can guarantee the clients will not lose connectivity with their neighbouring routers, which is explained in subsection 3.4.5.

**Algorithm 1** Outline of solving Problem 3.1

```
1: Move a router to $x_c(t)$ as the base    ▷ $t$ is current time instant
2: $\kappa \leftarrow 0$
3: $G^\kappa \leftarrow G(t)$
4: Choose $T^\kappa$ from $G^\kappa$ and determine $S_A^\kappa, S_I^\kappa$ which are sets of acitve and inactive routers
5: Construct extended tree $T_E^\kappa$ as in Algorithm 4
6: $t_0 \leftarrow t$
7: $t_0^\kappa \leftarrow t_0$
8: $n \leftarrow 0$
9: while True do
10:     Clients measure their own positions $x_i(t_0^\kappa + n\Delta), \forall i \in \mathcal{C}$.
11:     The base $b$ calculates and keeps $x_c(t_0^\kappa + n\Delta)$
12:     Calculate the desired positions of routers in $S_A^\kappa$ by using Algorithm 2 or 3
13:     Determine the feasibility of the desired positions of active routers
14:     if The desired positions are infeasible then
15:         $n_u \leftarrow 0$
16:         $T_{n_u}^\kappa = T^\kappa$
17:         while The desired positions are infeasible do
18:             Update $T_{n_u}^\kappa$ by using Algorithm 6 or 8 to get $T_{n_u+1}^\kappa$ and $T^\kappa$ is updated by
    $T^\kappa = T_{n_u+1}^\kappa$    ▷ $T_E^\kappa$ is also updated, there also exist $T_{E_{n_u}}^\kappa$,$S_{I_{n_u}}^\kappa$ and $S_{A_{n_u}}^\kappa$, please refer to 6
    or 8 for details.
19:             Clients measure their own positions $x_i(t_0^\kappa + n\Delta + n_u(t_u + t_x) + t_u), \forall i \in \mathcal{C}$.
20:             The base $b$ calculates and keeps $x_c(t_0^\kappa + n\Delta + n_u(t_u + t_x) + t_u)$
21:             Calculate new desired positions with updated $T_{n_u+1}^\kappa$ by using Algorithm 2
    or 3
22:             Determine the feasibility of the desired positions of active routers
23:             $n_u \leftarrow n_u + 1$
24:         end while
25:         Calculate desired positions of the routers in $S_{I_{n_u}}^\kappa$ by Algorithm 5
26:         Set the desired position of the base as $x_c(t_0^\kappa + n_u(t_u + t_x))$
27:         Steer routers to desired positions as in Subsection 3.4.4 with their desired po-
    sitions
28:         $\kappa \leftarrow \kappa + 1$
29:         $G^\kappa = G(t)$
30:         Choose $T^\kappa$ from $G^\kappa$ and determine $S_A^\kappa, S_I^\kappa$ which are sets of active and inactive
    routers
31:         Construct extended tree $T_E^\kappa$ as in Algorithm 4
32:         $t_0^\kappa \leftarrow t$
33:         $n \leftarrow 0$
34:     else
35:         Calculate desired positions of the routers in $S_I^\kappa$ by Algorithm 5
36:         Set the desired position of the base as $x_c(t_0^\kappa + n\Delta)$
37:         Steer routers to desired positions as in Subsection 3.4.4 with their desired po-
    sitions
38:         $n \leftarrow n + 1$
39:     end if
40: end while
```

construct an extended tree (denoted as $T_E^\kappa = (\mathcal{V}_{T_E^\kappa}, \mathcal{E}_{T_E^\kappa})$) by moving inactive routers close to some nearby active routers or clients (line 5, Algorithm 1). The tree $T_E^\kappa$ is constructed by using Algorithm 4. The generated tree $T_E^\kappa$ contains all agents in the system and all edges of $T^\kappa$, which are $\mathcal{V}_{T_E^\kappa} = \mathcal{V}$, $\mathcal{E}_{T^\kappa} \subset \mathcal{E}_{T_E^\kappa}$. This tree $T_E^\kappa$ will make inactive routers available if they are needed. At the end of the initialization stage, the current time is assigned to $t_0$ and $t_0$ is assigned to $t_0^\kappa$ (lines 6 and 7, Algorithm 1). Here $t_0$ denotes the start time of the system when clients start to move, and $t_0^\kappa$ denotes the start time of the optimization and control stage right after the initialization or reinitialization stage indexed by $\kappa$. Here $\kappa = 0$, that is the initialization stage. Meanwhile, $n$ is used and initialized as 0 to enumerate the number of periods in the next optimization and control stage (line 8, Algorithm 1). After $t_0$ as shown in Figure 3.1, the algorithm switches to the optimization and control stage.

*Remark:* We note that while the tree graph structure is simplest to implement, it may suffer from a lack of robustness as malfunction of any of the routers would break the connectivity between some clients. Considering other types of trees for the sake of ensuring better robustness is an interesting question that is left for further research.

The optimization and control stage starts at time instant $t_0^\kappa$, with $\kappa = 0$ if such a stage is right after the initialization stage. During the optimization and control stages in Figure 3.1, Algorithm 1 runs periodically with a time period $\Delta$. The variable $n^\kappa$ is used to denote the total number of periods in the optimization and control stage after the initialization or reinitialization indexed by $\kappa$. Note that $n^\kappa$ can be 0 in some situations. This $\Delta$ is further divided into several time intervals, as shown in Figure 3.2, according to different purposes of the steps in optimization and control stages. Time interval $t_x$ is used to

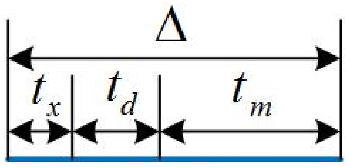


Figure 3.2: Divisions of $\Delta$

calculate desired positions of the active routers (lines $10 - 13$, Algorithm 1). Firstly, the positions of clients at the start time $t_0^\kappa + n\Delta$ of a period indexed by $n$ are measured and stored for further actions (line 10, Algorithm 1). With these measured client positions, the base calculates the centroid $x_c(t_0^\kappa + n\Delta)$ via communication links aided by other agents (line 11, Algorithm 1). In the next step (line 12, Algorithm 1), the desired positions of active routers are calculated by an optimization algorithm (Algorithm 2 or 3) with the measured positions of clients and the centroid $x_c(t_0^\kappa + n\Delta)$. These two algorithms are distributed optimization algorithms, where Algorithm 2 uses a primal method and Algorithm 3 uses a dual method to minimize the length of the edges in the worst case. Here, the worst case means the edges with the longest length. Then, the feasibility of the calculated desired positions will be checked (line 13, Algorithm 1). If the desired positions of active routers are feasible, the optimization and control stage continues. Otherwise, the algorithm would switch to reinitialization stage, which we will describe in the next paragraph. In the feasible case, the next time interval $t_d$ is assigned to obtain the desired positions of the inactive routers (lines $35 - 36$, Algorithm 1). The desired positions of the inactive routers are determined by Algorithm 5 (line 35, Algorithm 1). These positions are selected to make sure the inactive routers will not lose connection to their neighbours in $T_E^\kappa$. Meanwhile, the desired position of the base is set as $x_c(t_0^\kappa + n\Delta)$ to make sure the number of routers in the system is enough (line 36, Algorithm 1). During the last time interval $t_m$ (lines $37 - 38$, Algorithm 1), all the routers will be steered to their desired positions with some user designed controllers. The controllers are designed by using the desired positions of routers and should satisfy some requirements as described in Subsection 3.4.4 (line 37, Algorithm 1). At the end of current $\Delta$, $n$ is increased by 1 to index the next time period $\Delta$ if the optimization and control stage can be continued (line 38, Algorithm 1).

As stated above, Algorithm 1 steps into a reinitialization stage, as shown in Figure 3.1, if the desired positions of active routers are infeasible (lines $10 - 33$, Algorithm 1). The time interval of a reinitialization stage indexed by $\kappa$ is $\Delta_u^\kappa$. Time interval of this stage is also divided into several time intervals as shown in Figure 3.3. During the first time interval $t_x$ (lines $10 - 13$, Algorithm 1), the desired positions of the active routers are calculated. The actions taken in $t_x$ are the same as the actions in the time interval $t_x$ of the optimization and control stages, thus we will not provide detailed explanations.

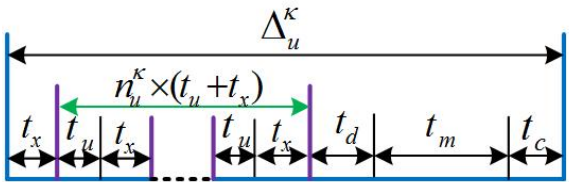


Figure 3.3: Divisions of $\Delta_u^\kappa$

Similarly, all the time intervals with the same names as in the optimization and control stages will have the same actions, the only difference is that they are with different trees. After time interval $t_x$, $n_u$ is initialized with 0 to enumerate the times of updates of the tree in the time interval $n_u^\kappa(t_u + t_x)$ as shown in Figure 3.3 (line 15, Algorithm 1). The index $n_u$ will be increased by 1 once the current tree is updated in current initialization stage. In the next step, $T_{n_u}^\kappa$ is defined to be used for the tree update in the next time interval and assigned with the current $T^\kappa$ (line 16, Algorithm 1). The subscript $n_u$ in $T_{n_u}^\kappa$ is used to indicate the tree will be used in the tree update process indexed by $n_u$. In the time interval $t_u$ (line 18, Algorithm 1), the current tree $T_{n_u}^\kappa$ will be updated with the inactive routers by using Algorithm 6 or Algorithm 8. Once the tree is updated, another $t_x$ time interval will be assigned to calculated the desired positions of active routers with the new tree (lines $19-22$, Algorithm 1). The update procedure composed of $t_x$ and $t_u$ will repeat until the desired positions of the active routers with the updated tree are feasible. As shown in Figure 3.3, we use $n_u^\kappa$ to denote the total number of updates of the tree $T^\kappa$. After the tree update processes, the desired positions of the inactive routers are determined in $t_d$ (lines $25-26$, Algorithm 1), and all the routers are steered to their desired positions in $t_m$ (line 27, Algorithm 1), as described before. The last part of the reinitialization stage is the same as the initialization stage (lines $28-33$, Algorithm 1), and a time interval $t_c$ is assigned to it. Firstly, $\kappa$ will be increased by 1 to index the current reinitialization stage (line 28, Algorithm 1). Then $G^\kappa$ is assigned with the current graph $G(t)$ (line 29, Algorithm 1). The procedures of choosing $T^\kappa$ and constructing $T_E^\kappa$ are the same as in the initialization stage (lines $30-31$, Algorithm 1). In the end, $t_0^\kappa$ is assigned with current time $t$ (line 32, Algorithm 1), and $n$ is reassigned with 0 to be used in the next stage (line 33, Algorithm 1). In the future time, Algorithm 1 will alternate between the optimization and control stage and the reinitialization stage depending on the feasibility of the desired positions of the active routers in each iteration (lines $10-39$, Algorithm 1).

*Remark:* Here the time intervals $t_u$ and $t_c$ are constants. As the time cost of the update of the selected tree depends on specific tree structure and agent positions when the update runs, $t_u$ can be chosen as the upper bound of the time cost of these updates. Similarly, $t_c$ can be chosen as the upper bound of the time cost of the corresponding procedure in Algorithm 1 (lines $28-33$, Algorithm 1). Note that our control policy designed in this chapter is also suitable for variable values of $t_u$ and $t_c$. This means that $t_u$ and $t_c$ can be time varying, in another word, they can be determined in a real-time fashion. This time-varying setting of $t_u$ and $t_c$ will not affect our analysis of the persistent connectivity of clients. We use the constants $t_u$ and $t_c$ in this chapter is to make our statement and notation clear.

During the reinitialization stages, the clients still move and may break connection with the others. Therefore, we introduce an assumption on the movement of clients during these stages to make sure they stay connected to their current neighbours.

**Assumption 3.6.** *During any reinitialization stage indexed by $\kappa$, $\kappa = 1, \ldots$ during the time interval $[t_0^\kappa - \Delta_u^\kappa, t_0^\kappa]$, the position of any client $i \in \mathcal{C}$ satisfies $\left\| x_i(t) - x_j\Big(t_0^\kappa - \Delta_u^\kappa + (n_u - 1)(t_x + t_u)\Big) \right\| \leq R - t_d v_c - e_r, \forall t \in \big(t_0^\kappa - \Delta_u^\kappa + t_x + (n_u - 1)(t_u + t_x), t_0^\kappa - \Delta_u^\kappa + t_x + n_u(t_u + t_x)\big]$, $\forall \{i,j\} \in \mathcal{N}_i^{T^{\kappa-1}_{E_{n_u-1}}}$, $\forall n_u = 1, \ldots, n_u^\kappa$ and $\|x_i(t) - x_j(t_0^\kappa - t_c)\| \leq R - \delta_G$, $\forall t \in [t_0^\kappa - t_c, t_0^\kappa]$, $\forall j \in \mathcal{N}_i^{T^\kappa}$.*

Note that if the clients do not move or move slowly during the reinitialization stages, this assumption would hold. Moreover, the clients can also cooperate during these time intervals in the sense specified by Assumption 3.6, if this is allowed by their own tasks.

The objective of Algorithm 1 is to make sure the clients are persistently connected. The intuition behind the algorithm is to maintain a tree with vertices including all the clients connected and update the tree when necessary. Obviously, with the trees $T^\kappa$ selected in Algorithm 1, ensuring that $T^\kappa, \forall \kappa = 0, 1, 2, \ldots$ remain connected is a solution for Problem 3.1. Here, the connectivity of $T^\kappa$ means all the edges of $T^\kappa$ satisfy the following conditions,

$$\|x_i(t) - x_j(t)\| \leq R, \forall \{i,j\} \in \mathcal{E}_{T^\kappa}, \ \forall t \in \big(t_0^\kappa, t_0^{\kappa+1}\big]. \tag{3.8}$$

In the remainder of this section, we will first present the algorithms (subroutines) that are needed in Algorithm 1. Then, we will prove that Algorithm 1 guarantees that the conditions in (3.8) are satisfied for each $T^\kappa$, that is, persistent connectivity of clients is guaranteed under the stated assumptions in this chapter. In Subsection 3.4.1, two alternative algorithms for optimization are given to compute desired positions of active routers. The methods to construct $T_E^\kappa$ and calculate desired positions of inactive routers are given in Subsection 3.4.2. The details of the update of $T^\kappa$ are stated in Subsection 3.4.3, and controllers design is given in Subsection 3.4.4. Finally, the conclusion and the proof of ensuring persistent connectivity of clients using Algorithm 1 is stated in Subsection 3.4.5.

### 3.4.1 Desired Positions of the Active Routers from Optimization

In this subsection, we present two optimization algorithms (Algorithms 2 and 3) which calculate the desired positions of the active routers on the selected tree. These two algorithms minimize the length of the longest edges of the selected tree. Therefore, the desired positions of the active routers calculated by the optimization algorithms can be used to achieve connectivity of the selected tree, that is, satisfy the conditions in (3.8).

As in Algorithm 1, the optimization algorithm is considered for the tree $T^\kappa$ if it is at the start of an iteration (line 12, Algorithm 1), whereas it is considered for the tree $T_{n_u}^\kappa$ if it is in reinitialization stages (line 21, Algorithm 1). For clarity, we let $T$ stand for the selected tree $T^\kappa$ or $T_{n_u}^\kappa$, and let $T_E$ stand for the extended tree $T_E^\kappa$ or $T_{E_{n_u}}^\kappa$. Similar simplification is applied for other notations, such as $\mathcal{V}_T, \mathcal{E}_T$. Furthermore, we use $x_i$ to stand for $x_i(t_0^\kappa + n\Delta)$ or $x_i\Big(t_0^\kappa + n\Delta + n_u(t_u + t_x) + t_u\Big)$ if $i \in \mathcal{C}$, use $x_c$ to stand for $x_c(t_0^\kappa + n\Delta)$ or $x_c\Big(t_0^\kappa + n\Delta + n_u(t_u + t_x) + t_u\Big)$ and use $x_j$ to stand for the decision variables corresponding to router $j$ of the optimization problem if $j \in \mathcal{R}_b$. Note that as in Algorithm 1, the base keeps the centroid position $x_c$. In Algorithm 2 and 3, we will use $x_c$ for iteration instead of the measured position of the base $b$ because the desired positions of the base will be set as $x_c$. In the following, when $x_b$ is used, we mean that $x_b = x_c$.

We use $x_T = [x_{r_1}^\intercal, \dots, x_{r_{n_T}}^\intercal]^\intercal$ to denote the collection of positions of the active routers on $T$, where $r_i$ denotes the indices of active routers, $n_T$ is the total number of the active routers on $T$, which is defined as $n_T := |\mathcal{R}_b \cap \mathcal{V}_T|$. First, we define $f_{ij}(x_T)$ as $f_{ij}(x_T) :=$

$\|x_i - x_j\|, \forall\{i,j\} \in \mathcal{E}_T$. With these $f_{ij}$, we define the objective function $f(x_T)$ as $f(x_T) := \max_{\{i,j\}\in\mathcal{E}_T} f_{ij}(x_T)$.

As stated in Algorithm 1, the active routers can only move during the time interval $t_m$ as in Figure 3.2 and 3.3. Moreover, we use $\bar{v}_r$ in Assumption 3.1 to bound the distance a router can move during $t_m$ to make sure that the routers can reach the desired positions solved by the optimization algorithm. We use $x_i^0$ to denote the position of router $i$ at the start of the optimization algorithms, which is $x_i^0 := x_i(t_0^\kappa + n\Delta)$ or $x_i^0 := x_i\Big(t_0^\kappa + n\Delta + n_u(t_u + t_x) + t_u\Big)$. Therefore, we define the reachable set of router $i$ as $C_i := \big\{x_i \big| \|x_i - x_i^0\| \leq t_m\bar{v}_r\big\}$, which constrains the position of router $i$. Instead of using conditions in equation (3.8), we use the following conditions to constrain the positions of the active routers in our optimization problem in order to guarantee persistent connectivity of clients,

$$f_{ij}(x_T) = \|x_i - x_j\| \leq R - \delta,\ \forall\{i,j\} \in \mathcal{E}_T, \tag{3.9}$$

where $\delta$ is a positive constant. As stated in Algorithm 1, the active routers stay static during the time interval $t_x$, thus lengths of some edges of $T$ might become larger in this time interval because of the movement of clients. Additionally, there are some errors between the desired positions and reached positions of routers that they are steered to since the controllers are not perfect. This $\delta$ in (3.9) is used to make sure that the tree $T$ will not lose connectivity in these situations with the desired positions of the active routers calculated by our optimization algorithms. More information on $\delta$ will be presented in subsection 3.4.5. With the objective function $f(x_T)$ and the constraints stated above, the optimization problem considered is expressed as follows,

$$\begin{aligned} &\min_{x_T} f(x_T) \\ &s.t. \quad x_i \in C_i,\ \forall i \in \mathcal{R}_b \cap \mathcal{V}_T \\ &\qquad\quad f_{ij}(x_T) \leq R - \delta,\ \forall\{i,j\} \in \mathcal{E}_T. \end{aligned} \tag{3.10}$$

If the optimization problem in (3.10) is feasible, the positions of the active routers determined by optimization algorithms are set as the desired positions of the active routers.

Otherwise we need to update the selected tree.

Instead of solving the optimization problem (3.10) directly with the correlated constraints as in (3.9), we can deal with it by an alternative way. In the alternative way, we solve an alternative optimization problem as in (3.11) without constraints in (3.9) and check the feasibility of the results of the alternative problem.

$$\begin{aligned} &\min_{x_T} f(x_T) \\ &s.t. \quad x_i \in C_i,\ \forall i \in \mathcal{R}_b \cap \mathcal{V}_T. \end{aligned} \tag{3.11}$$

In Proposition 3.1 and 3.2 we show that the alternative way can solve the optimization problem (3.10). Here we use $x_T^* = [x_{r_1}^{*\intercal}, \dots, x_{r_{n_T}}^{*\intercal}]^\intercal$ to denote an optimizer of the optimization problem in (3.11). We denote the feasible set of the optimization problem in (3.10) as $X_T^1 = \{x_T | x_i \in C_i, \forall i \in \mathcal{R}_b \cap \mathcal{V}_T\ ; f_{ij}(x_T) \le R - \delta,\ \forall \{i,j\} \in \mathcal{E}_T\}$, and the feasible set of the problem in (3.11) as $X_T^2 = \{x_T | x_i \in C_i, \forall i \in \mathcal{R}_b \cap \mathcal{V}_T\}$. First we show the following conclusion regarding the cases when the optimization problem in (3.10) is feasible.

**Proposition 3.1.** *If $x_T^*$, a solution to (3.11), satisfies constraints in (3.9), the optimization problem (3.10) is feasible and $x_T^*$ is also its optimizer.*

*Proof.* Because $x_T^*$ is an optimizer of the optimization problem in (3.9), then $x_T^* \in X_T^2$. Furthermore, $x_T^*$ satisfies the constraints in (3.9), then $x_T^* \in X_T^1$. Obviously, there is at least one element in $X_T^1$. Therefore, the optimization problem in (3.10) is feasible. Based on the conditions that $X_T^1 \subset X_T^2$, $x_T^* \in X_T^1$, we conclude that $x_T^*$ is also an optimizer of the optimization problem in (3.10). □

Then we have another conclusion to determine the cases when the optimization problem in (3.10) is infeasible.

**Proposition 3.2.** *If $x_T^*$ does not satisfy the constraints in (3.9), the optimization problem (3.10) is infeasible.*

*Proof.* Because $x_T^*$ is an optimizer of the optimization problem (3.11), we have $f(x_T^*) \le f(x_T),\ \forall x_T \in X_T^2$. With the fact that $X_T^1 \subset X_T^2$, we also have $f(x_T^*) \le f(x_T),\ \forall x_T \in X_T^1$. When $x_T^*$ does not satisfy the conditions in (3.9), $f(x_T^*) > R - \delta$. Based on the above analysis, we have $f(x_T) > R - \delta$, $\forall x_T \in X_T^1$, which is equivalent to $\forall x_T \in X_T^1$, $\exists \{i,j\} \in \mathcal{E}_T$ s.t. $f_{ij}(x_T) > R - \delta$. Therefore, $X_T^1$ is empty, which means the optimization problem in (3.10) is infeasible. □

Based on the Propositions 3.1 and 3.2, instead of solving the optimization problem in (3.10), first, the optimization problem in (3.11) is solved. If this solution satisfies conditions in (3.9), it is termed *feasible*. Otherwise, it is *infeasible*, which means we need to select a new tree and possibly introduce new routers to support the persistent connectivity of clients. If the obtained solution is feasible, we set the desired positions of the active routers to be the same as this solution.

In the following two parts, we present two distributed algorithms to solve the optimization problem (3.11).

**Primal Sub-gradient Method to Solve the Optimization Problem (3.11)**

In this part, we use a primal sub-gradient method to solve the optimization problem (3.11), which uses the sub-gradients of the primal objective function $f(x_T)$ to compute the descent direction to solve (3.11). This method is implemented in a distributed way as described in what follows. Every agent $i$ on the selected tree runs a local algorithm as presented in the Algorithm 2 only by using its local information. In each iteration of Algorithm 2, agent $i$ first obtains the set of the longest edges of the tree $T$, which is denoted as $\mathcal{M}_k$, by a local estimation procedure (lines $3-9$, Algorithm 2). Here $k = 0, 1, \dots$ denote the indices of iterations. The estimation procedure is proceeded by communicating with agent $i$'s neighbours, and $\sigma_i$ denotes the local estimate of the set of the longest edges (lines $3-8$, Algorithm 2). Then $x_i$ is updated locally by using a sub-gradient corresponding to $\mathcal{M}_k$ if agent $i$ is a router and an ending vertex of an edge in $\mathcal{M}_k$, and $x_i$ keeps the current value (lines $10-14$, Algorithm 2) otherwise. In the following we first present the primal sub-gradient method, then we address the details of Algorithm 2 based on the method.

Let $[\nabla f_{ij}]_l = \frac{\partial f_{ij}}{\partial x_l}$, then $[\nabla f_{ij}]_l = 0$ for $l \notin \{i,j\} \cap \mathcal{R}_b$, $[\nabla f_{ij}]_i = \frac{(x_i - x_j)}{\|x_i - x_j\|}$ if $i \in \mathcal{R}_b$, and $[\nabla f_{ij}]_j = \frac{(x_j - x_i)}{\|x_i - x_j\|}$ if $j \in \mathcal{R}_b$. In the optimization algorithm, $k$ denotes the $k_{th}$ iteration, and

**Algorithm 2** Local Algorithm of Agent $i$ with primal Sub-gradient method for solving (3.11).

```
1: $x_i[0] \leftarrow x_i^0, k \leftarrow 0$
2: **while** A termination condition for (3.11) is not satisfied **do**
3:     $k' \leftarrow 0$
4:     $\sigma_i(0|k) = \left\{\{i,j\}|\{i,j\} \in \arg\max\left\{f_{ij}(x_T[0]), j \in \mathcal{N}_i^T\right\}\right\}$
5:     **while** $k' \leq |\mathcal{V}_T| - 1$ **do**                ▷ This loop obtains the longest edges.
6:         $\sigma_i(k'+1|k) = \left\{\{i,j\}|\{i,j\} \in \arg\max\left\{f_{ij}(x_T[k])|\{i,j\} \in \sigma_i(k'|k), \{i,j\} \in \sigma_j(k'|k), j \in \mathcal{N}_i^T\right\}\right\}$
                                   ▷ Local estimation of the longest edges for each $i$.
7:         $k' \leftarrow k' + 1$
8:     **end while**
9:     $\mathcal{M}_k = \sigma_i(k'|k)$
10:    **if** $i \in \mathcal{R}_b$ and $\exists j \in \mathcal{N}_i^T$, such that $\{i,j\} \in \mathcal{M}_k$ **then**
11:        $x_i[k+1] = P_{C_i}\left(x_i[k] + \rho[k] \sum_{j,\{i,j\}\in\mathcal{M}_k} \alpha_{ij}[k] \dfrac{x_j[k] - x_i[k]}{\|x_i[k] - x_j[k]\|}\right),$
12:    **else**
13:        $x_i[k+1] = x_i[k]$
14:    **end if**
15:    $k \leftarrow k + 1$
16: **end while**
```

the $k$ in square brackets is used to denote the corresponding values of the variables at the iteration indexed by $k$, for example $x_T[k]$, $x_i[k]$. Let $\mathcal{M}_k$ denote the set of edges with maximum length at iteration $k$, which is, $\mathcal{M}_k := \left\{\{i,j\}|\{i,j\} \in \arg\max_{\{i,j\}\in\mathcal{E}_T}\left\{f_{ij}(x_T[k])\right\}\right\}$. Based on the preliminaries in Section 3.2, we have a set of sub-gradients of $f(x_T)$ at $x_T[k]$ as follows

$$S_f(x_T[k]) = \left\{s_f | s_f = \sum_{\{i,j\}\in\mathcal{M}_k} \alpha_{ij}[k] \nabla f_{ij}(x_T[k]),\ \alpha_{ij}[k] \geq 0 \ and \sum_{\{i,j\}\in\mathcal{M}_k} \alpha_{ij}[k] = 1\right\}. \quad (3.12)$$

From conclusions of convergence analysis of sub-gradient method in literature, such as [10], we know that the following iterations minimize $f(x_T)$,

$$x_T[k+1] = P_{X_T^2}\left(x_T[k] - \rho[k] g_f(x_T[k])\right), \quad (3.13)$$

where $\rho[k]$ is an appropriately chosen step-length, $g_f(x_T[k]) \in S_f(x_T[k])$, $X_T^2$ is the feasible set of optimization problem (3.11), $P_{X_T^2}(\cdot)$ stands for the projection operation as in

Section 3.2.

With the sub-gradient method given above, the details of Algorithm 2 are described as follows. After initializing $k$ by 0 and $x_i$ by $x_i^0$ (line 1, Algorithm 2), Algorithm 2 goes into iterations (lines $2-16$, Algorithm 2). From the analysis above, it is necessary for the edge with the largest length to be identified first at each iteration. The simplest way to achieve this is that each agent runs the following algorithm at each iteration indexed by $k$ (line 6, Algorithm 2).

$$\begin{aligned}\sigma_i(k'+1|k) = \Big\{\{i,j\}|\{i,j\} \in \arg\max_{\{i,j\}\in\mathcal{E}_T} \{f_{ij}(x_T[k])|\{i,j\} \in \sigma_i(k'|k), \\ (i,j) \in \sigma_j(k'|k), j \in \mathcal{N}_i^T\}\Big\}, \quad \forall i \in \mathcal{V}_T,\end{aligned} \tag{3.14}$$

where $\mathcal{N}_i^T$ is the neighbourhood of $i$ corresponding to the tree $T$. The initial set $\sigma_i(0|k)$ is set as $\sigma_i(0|k) = \{\{i,j\}|\{i,j\} \in \arg\max\{f_{ij}(x_T[0]), j \in \mathcal{N}_i^T\}\}$ (line 4, Algorithm 2). It is easy to show that the algorithm converges to the set $\sigma_i(k)$ at most after $|\mathcal{V}_T|-1$ exchanges, then $\mathcal{M}_k = \sigma_i(k)$ (line 9, Algorithm 2). Thus, as stated in (3.13), all $i \in \mathcal{V}_T \cap \mathcal{R}_b$ such that $\{i,j\} \in \mathcal{M}_k$ update their positions locally according to the following rule (line 11, Algorithm 2):

$$x_i[k+1] = P_{C_i}\left(x_i[k] + \rho[k] \sum_{j,\{i,j\}\in\mathcal{M}_k} \alpha_{ij}[k] \frac{x_j[k]-x_i[k]}{\|x_i[k]-x_j[k]\|}\right), \tag{3.15}$$

here $\alpha_{ij}[k] = \alpha_{ji}[k]$ for $\{i,j\} \in \mathcal{M}_k$, $P_{C_i}(\cdot)$ is the projection operation which is defined in Section 3.2. As stated above, all $i$ will have the information of set $\mathcal{M}_k$, thus a predefined rule of choosing $\alpha_{ij}[k]$ can be stored in each agent $i$. For example, we can simply set them as $\alpha_{ij} = \frac{1}{|\mathcal{M}_k|}$, here $|\mathcal{M}_k|$ denotes the number of elements in $\mathcal{M}_k$. In fact, when running the algorithm, $\mathcal{M}_k$ has only one element at almost all the iterations, thus $a_{ij}[k]$ is set as 1 for all most all the $k$. If agent $i$ does not need to be updated, $x_i$ keeps the current value (line 13, Algorithm 2). After the update of $x_i$, if a termination condition is not satisfied, Algorithm 2 goes into next iteration. Otherwise, we use the current values of $x_T$ as the optimizer.

In Algorithm 2, the choice of step-length $\rho[k]$ has a major impact on the convergence properties of the algorithm. One can choose a constant step-size rule or an adaptive one, e.g. Polyak step-size rule, or a diminishing one. When we choose constant step-size rule, the convergence will be sub-linear of order $O(1/\sqrt{k})$ [10]. However, in practice this might be good enough. Particularly, note that the sub-gradient satisfies,

$$\|g(x_T[k])\| \leq \sum_{\{i,j\}\in\mathcal{M}_k} (\alpha_{ij}[k] + \alpha_{ji}[k]) = 2, \forall k = 0, 1, 2, \ldots. \tag{3.16}$$

Selecting a fixed step-length $\rho$, yields

$$\lim_{k\to\infty} \inf f(x_T[k]) \leq f^\star + 2\rho,$$

where $f^\star$ is the optimal value of the objective function in optimization problem (3.11).

Since the norm of $g(x_T[k])$ is bounded as in (3.16), by using fixed step-size $\rho$, one has the following result for any positive scalar $\varepsilon$ [10],

$$\begin{aligned} &\min_{0\leq k\leq K} f(x_T[k]) \leq f^* + \frac{4\rho+\varepsilon}{2}, \\ &K = \Big\lfloor \frac{d(x_T[0])^2}{\rho\varepsilon} \Big\rfloor, \end{aligned} \tag{3.17}$$

where $d(x_T) = \min_{x_T^*\in X_T^*} \|x_T - x_T^*\|$, $X_T^*$ is the set of all optimizers of (3.11). In our problem, because of the reachable set $C_i$, we have $\|x_i[0] - x_i^*\| \leq t_m\bar{v}_r$, $\forall x_i^* \in X_i^*$, where $X_i^*$ is the set of all values of the element $x_i^*$ of $X_T^*$. Then we have,

$$\begin{aligned} d(x_T[0]) &= \min_{x_T^*\in X_T^*} \|x_T[0] - x_T^*\| \\ &\leq \sum_{i\in\mathcal{V}_T\cap\mathcal{R}_b} \|x_i[0] - x_i^*\|, \\ &\leq |\mathcal{V}_T \cap \mathcal{R}_b|(t_m\bar{v}_r) \end{aligned} \tag{3.18}$$

where $|\mathcal{V}_T \cap \mathcal{R}_b|$ is the number of elements in the set $\mathcal{V}_T \cap \mathcal{R}_b$. By using this upper bound, at most $\lfloor \frac{(|\mathcal{V}_T\cap\mathcal{R}_b|(t_m\bar{v}_r))^2}{2\varepsilon} \rfloor$ is needed to achieve the optimal solution within objective function error $\frac{4\rho+\varepsilon}{2}$.

**Dual Sub-gradient Method to Solve the Optimization Problem (3.11)**

In this part we will use Lagrange dual method to decompose the optimization problem (3.11) so that it can be solved locally. Each agent on the selected tree does the computations to run Algorithm 3 using its local information. In this algorithm, first an internal optimization problem is solved locally to get the information needed to update the dual variables (lines $3-8$, Algorithm 3). Then the dual variables are updated with local information (lines $9-11$, Algorithm 3). In the following, we first present the basis of the dual sub-gradient method with an alternative problem of the optimization problem (3.11). Then we will address the details of Algorithm 3.

**Algorithm 3** An algorithm of agent $i$ by using dual sub-gradient method.

```
1:  Initialization: Choose $\mu_{ij}[0], \lambda_{ij}[0], \theta_{ij}[0], \forall j \in \mathcal{N}_i^T, m \leftarrow 0$.
2:  while A termination condition for $\mu_{ij}, \lambda_{ij}, \theta_{ij}$ is not satisfied do
3:      if $i \in \mathcal{R}_b \cap \mathcal{V}_T$ then
4:          $\big[x_i^{*\mathsf{T}}[m], y_{ij}^{*\mathsf{T}}[m], t_i^{*\mathsf{T}}[m]\big]^\mathsf{T} \leftarrow \arg\min L_i\big(x_i, y_{ij}, t_i, \mu[m], \lambda[m], \theta[m]\big)$,
5:      else
6:          $\big[y_{ij}^{*\mathsf{T}}[m], t_i^{*\mathsf{T}}[m]\big]^\mathsf{T} \leftarrow \arg\min L_i\big(x_i, y_{ij}, t_i, \mu[m], \lambda[m], \theta[m]\big)$,
7:          $x_i^*[m] = x_i$
8:      end if
9:      $\mu_{ij}[m+1] = \mu_{ij}[m] + \beta[m]\Big(-\big(t_i^*[m] - t_j^*[m]\big)\Big)$
10:     $\lambda_{ij}[m+1] = \lambda_{ij}[m] + \beta[m]\Big(-\big(y_{ij}^*[m] - x_j^*[m]\big)\Big)$
11:     $\theta_{ij}[m+1] = P_{\mathbb{R}^+}\Big(\theta_{ij}[m] + \beta[m]\Big(-\big(t_i^*[m] - \|x_i^*[m] - y_{ij}^*[m]\|\big)\Big)\Big)$
12:     $m = m+1$
13: end while
```

We use $n_A := |\mathcal{V}_T|$ to denote the total number of agents on the selected tree, and use $a_i,\ i = 1, \dots, n_A$ to denote the indices of the agents on the tree. Let $n_i := |\mathcal{N}_i^T|$ denote the total number of the neighbours of agent $i$, and $i_l,\ l = 1, \dots, n_i$ denote the indices of the neighbours of agent $i$. Based on the structure of tree $T$, we have an alternative problem of the optimization problem (3.11) whose optimizer is also an optimizer of (3.11). This problem is formulated as follows,

$$\begin{aligned} \min_{x_T,y,t} & \sum_{i\in\mathcal{V}_T} t_i \\ s.t. \quad & t_i = t_j,\ \forall\{i,j\}\in\mathcal{E}_T, \\ & t_i \geq \|x_i - y_{ij}\|,\ \forall i\in\mathcal{V}_T,\ \forall j\in\mathcal{N}_i^T \\ & y_{ij} = x_j,\ \forall i\in\mathcal{V}_T,\ \forall j\in\mathcal{N}_i^T \\ & x_i\in C_i,\ \forall i\in\mathcal{V}_T\cap\mathcal{R}_b \end{aligned} \tag{3.19}$$

where $y_{ij}\in\mathbb{R}^2$, $t_i\in\mathbb{R}$, $C_i$ is the reachable set of agent $i$, $t=[t_{a_1},\dots,t_{a_{n_A}}]^\intercal$, $y=[y_{a_1}^\intercal,\dots,y_{a_{n_A}}^\intercal]^\intercal$, $y_i=[y_{ii_1}^\intercal,\dots,y_{ii_{n_I}}^\intercal]^\intercal$. The optimizers of the optimization problem (3.19) are denoted by $\bar{x}_T^*=[\bar{x}_{r_1}^{*\intercal},\dots,\bar{x}_{r_{n_T}}^{*\intercal}]^\intercal$, $t^*=[t_{a_1}^*,\dots,t_{a_{n_A}}^*]^\intercal$, $y^*=[y_{a_1}^{*\intercal},\dots,y_{a_{n_A}}^{*\intercal}]^\intercal$, $y_i^*=[y_{ii_1}^{*\intercal},\dots,y_{ii_{n_I}}^{*\intercal}]^\intercal$. In the following, we show that the optimizer $\bar{x}_T^*$ is also an optimizer of the optimization problem (3.11).

**Proposition 3.3.** *$\bar{x}_T^*$ is an optimizer of the optimization problem (3.11), that is, $f(\bar{x}_T^*)=f(x_T^*)$, and $f^*=f(x_T^*)=\frac{1}{|\mathcal{V}_T|}t_i^*$.*

*Proof.* First we denote the feasible set of $x_T$ in optimization problem (3.19) as $X_T^3=\{x_T|t_i=t_j,\ \forall\{i,j\}\in\mathcal{E}_T;\ t_i\geq\|x_i-y_{ij}\|,\ \forall i\in\mathcal{V}_T,\ \forall j\in\mathcal{N}_i^T;\ y_{ij}=x_j,\ \forall i\in\mathcal{V}_T,\ \forall j\in\mathcal{N}_i^T;\ x_i\in C_i,\ \forall i\in\mathcal{V}_T\cap\mathcal{R}_b\}$. Because $t_i\in\mathbb{R}$, $y_{ij}\in\mathbb{R}^2$, it is easy to check that $X_T^3=X_T^2$. According to the first three groups of constraints in (3.19), we have $t_l\geq\max_{\{i,j\}\in\mathcal{E}_T}\|x_i-x_j\|,\ \forall l\in\mathcal{V}_T$ and $t_i=t_j,\ \forall\{i,j\}\in\mathcal{E}_T$. Thus we have $\sum_{i\in\mathcal{V}_T}t_i=|\mathcal{V}_T|t_l\geq|\mathcal{V}_T|\max_{\{i,j\}\in\mathcal{E}_T}\|x_i-x_j\|,\ \forall l\in\mathcal{V}_T$ and $\sum_{i\in\mathcal{V}_T}t_i^*=|\mathcal{V}_T|\min_{x_T\in X_T^3}\max_{\{i,j\}\in\mathcal{E}_T}\|x_i-x_j\|$. Based on the relationship $X_T^3=X_T^2$, we have $\sum_{i\in\mathcal{V}_T}t_i^*=|\mathcal{V}_T|f^*$. Additionally, because $\sum_{i\in\mathcal{V}_T}t_i^*=|\mathcal{V}_T|\max_{\{i,j\}\in\mathcal{E}_T}\|\bar{x}_i^*-\bar{x}_j^*\|=|\mathcal{V}_T|f(\bar{x}_T^*)$, we have $f(\bar{x}_T^*)=f^*$. Then $\bar{x}_T^*$ is an optimizer of the optimization problem (3.11) and $f^*=f(x_T^*)=\frac{1}{|\mathcal{V}_T|}t_i^*$. □

This optimization problem in (3.19) is a convex optimization problem because, $\sum_{i\in\mathcal{V}_T}t_i$ is concave, $t_i-t_j,\ y_{ij}-x_j,\ \forall\{i,j\}\in E_T$ are affine, $\|x_i-y_{ij}\|-t_i,\ \forall i\in\mathcal{V}_T,\forall j\in\mathcal{N}_i^T$, $\|x_i-x_i[0]\|-t_m\bar{v}_r,\ \forall i\in\mathcal{V}_T\cap\mathcal{R}_b$ are convex. Furthermore, the equality constraint functions are linear, and inequality constraint functions are convex, thus the Slater's Condition is satisfied and the strong duality holds [12]. Therefore, we can solve the Lagrange dual problem instead of the primal problem in (3.19) to get the optimal value of the objective function and the desired positions of the active routers. The details are described as follows.

The Lagrangian function of the primal optimization problem is expressed as,

$$\begin{aligned}L(x_T,y,t,\mu,\lambda,\theta) &= \sum_{i\in\mathcal{V}_T} t_i - \sum_{i\in\mathcal{V}_T}\sum_{j\in\mathcal{N}_i^T}\mu_{ij}(t_i-t_j) - \sum_{i\in\mathcal{V}_T}\sum_{j\in\mathcal{N}_i^T}\lambda_{ij}^T(y_{ij}-x_j)\\ &- \sum_{i\in\mathcal{V}_T}\sum_{j\in\mathcal{N}_i^T}\theta_{ij}(t_i-\|x_i-y_{ij}\|)\\ &= \sum_{i\in\mathcal{V}_T}\Big[t_i - \sum_{j\in\mathcal{N}_i^T}\mu_{ij}t_i + \sum_{p\in\mathcal{N}_i^T}\mu_{pi}t_i - \sum_{j\in\mathcal{N}_i^T}\lambda_{ij}^T y_{ij}\\ &+ \sum_{p\in\mathcal{N}_i^T}\lambda_{pi}^T x_i - \sum_{j\in\mathcal{N}_i^T}\theta_{ij}(t_i-\|x_i-y_{ij}\|)\Big]\end{aligned} \tag{3.20}$$

where $\mu = [\mu_{a_1}^\intercal,\dots,\mu_{a_{n_A}}^\intercal]^\intercal$, $\mu_i = [\mu_{ii_1},\dots,\mu_{ii_{n_I}}]^\intercal$, $\lambda = [\lambda_{a_1}^\intercal,\dots,\lambda_{a_{n_A}}^\intercal]^\intercal$, $\lambda_i = [\lambda_{ib_1},\dots,\lambda_{ib_{n_I}}]^\intercal$, $\theta = [\theta_{a_1}^\intercal,\dots,\theta_{a_{n_A}}^\intercal]^\intercal$, $\theta_i = [\theta_{ii_1},\dots,\theta_{ii_{n_I}}]^\intercal$ and $\theta_{ij}\geq 0,\ \forall i\in\mathcal{V}_T,\ j\in\mathcal{N}_i^T$. $\mu_{ij},\ \lambda_{ij},\ \theta_{ij}$ are the Lagrangian multipliers. We define $L_i,\ \forall i\in\mathcal{V}_T$ as,

$$\begin{aligned}L_i(x_i,y_{ij},t_i,\mu,\lambda,\theta) :=& t_i - \sum_{j\in\mathcal{N}_i^T}\mu_{ij}t_i + \sum_{p\in\mathcal{N}_i^T}\mu_{pi}t_i - \sum_{j\in\mathcal{N}_i^T}\lambda_{ij}^T y_{ij}\\ &+ \sum_{p\in\mathcal{N}_i^T}\lambda_{pi}^T x_i - \sum_{j\in\mathcal{N}_i^T}\theta_{ij}(t_i-\|x_i-y_{ij}\|)\end{aligned} \tag{3.21}$$

The Lagrange Dual Function $q(\mu,\lambda,\theta)$ is equal to the solution of the following optimization problem,

$$\begin{aligned}&\min_{x_T,y,t} L(x_T,y,t,\mu,\lambda,\theta)\\ &s.t.\quad x_i\in C_i,\ \forall i\in\mathcal{V}_T\cap\mathcal{R}_b\end{aligned} \tag{3.22}$$

which is,

$$q(\mu,\lambda,\theta) = \min_{x_T,y,t} L(x_T,y,t,\mu,\lambda,\theta) \tag{3.23}$$

Therefore, the dual problem of the optimization problem (3.19) is,

$$\begin{aligned} &\min_{\mu,\lambda,\theta} \ -q(\mu,\lambda,\theta) \\ &s.t. \quad \theta_{ij} \geq 0,\ \forall i \in \mathcal{V}_T,\ \forall j \in \mathcal{N}_i^T. \end{aligned} \tag{3.24}$$

From the strong duality and convexity of the optimization problem (3.19), we have $\sum_{i\in\mathcal{V}_T} t_i^*$ $= q*$ and $x_T^*, y^*, t^*$ are primal optimal. Therefore, we can solve the dual problem (3.24) to get the desired positions of inactive routers.

In Algorithm 3, we use $m = 0, 1, \ldots$ to index iterations, and the variables with $m$, such as $\mu[m], \lambda[m], \theta[m]$, denote the values of these variables at iteration $m$. For any iteration $m$, we denote $[x_T^{*\mathsf{T}}[m], y^{*\mathsf{T}}[m], t^{*\mathsf{T}}[m]]^\mathsf{T}$ as the optimizer of the problem (3.22), which is,

$$\big[x_T^{*\mathsf{T}}[m], y^{*\mathsf{T}}[m], t^{*\mathsf{T}}[m]\big]^\mathsf{T} \in \arg\min_{x_T \in X_T^3, y, t} L\big(x_T, y, t, \mu[m], \lambda[m], \theta[m]\big). \tag{3.25}$$

It can be proven that the partial derivative of $-L(z^*[m], y^*[m], t^*[m], \mu, \lambda, \theta)$ with respect to $\mu, \lambda, \theta$ is a sub-gradient of $-q$ at $\mu[m], \lambda[m], \theta[m]$ [10]. This sub-gradient is expressed as,

$$g_d(\mu[m], \lambda[m], \theta[m]) = \frac{\partial L(x_T^*[m], y^*[m], t^*[m], \mu, \lambda, \theta)}{\partial[\mu^\mathsf{T}, \lambda^\mathsf{T}, \theta^\mathsf{T}]^\mathsf{T}}\Bigg|_{\mu=\mu[m],\lambda=\lambda[m],\theta=\theta[m]} \tag{3.26}$$

Furthermore, we define the feasible set of the dual variables as $D = \big\{[\mu^\mathsf{T}, \lambda^\mathsf{T}, \theta^\mathsf{T}]^\mathsf{T} | \theta_{ij} \geq 0,\ \forall i \in \mathcal{V}_T,\ \forall j \in \mathcal{N}_i^T\big\}$. Similar to the sub-gradient method in Algorithm 2, the dual variables can be updated as follows to minimize the dual function.

$$\begin{aligned} [\mu^\mathsf{T}[m+1], \lambda^\mathsf{T}[m+1], \theta^\mathsf{T}[m+1]]^\mathsf{T} =& P_D([\mu^\mathsf{T}[m+1], \lambda^\mathsf{T}[m+1], \theta^\mathsf{T}[m+1]]^\mathsf{T} \\ &- \beta[m] g_d(\mu[m], \lambda[m], \theta[m])), \end{aligned} \tag{3.27}$$

where $\beta[m]$ is an appropriately chosen step-length.

The dual sub-gradient method is implemented locally, and the local algorithm running on an agent $i$ on the selected tree is Algorithm 3. In this algorithm, we use $m = 0, 1, \ldots$ to index iterations, and the variables with $m$, such as $\mu[m], \lambda[m], \theta[m]$, denote the values of these variables at iteration $m$. First the initial values of $\mu_{ij}$, $\lambda_{ij}$, $\theta_{ij}$ are chosen and $m$ is initialized by 0 (line 1, Algorithm 3). In the next step, we solve the optimization problem (3.22) locally in order to obtain a sub-gradient of the dual problem (3.24) (lines $3-8$, Algorithm 3). If $i$ is a router except the base, it calculates the elements $x_i^*[m], y_i^*[m], t_i^*[m]$ of the optimizers of problem (3.22) (line 4, Algorithm 3), which are denoted by,

$$[x_i^{*\mathsf{T}}[m], y_i^{*\mathsf{T}}[m], t_i^{*\mathsf{T}}[m]]^{\mathsf{T}} \leftarrow \arg\min L_i(x_i, y_{ij}, t_i, \mu[m], \lambda[m], \theta[m]). \tag{3.28}$$

If $i$ is a client or the base, only $y_i^*[m], t_i^*[m]$ need to be calculated (line 6, Algorithm 3),

$$[y_i^{*\mathsf{T}}[m], t_i^{*\mathsf{T}}[m]]^{\mathsf{T}} \leftarrow \arg\min L_i(x_i, y_{ij}, t_i, \mu[m], \lambda[m], \theta[m]). \tag{3.29}$$

Then the variables $\mu_{ij}[m], \lambda_{ij}[m], \theta_{ij}[m]$ are updated as follows based on the update rule in (3.27) (lines $9-11$, Algorithm 3).

$$\begin{aligned}
\mu_{ij}[m+1] &= \mu_{ij}[m] + \beta[m](-(t_i^*[m] - t_j^*[m])) \\
\lambda_{ij}[m+1] &= \lambda_{ij}[m] + \beta[m](-(y_{ij}^*[m] - x_j^*[m])) \\
\theta_{ij}[m+1] &= P_{\mathbb{R}^+}(\theta_{ij}[m] + \beta[m](-(t_i^*[m] - \|x_i^*[m] - y_{ij}^*[m]\|))),
\end{aligned} \tag{3.30}$$

where $x_i^*[m] = x_i$ if $i \in \mathcal{C} \cup \{b\}$ (line 7, Algorithm 3). We use $x_i^*[m]$ for all $i \in \mathcal{V}_T$ is just to simplify the expressions. If a termination condition is not satisfied, Algorithm 3 steps into the next iteration, otherwise the current values of $x_i,\ \forall i \in \mathcal{V}_T \cap \mathcal{R}_b$ are set as the desired positions of corresponding the active routers.

By letting agent $i$ keep the values of local variables $\mu_{ij}, \lambda_{ij}, \theta_{ij}, y_{ij},\ \forall j \in \mathcal{N}_i^T$ and $x_i, t_i$, the updates in (3.30) can be processed locally only by using its own information $\mu_{ij}[m], \lambda_{ij}[m], \theta_{ij}[m], y_{ij}^*[m], x_i^*[m], t_i^*[m]$ and its neighbours' information $t_j^*[m],\ x_j^*[m]$.

*Remark:* The proposed dual sub-gradient algorithm can be implemented locally because we can also calculate the optimizers $x_T^*[m], y^*[m], t^*[m]$ in (3.25) locally. The Lagrangian function $L(z, y, t, \mu[m], \lambda[m], \theta[m])$ can be written as the summation of $L_i,\ \forall i \in \mathcal{V}_T$, which is defined in equation (3.21). It is obvious that there is no coupling between $L_i$ regarding the variables $x_i, y_{ij}, t_i$. Therefore, we can decompose the minimization problem in (3.22) and solve it locally. In this way, an agent $i$ only needs the values of variables of its own and $\mu_{ij}[m], \mu_{pi}[m], \lambda_{ij}[m], \lambda_{pi}[m], \theta_{ij}[m],\ j, p \in \mathcal{N}_i^T$ from agent $i$'s neighbours when solving the problem (3.22).

In the above statement, we assume we can exactly obtain $x_i^*[m], y_{ij}^*[m], t_i^*[m]$. Then the choice of step-length and convergence analysis of this dual sub-gradient method are the same as in the part 3.4.1 of Algorithm 2. However, in practical implementation, we can only calculate their approximation. Let $x_i[m], y_{ij}[m], t_i[m]$ denote the approximation, we have,

$$L(x_T[m], y[m], t[m], \mu[m], \lambda[m], \theta[m]) \leq L(x_T^*[m], y^*[m], t^*[m], \mu[m], \lambda[m], \theta[m]) + \epsilon_m, \tag{3.31}$$

where $\epsilon_m$ is the error between the optimal value and approximation value of $L$ at iteration $m$. Therefore,

$$\begin{aligned}
-q(\bar{\mu}, \bar{\lambda}, \bar{\theta}) &= -\min_{x_T, y, t} L(x_T, y, t, \bar{\mu}, \bar{\lambda}, \bar{\theta}) \\
&\geq -L(x_T[m], y[m], t[m], \bar{\mu}, \bar{\lambda}, \bar{\theta}) \\
&\geq -L(x_T[m], y[m], t[m], \mu[m], \lambda[m], \theta[m]) \\
&\quad + h^\intercal(\mu[m], \lambda[m], \theta[m])([\bar{\mu}^\intercal, \bar{\lambda}^\intercal, \bar{\theta}^\intercal]^\intercal - [\mu^\intercal[m], \lambda^\intercal[m], \theta^\intercal[m]]^\intercal) \\
&\geq -L(x_T^*[m], y^*[m], t^*[m], \mu[m], \lambda[m], \theta[m], \eta[m]) - \epsilon_m \\
&\quad + h^\intercal(\mu[m], \lambda[m], \theta[m])([\bar{\mu}^\intercal, \bar{\lambda}^\intercal, \bar{\theta}^\intercal]^\intercal - [\mu^\intercal[m], \lambda^\intercal[m], \theta^\intercal[m]]^\intercal)
\end{aligned} \tag{3.32}$$

where $h(\mu[m], \lambda[m], \theta[m]) := \left.\frac{\partial L(x_T[m], y[m], t[m], \mu, \lambda, \theta)}{\partial[\mu^\intercal, \lambda^\intercal, \theta^\intercal]^\intercal}\right|_{\mu=\mu[m], \lambda=\lambda[m], \theta=\theta[m]}$, $\epsilon = \limsup_{m\to\infty} \epsilon_m$. From the above analysis, we conclude that $h(\mu[m], \lambda[m], \theta[m])$ is a $\epsilon$-sub-gradient of $q(\mu, \lambda, \theta)$ at $[\mu^\intercal[m], \lambda^\intercal[m], \theta^\intercal[m]]^\intercal$ [10].

If we choose a constant step length as $\beta$ for the update of $\mu, \lambda, \theta$, that is $\beta[m] = \beta,\ \forall m = 0, 1, \ldots$, then we have the following result [41],

$$\lim_{m\to\infty} \inf q(\mu[m], \lambda[m], \theta[m]) \leq q^* + \frac{\beta\gamma^2}{2} + \epsilon \tag{3.33}$$

where $\gamma \geq \sup\{\|h(\mu[m], \lambda[m], \theta[m])\| | m = 0, 1, \ldots\}$. Based on this analysis, we conclude that the optimization problem can still be solved approximately with some bounded error in practical implementation.

### 3.4.2 Desired Positions of Routers in $S_I$

In this subsection, we present Algorithms 4 and 5 to obtain desired positions of the inactive routers and make sure that the inactive routers do not lose connection to the selected tree. Algorithm 4 constructs the extended tree by connecting every inactive router to a nearby router on the selected tree. In the extended tree obtained, because of the method used in Algorithm 4, every inactive router has one and only one neighbour which is a router or the base on the selected tree. Based on the extended tree, Algorithm 5 calculates the desired positions of the inactive routers after the active routers obtain their desired positions from optimization algorithms. The conclusion that the inactive routers always connect to the selected tree by using Algorithms 4 and 5 will be presented in subsection 3.4.5. Algorithms 4 and 5 are also applied in different places in the general Algorithm 1 (lines $5, 21, 31, 35$, Algorithm 1). For clarity, we use simplified notation without some superscripts or subscripts. For example, we use $T$ for simplicity of $T^{\kappa}$ or $T^{\kappa}_{n_u}$, $\mathcal{E}_{T_E}$ for $\mathcal{E}_{T^{\kappa}_E}$ or $\mathcal{E}^{\kappa}_{T_{E n_u}}$.

At the start of Algorithm 4, the extended tree is initialized by $T$, and a variable $flag_i$ is used to denote if the inactive router $i$ has been added to $T_E$ (line 1, Algorithm 4). Note that router $i$ only needs to hold the information of the neighbourhoods $\mathcal{N}_i$, $\mathcal{N}^T_i$, $\mathcal{N}^{T_E}_i$. Inactive router $i$ will first check if there exists a neighbour in $\mathcal{N}_i$ that is an agent on the selected tree (lines $2 - 13$, Algorithm 4). If a neighbour $j$ in $\mathcal{N}_i$ is a router or the base (line 3, Algorithm 4), the inactive router $i$ will choose such a neighbour as its neighbour on $T_E$ (line 4) and set $flag_i$ as 1 (line 5, Algorithm 4). If a neighbour $j$ of $i$ is a client (line 8, Algorithm 4), router $i$ will choose a neighbour $l$ in $\mathcal{N}^T_j$ as its neighbour on $T_E$ (line 9, Algorithm 4) and set $flag_i$ as 1 (line 10, Algorithm 4). If $flag_i$ is still 0, agent $i$ will keep checking if any of its neighbours in $\mathcal{N}_i$ has already been added to $T_E$ (lines $14 - 22$, Algorithm 4). If a neighbour $j$'s $flag_j$ is set as 1, it means $j$ has been already added to $T_E$ (line 16, Algorithm 4). Then $i$ will add $\{i, l\}$ to $\mathcal{E}_{T_E}$ and add $i$ to $\mathcal{V}_{T_E}$, where $l$ is the only neighbour of $j$ on $T_E$ (line 17, Algorithm 4). Thus $flag_i$ is set as 1 (line 18, Algorithm 4).

**Algorithm 4** An algorithm of deriving $T_E$ for $i \in S_I$

```
1: Initialization: $T_E = T, flag_i = 0$
2: for $j \in \mathcal{N}_i$ do
3:     if $j \in \mathcal{V}_T \cap \mathcal{R}$ then
4:         Add $j$ to $\mathcal{V}_{T_E}$, add the edge $\{i, j\}$ to $\mathcal{E}_{T_E}$
5:         $flag_i = 1$
6:         Break and step out of the current $for$ loop
7:     end if
8:     if $j \in \mathcal{C}$ then
9:         Add $i$ to $V_{T_E}$ and the edge $\{i, l\}$ to $T_E$, where $l \in \mathcal{N}_j^T$
10:        $flag_i = 1$
11:        Break and step out of the current $for$ loop
12:    end if
13: end for
14: while $flag_i = 0$ do
15:    for $j \in \mathcal{N}_i$ do
16:        if $flag_j = 1$ then
17:            Add $i$ to $V_{T_E}$ and the edge $\{i, l\}$ to $T_E$, where $l \in \mathcal{N}_j^{T_E}$
18:            $flag_i = 1$
19:            Break and step out the current $for$ loop
20:        end if
21:    end for
22: end while
23: Wait until $flag_i = 1$ for all $i \in S_I$
24: Steer $i$ to the position of $l$, that is $x_l(t)$, where $l \in \mathcal{N}_j^{T_E}$
```

When all $flag_i$, $\forall i \in S_I$ are set as 1, router $i$ will be steered to the current position of its neighbour in $\mathcal{N}_{T_E}$ (lines $23 - 24$, Algorithm 4). Here, $x_k(t)$ denotes the current position of the agent $k$. Note that when the extended tree is constructed by Algorithm 4, every inactive router has one and only one neighbour on $T_E$, which is a router in $\mathcal{V}_T$.

In Algorithm 5, we use $x_j^*$ to stand for the desired positions calculated by the optimization algorithms if $j \in \mathcal{V}_T \cap \mathcal{R}_b$, and use $x_c$ to stand for the current desired position of the base. If the neighbour $j$ of $i$ on $T_E$ is not the base, the desired position of the inactive router $i$ is set as $x_j^*$ (line 1, Algorithm 5) (lines $1 - 3$, Algorithm 5). Otherwise $i$ choose $x_b$ as its desired position (lines $4 - 6$, Algorithm 5).

*Remark:* By choosing the desired positions of the inactive routers using Algorithm 5, a situation may arise where the distance between the current position and the desired position of a router is larger than $t_m \bar{v}_r$. Note that, $t_m \bar{v}_r$ is the largest distance that a router

**Algorithm 5** An algorithm of deriving the desired position for $i \in S_I$

```
1: if $j \in \mathcal{R}$, where $j \in \mathcal{N}_i^{T_E}$ then
2:     $x_i^* = x_j^*$
3: end if
4: if $j = b$, where $j \in \mathcal{N}_i^{T_E}$ then
5:     $x_i^* = x_c$
6: end if
```

is assumed to move during the time interval $t_m$ as in Algorithm 1. This situation happens because the tracking controllers we design in Subsection 3.4.4 will typically have tracking errors. Detailed explanation for this case and how to deal with it will be presented in Subsection 3.4.4. As we will see in the subsection 3.4.5, the routers are always connected to the tree $T(t)$ by using Algorithms 4 and 5. Therefore, when the solution of the optimization problem (3.10) is infeasible, the information can reach the inactive routers through other agents, and the inactive routers can be used to update $T(t)$.

### 3.4.3 Tree and Graph Update

In this subsection, we present Algorithms 6 and 8 for updating of the selected tree in the reinitialization stages as stated in Algorithm 1 (line 18, Algorithm 1). If the number of the active routers is less than $N_c \left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} + 1 \right\rceil + 1$, we use Algorithm 6 to update $T$. It will steer an inactive router towards the longest edge corresponding to the desired positions of the active routers, and add this router to the set of the active routers. Otherwise, we use Algorithm 8 to rebuild the selected tree by connecting every client to the base separately. In this subsection, we will first present the details of the two algorithms and then show that the number of routers needed is always less than $N_r$ (see Assumption 3.4).

In this subsection, as in subsection 3.4.1, we simpl the notation. Furthermore, we use $x_i$ to stand for $x_i(t_0^\kappa + n^\kappa \Delta + n_u(t_x + t_u))$ if $i \in \mathcal{V}_T \setminus \{b\}$, use $x_b$ to stand for $x_c(t_0^\kappa + n^\kappa \Delta)$ and use $x_j^*$ to stand for the desired positions calculated by the optimization algorithms right before the execution of Algorithm 6 or 8 if $j \in \mathcal{V}_T \cap \mathcal{R}_b$. Note that the time instant $t_0^\kappa + n\Delta$ is also denoted by $t_0^{\kappa+1} - \Delta_u^{\kappa+1}$ as shown in Figure 3.1 and 3.3.

In Algorithm 6, first, one of the longest edges is identified and is denoted by $\{i, j\}$ (line 1, Algorithm 6). The same method as in Algorithm 2 (lines $3-8$, Algorithm 2) can be used to achieve this purpose. Then, an inactive router $l$ will be identified such that $l$ is the closest inactive router to the middle point of the longest edge (line 2, Algorithm 5). The identification of $l$ can be processed locally through the direct communication between the agents in the system. Details of this identification procedure is illustrated in Algorithm 7. Note that the execution of Algorithm 7 requires a period of time covering $|\mathcal{V}_T| - 1$ times of direct communication and this time cost is counted in the time interval $t_u$ (see Figure 3.3). After router $l$ is selected, it is steered to the position $\frac{x_i+x_j}{2}$ (line 3, Algorithm 6). When router $l$ reaches the destination, edge $\{i,j\}$ is deleted from $\mathcal{E}_T$ and $\mathcal{E}_{T_E}$ (line 4, Algorithm 6). Meanwhile, $\{i,l\}$, $\{l,j\}$ are added to these two edge sets and $l$ is added to $\mathcal{V}_T$ (line 5, Algorithm 6). Finally, the sets of the active routers and inactive routers are updated by moving $l$ from $S_I$ to $S_A$ (line 6, Algorithm 6). After update as in Algorithm 6, the new edges connecting a client and a router still satisfy the conditions in Assumption 3.6. Moreover, the length of the new edges connecting two routers (include the base) is less than or equal to $R - 2e_r$. These two conclusions will be presented in Lemma 3.2 in Subsection 3.4.5.

**Algorithm 6** An algorithm of update of the tree

```
1: Find an $\{i,j\}$ such that $f_{ij}(x_T^*) = f(x_T^*)$
2: Determine $l \in S_I$ such that $\left\|x_l - \frac{x_i+x_j}{2}\right\| = \min_{k,k\in S_I} \left\|x_k - \frac{x_i+x_j}{2}\right\|$ (refer to Algorithm 7)
3: Steer router $l$ to the position $\frac{x_i+x_j}{2}$
4: $\mathcal{E}_T \leftarrow \mathcal{E}_T \setminus \{\{i,j\}\}, \mathcal{E}_{T_E} \leftarrow \mathcal{E}_{T_E} \setminus \{\{i,j\}\}$
5: $\mathcal{V}_T \leftarrow \mathcal{V}_T \cup \{l\}, \mathcal{E}_T \leftarrow \mathcal{E}_T \cup \{\{i,l\},\{l,j\}\}, \mathcal{E}_{T_E} \leftarrow \mathcal{E}_{T_E} \cup \{\{i,l\},\{l,j\}\}$
6: $S_A \leftarrow S_A \cup \{k\}, S_I \leftarrow S_I \setminus \{k\}$,
```

**Algorithm 7** A local algorithm of determining an inactive router for tree update

```
1: $k' \leftarrow 0$
2: $\phi_l(0) = \{l\}$ if $l \in S_I$, $\phi_l(0) = \emptyset$ otherwise.
3: while $k' \le |\mathcal{V}_T| - 1$ do
4:     $\phi_l(k'+1) = \left\{\{l\} | \{l\} \in \arg\max\left\{\|x_l - \frac{x_i+x_j}{2}\|\,|\,l \in \phi_l(k'), l \in \phi_p(k'), p \in \mathcal{N}_i^T\right\}\right\}$
5:     $k' \leftarrow k' + 1$
6: end while
```

If $|S_A| \ge N_c \lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} + 1 \rceil + 1$, we use Alogrithm 8 to update the selected tree. This algorithm will connect all clients to the base through separate paths, which means there is no common router between any two paths except the base. First, we define a set of indices of agents $\bar{S}_A$ and initialize it with an empty set (line 1, Algorithm 8). For a client $i$, one of its neighbours in $\mathcal{N}_i^T$ is chosen, which is denoted by agent $j$ (line 3, Algorithm 8). Then, the distance $d_i$ between $x_j$ and $x_b$ is calculated (line 4, Algorithm 8). Based on $d_i$, a variable $k_i$ is calculated, which determines the number of routers we will choose for the path connecting $i$ and the base $b$ (line 5, Algorithm 8). Then a group of inactive routers are chosen, their number is $k_i + 1$ and we use $i_1, \ldots, i_{k_i+1}$ to denote their indices (line 6, Algorithm 8). Selection of these $k_i + 1$ inactive routers is processed locally as illustrated in Algorithm 9. Next the indices of the chosen inactive routers are removed from $S_I$, meanwhile these indices are added to the set $\bar{S}_A$ to be stored (lines $7-8$, Algorithm 8). The process of choosing such a group of routers will continue until this has been completed for all the clients (lines $2-9$, Algorithm 8). In the next step, the inactive routers with indices $i_1, \ldots, i_{k_i}$ are steered to the positions which are evenly distributed on the straight line connecting the neighbour $j$ of $i$ and the base $b$ (line 10, Algorithm 8). An extra router $k_i + 1$ is also appointed to the position $x_j$, where $j$ is the neighbour chosen for the client $i$ (line 11, Algorithm 8). This steering process (lines $10-11$, Algorithm 8) is addressed for every $i \in \mathcal{C}$. After steering, the tree $T$, $S_A$ and $S_I$ will be re-defined and based on the new positions of the chosen inactive routers (lines $12-13$, Algorithm 8). Then, all the routers in the new inactive set $S_I$ will be steered to the position $x_b$ (line 14, Algorithm 8). After that, a new extended tree is constructed (line 15, Algorithm 8). After the selected tree is updated by Algorithm 8, the length of each updated edge is less than or equal to $R - 2e_r$. This conclusion will be presented in Lemma 3.3 in Subsection 3.4.5.

*Remark:* Algorithms 6 and 8 present the procedures of tree update on system level. In execution, these two algorithms are processed locally as described above and in Algorithms 7 and 9. In Algorithm 6, the first step (line 1, Algorithm 6) is executed locally as stated above and the second step (line 2, Algorithm 6) is executed locally by having each agent execute Algorithm 7. Moreover, the positions of the corresponding agents in the sets $\{i, j\}$ and $\phi_j$ can be broadcast to each agent simultaneously with the processes of identifying these sets. Thus the third step (line 3, Algorithm 6) can be processed locally. By noting that each agent only keeps the set of its neighbours regarding the tree graph

**Algorithm 8** An algorithm of update of the whole graph $G$

```
1: $\bar{S}_A = \emptyset$
2: for $i \in \mathcal{C}$ do
3:     Choose $j$ such that $j \in \mathcal{N}_i^T$
4:     $d_i = \|x_j - x_c\|$
5:     $k_i = \lceil \frac{d_i}{R-\delta-2e_r} \rceil$
6:     Choose $k_i + 1$ routers from $S_I$ and the indices of them are denoted as $i_1, \ldots, i_{k_i+1}$
       (refer to Algorithm 9). Broadcast the indices of these selected routers to all the agents
       through $|\mathcal{V}| - 1$ times of direct communication.
7:     $S_I \leftarrow S_I \setminus \{i_1, \ldots, i_{k_i+1}\}$
8:     $\bar{S}_A \leftarrow \bar{S}_A \cup \{i_1, \ldots, i_{k_i+1}\}$
9: end for
10: Steer router $i_l$ to $x_b + l\frac{x_j - x_b}{k_i}$, $\forall l \in \{1, \ldots, k_i\}$, $\forall i \in \mathcal{C}$
11: Steer router $i_{k_i+1}$ to the position $\frac{x_i + x_j}{2}$, where $j \in \mathcal{N}_i^T$ is the same neighbour of $i$ chosen
    in the $for$ loop above
12: $S_A \leftarrow \bar{S}_A$, $\mathcal{V}_T \leftarrow S_A$, $\mathcal{E}_T = \{\{i_l, i_{l+1}\} | \forall i \in \mathcal{C}, \forall l \in \{1, \ldots, k_i\}\} \cup \{\{i_1, b\} | \forall i \in \mathcal{C}\} \cup$
    $\{\{i_{k_i+1}, i\} | \forall i \in \mathcal{C}\}$
13: $S_I \leftarrow \mathcal{R}_b \setminus \bar{S}_A$
14: Steer router $q$ to the base position $x_b$, $\forall q \in S_I$
15: $\mathcal{V}_{T_E} \leftarrow \mathcal{V}_T$, $\mathcal{E}_{T_E} = \mathcal{E}_T \cup \{\{q, b\} | \forall q \in S_I\}$
```

**Algorithm 9** A local algorithm of selecting inactive routers for client $i$ in graph update

```
1: $k' \leftarrow 0$
2: $\psi_j(0) = \{j\}$ if $j \in S_I$, $\psi_j(0) = \emptyset$ otherwise.
3: while $|\psi_j| < k_i + 1$ do
4:     $\psi_j(k'+1) = \{\{l\} | l \in \psi_j,\ l \in \psi_p,\ p \in \mathcal{N}_j^T\}$
5:     $k' \leftarrow k' + 1$
6: end while
7: Denote the indices in $\psi_j$ as $\psi_j = \{i_1, \ldots, i_{k_{\psi_j}}\}$ with $k_{\psi_j} \geq k_i + 1$ and $i_p < i_{p+1},\ 1 \leq p \leq$
   $k_{\psi_j} - 1$. Then select the first $k_i + 1$ routers from $\psi_j$ as the selected inactive routers.
```

structure and each router keeps the flag whether it is an inactive router or an active router, the other steps (lines 3-6, Algorithm 6) can be executed locally. In Algorithm 8, the sixth step (line 6, Algorithm 8) is executed locally by having each agent execute Algorithm 9. The other steps can be executed locally for the same reason as stated for Algorithm 6.

In the following, we will prove that the number of routers needed in Algorithms 6 and 8 is less than or equal to $2N_c\left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} + 1 \right\rceil + 1$. This conclusion is presented in Proposition 3.4.

**Proposition 3.4.** *The following statements are true regarding Algorithms 6 and 8: the number of inactive routers needed to update the selected tree is less than or equal to* $N_c\left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r}+1\right\rceil$*; the number of the active routers on the updated selected tree is always less than or equal to* $N_c\left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r}+1\right\rceil+1$*; the total number of routers (active and inactive routers) is less than or equal to* $2N_c\left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r}+1\right\rceil+1$.

*Proof.* Define $N_a$ as $N_a =: N_c\left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r}+1\right\rceil+1$. Algorithm 6 is executed when $|S_A| < N_a$ and only one inactive router can be added to the selected tree. Therefore, the number of active routers on the updated selected tree by Algorithm 6 is less than or equal to $N_a$ and the number of inactive routers needed in Algorithm 6 is 1. Obviously, the number of routers needed in Algorithm 6 is less than $2N_a - 1$.

Now we prove the conclusion of inactive router number regarding Algorithm 8. From Assumption 3.2, we have $\|x_i - x_b\| \leq \frac{D}{2}$ for any client $i$. During the time interval of running Algorithm 8, based on Assumption 3.6, we have $\|x_i - x_j\| \leq R - 2e_r$, where $j \in \mathcal{N}_i^T$ is a chosen neighbour of $i$ as in Algorithm 8 (line 3, Algorithm 8). Therefore $\|x_j - x_b\| \leq \|x_i - x_b\| + \|x_i - x_j\| = \frac{D}{2} + R - 2e_r$. Then we have,

$$k_i \leq \left\lceil \frac{\frac{D}{2}+R-2e_r}{R-\delta-2e_r} \right\rceil \tag{3.34}$$

The condition in (3.34) is satisfied for every client. Since we choose $k_i + 1$ inactive routers regarding a client $i$ (line 6, Algorithm 8), the inactive routers needed for update using Algorithm 8 is less than or equal to $N_a - 1$. There are at most $N_a$ active routers on the original $T$, thus the total number of routers needed is less than or equal to $2N_a - 1$. As the construction of the new selected tree and the new set $S_A$ in Algorithm 8 (lines $12 - 13$, Algorithm 8), it can be shown that the number of the active routers in the new tree is less than or equal to $N_a$. Note that here the active routers include the base. □

### 3.4.4 Routers' Desired Trajectories and Control Requirements

In this subsection, we define the routers' desired trajectories, and provide control requirements on their transient behaviour. In this chapter, we will not consider collision and obstacle avoidance.

Next, we introduce the routers' desired trajectories. We define desired trajectory on different time intervals $t_m$, $t_u$ and $t_c$ (Figures 3.2 and 3.3), on which the routers move. Note that the routers do not move on the other time intervals $t_x$ and $t_d$ (Figures 3.2 and 3.3). Below, we will use $x_i^*$ to denote the desired position of router $i$, $x_i(t)$ to denote the position of agent $i$ at time instant $t$ and $\hat{x}_i$ to denote the actual position agent $i$ reaches at the end of the corresponding time interval.

Next, we present the properties of the desired trajectory during the time interval $t_m$ (Figures 3.2 and 3.3). We use $t_n$ to denote the time instant $t_0^\kappa + n\Delta + t_x + t_d$ (line 37, Algorithm 1) or $t_0^\kappa + n\Delta + t_x + n_u^\kappa(t_u + t_x) + t_d$ (line 27, Algorithm 1), which is the initial time instant of this time interval.

For a router $i \in \mathcal{R}$, we define its desired trajectory during $(t_n, t_n + t_m]$ as

$$r_i(t) := x_i(t_n) + \frac{t - t_n}{t_m}\Big(x_i^* - x_i(t_n)\Big), \tag{3.35}$$

and define tracking error at time instant $t$ as $e_i(t) := x_i(t) - r_i(t)$. In order to guarantee the communication connectivity of clients, we have the following requirement

$$\|e_i(t)\| \le e_r, \forall i \in \mathcal{R},\ \forall t \in (t_n, t_n + t_m], \tag{3.36}$$

where $e_r$ is a positive constant. This constant $e_r$ should satisfy

$$e_r \le \frac{1}{2} t_x \bar{v}_r. \tag{3.37}$$

*Remark:* The desired position for the base at $(n+1)\Delta$ is the centroid of client positions, that is, $x_c(n\Delta)$. According to the definition of $x_c(t)$, note that $\|\dot{x}_c(t)\| \le v_c$. Based on Assumption 3.1, the distance the base needs to move is less than $t_m \bar{v}_r$.

In the following, the routers' desired trajectories in time intervals $t_u$ and $t_c$ (Figures 3.2 and 3.3) are characterised. We use $t_s$ to denote the initial time instant of the movement of the corresponding inactive routers.

We define the desired trajectory of router $i \in S_I$ as

$$r_i(t) = x_i(t_s) + \frac{x_i^* - x_i(t_s)}{\|x_i^* - x_i(t_s)\|}(t - t_s)\bar{v}_r,\ t \in \Big(t_s, t_s + \frac{\|x_i^* - x_i(t_s)\|}{\bar{v}_r}\Big]. \tag{3.38}$$

The requirement for the transient behaviour of inactive router $i$ on these time intervals is

$$\|e_i(t)\| \leq e_r,\ i \in S_I,\ \forall t \in \Big(t_s, t_s + \frac{\|x_i^* - x_i(t_s)\|}{\bar{v}_r}\Big]. \tag{3.39}$$

### 3.4.5 Persistent Connectivity of Clients

In this subsection, under previous assumptions, we prove that the clients in the system are persistently connected with the control strategy we design in Algorithm 1. First we choose $\delta$ in (3.6) and (3.7) as

$$\delta := \max\big\{(\Delta + t_x + t_d)v_c + 2e_r, 4e_r\big\}, \tag{3.40}$$

and we choose $\delta_G$ in (3.4) and (3.5) as

$$\delta_G := \max\big\{(t_x + t_d)v_c + e_r, 2e_r\big\}. \tag{3.41}$$

The following conclusions are all based on this choice of $\delta$ and $\delta_G$. First, we present the results of communication connectivity of clients during optimization and control stages and reinitialization stages. Finally, the persistent connectivity of clients will be proved.

#### Communication Connectivity of Clients During Optimization and Control Stage $\Delta$

Lemma 3.1 states that the corresponding selected tree is always connected during time interval $t_m$ (Figures 3.2 and 3.3).

**Lemma 3.1.** *Suppose Assumptions 3.1, 1.1, 1.3, 3.3, and 3.6 hold and let the motion of routers satisfy the requirements in (3.36). Let $\delta$ and $\delta_G$ be defined as in (3.40) and (3.41), respectively. Then, $\|x_i(t) - x_j(t)\| \leq R,\ \forall\{i,j\} \in \mathcal{E}_T^\kappa,\ \forall t \in \big(t_0^\kappa + n\Delta, t_0^\kappa + (n+1)\Delta\big],\ \forall\kappa = 0, 1, \ldots,\ \forall n = 0, \ldots, n^\kappa - 1$.*

*Proof.* In this proof, we will use $x_i^*(t_0^\kappa + n\Delta)$ to denote the desired position of the active router $i$ calculated during time interval $(t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ and $x_b^*(t_0^\kappa + n\Delta)$ to denote $x_c(t_0^\kappa + n\Delta)$.

*Case 1*: First, we consider the edge $\{i, j\}$ connecting a router and a client, $i \in \mathcal{R}, j \in \mathcal{C}$.

Consider $n = 0$, with Assumptions 3.3 and 3.6, we have

$$\|x_i(t) - x_j(t)\| \le R - (t_x + t_d)v_c - e_r + (t_x + t_d)v_c = R - e_r, \ \forall t \in (t_0^\kappa, t_0^\kappa + t_x + t_d].$$

Moreover, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ where $n > 0$, we have

$$\|x_i(t) - x_j(t)\| \le R - \delta + (\Delta + t_x + t_d)v_c + e_r \le R - e_r.$$

The above inequalities hold because the routers stay static and only clients move during these time intervals. Let $x_i^*(t_0^\kappa + n\Delta)$ be a solution to the optimization problem (3.11) considered in time interval $(t_0^\kappa + n\delta, t_0^\kappa + n\Delta + t_x]$. We have

$$\|x_i^*(t_0^\kappa + n\Delta) - x_j(t_0^\kappa + n\Delta + t_x + t_d)\| \le R - \delta + (t_x + t_d)v_c \le R - \Delta v_c - 2e_r.$$

During any time interval $(t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, with the trajectory $r_i(t)$ in (3.35) and requirement (3.36) in Subsection 3.4.4, the length of edge $\{i, j\}$ satisfies the following condition for all $t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$

$$\begin{aligned}
\|x_i(t) - x_j(t)\| =& \left\| x_i(t_0^\kappa + n\Delta + t_x + t_d) + \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m}(x_i^*(t_0^\kappa + n\Delta) - x_i(t_0^\kappa + n\Delta)) \right. \\
& \left. + e_i(t) - x_j(t) \right\| \\
=& \|(1-\alpha)(x_i(t_0^\kappa + n\Delta + t_x + t_d) - x_j(t)) + \alpha(x_i^*(t_0^\kappa + n\Delta) - x_j(t)) + e_i(t)\| \\
\le& (1-\alpha)(R - e_r + \alpha t_m v_c) + \alpha(R - \Delta v_c - 2e_r + \alpha t_m v_c) + e_r \\
=& R + \alpha(t_m v_c - \Delta v_c - e_r) \\
\le& R
\end{aligned},$$

where $\alpha = \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m}$ and $0 \le \alpha \le 1$. The largest value on the right hand side of this inequality is achieved when $\alpha = 0$.

Therefore, $\|x_i(t) - x_j(t)\| \leq R$, $\forall t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, $\forall \{i,j\} \in \{\{i,j\} | i \in \mathcal{R}, j \in \mathcal{C}, \{i,j\} \in \mathcal{E}_{T^\kappa}\}$, $\forall \kappa = 0, 1, \dots, \forall n = 0, \dots, n^\kappa - 1$.

*Case 2*: Now, we consider the edges $\{i,j\}$ with $i, j \in \mathcal{R}$.

Consider $n = 0$, with Assumptions 3.3 and 3.6, we have

$$\|x_i(t) - x_j(t)\| \leq R - 2e_r \; \forall t \in (t_0^\kappa, t_0^\kappa + t_x + t_d]. \tag{3.42}$$

Moreover, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ where $n > 0$, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta + 2e_r \leq R - 2e_r. \tag{3.43}$$

The above inequalities hold because the routers and base stay static during the time intervals considered.

The desired position calculated by the optimization problem satisfies the following condition for any $\kappa,\ n$

$$\|x_i^*(t_0^\kappa + n\Delta) - x_j^*(t_0^\kappa + n\Delta)\| \leq R - \delta \leq R - 4e_r.$$

During any time interval $(t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, with the trajectory $r_i(t)$ in (3.35) and requirement (3.36) in Subsection 3.4.4, the length of edge $\{i,j\}$ satisfies the following condition for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$

$$\begin{aligned}\|x_i(t) - x_j(t)\| =& \|[x_i(t_0^\kappa + n\Delta + t_x + t_d) + \alpha(x_i^*(t_0^\kappa + n\Delta) - x_i(t_0^\kappa + n\Delta + t_x + t_d)) + e_i(t)] \\ &- [x_j(t_0^\kappa + n\Delta + t_x + t_d) + \alpha(x_j^*(t_0^\kappa + n\Delta) - x_j(t_0^\kappa + n\Delta + t_x + t_d)) + e_j(t)]\| \\ \leq& (1-\alpha)\|x_i(t_0^\kappa + n\Delta + t_x + t_d) - x_j(t_0^\kappa + n\Delta + t_x + t_d)\| \\ &+ \alpha\|x_i^*(t_0^\kappa + n\Delta) - x_j^*(t_0^\kappa + n\Delta)\| + \|e_i(t)\| + \|e_j(t)\| \\ \leq& (1-\alpha)(R - 2e_r) + \alpha(R - 4e_r) + 2e_r \\ \leq& R\end{aligned},$$

where $\alpha$ is defined as before in this proof.

Therefore, $\|x_i(t) - x_j(t)\| \leq R$, $\forall t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, $\forall \{i,j\} \in \{\{i,j\} | i \in \mathcal{R}, j \in \mathcal{R}, \{i,j\} \in \mathcal{E}_{T^\kappa}\}, \forall \kappa = 0, 1, \ldots, \forall n = 0, \ldots, n^\kappa$.

With the analysis of *Case 1* and *Case 2*, we conclude that $\|x_i(t) - x_j(t)\| \leq R$, $\forall \{i,j\} \in \mathcal{E}_T^\kappa$, $\forall t \in (t_0^\kappa + n\Delta, t_0^\kappa + (n+1)\Delta]$, $\forall \kappa = 0, 1, \ldots,$ $\forall n = 0, \ldots, n^\kappa - 1$. □

## Communication Connectivity of Clients during Reinitialization Stage $\Delta_u$

In this part, we demonstrate that the clients remain connected during reinitialization stage $\Delta_u$ (lines $10 - 33$, Algorithm 1). Here, we drop the explicit dependence of $\Delta_u$ on $\kappa$ to simplify the notation. First, we present that the corresponding selected tree is connected during the tree update process when Algorithm 6 or 8 is executed in time interval $t_u$ (Figure 3.3). Then, it will be shown that the corresponding selected tree is always connected during reinitialization stage $\Delta_u$ in Lemma 3.4 .

Note that in Lemma 3.2, we use the same simplified notations as in Algorithm 6.

**Lemma 3.2.** *Suppose Assumptions 1.1, 1.3, 3.3, 3.5 and 3.6 hold and let the motion of routers satisfy the requirement in (3.39). Let $\delta$ and $\delta_G$ be defined as in (3.40) and (3.41), respectively. After $\{i,j\} \in \mathcal{V}_T$ is updated as in Algorithm 6, the length of the new edges $\{i,l\}, \{l,j\}$ satisfies $\|x_i - x_l\| \leq R - t_d v_c - e_r$, $\|x_l - x_j\| \leq R - t_d v_c - e_r$ if $i \in \mathcal{C}$ or $j \in \mathcal{C}$, and $\|x_i - x_l\| \leq R - 2e_r$, $\|x_l - x_j\| \leq R - 2e_r$ if $i \in \mathcal{R}$ and $j \in \mathcal{R}$.*

*Proof. Case 1*: First, we consider the edge $\{i,j\}$ with $i \in \mathcal{C}$ or $j \in \mathcal{C}$. With Assumptions 3.3 and 3.6, we have $\|x_i - x_j\| \leq R - t_d v_c - e_r$. After the router $l$ is steered to its destination and $\{i,l\}$ is added to the edge set $\mathcal{E}_T$, with control requirement (3.39) and Assumption 3.5, we have

$$\|x_i - x_l\| \leq \frac{\|x_i - x_j\|}{2} + e_r \leq \frac{R - t_d v_c - e_r}{2} + e_r \leq R - t_d v_c - e_r \tag{3.44}$$

Moreover, with Assumption 3.6, the edge $\{l,j\}$ also satisfies $\|x_l - x_j\| \leq R - t_d v_c - e_r$.

*Case 2:* Now, we consider the edge $\{i,j\}$ with $i \in \mathcal{R}$ and $j \in \mathcal{R}$.

At the start time instant of $\Delta_u$ (Figure 3.1), with the feasible condition (3.9) and definition of $\delta$ in (3.40), we have

$$\|x_i - x_j\| \leq R - \delta + 2e_r \leq R - 2e_r. \tag{3.45}$$

After the execution of Algorithm 6 for the first time, under the control requirement (3.39) and Assumption 3.5, we obtain

$$\|x_i - x_l\| \leq \frac{\|x_i - x_j\|}{2} + 2e_r \leq \frac{R - 2e_r}{2} + 2e_r \leq R - 2e_r \tag{3.46}$$

The above inequality shows that the length of any edge $\{i, j\}$ with $i, j \in \mathcal{R}$ on the new selected tree still satisfy the condition in (3.45). By induction, we know that after every time Algorithm 6 is executed in $\Delta_u$ (Figures 3.3 and 3.1), the conditions in (3.46) are always satisfied. Moreover, with the same analysis as above, we also have $\|x_l - x_j\| \leq R - 2e_r$.

Combining *Case 1* and *Case 2* completes the proof. □

In the following Lemma 3.3, we establish the connectivity of the selected tree after the execution of Algorithm 8. Note that in this lemma, we still use the same notation as in Algorithm 8.

**Lemma 3.3.** *Suppose Assumptions 1.1, 1.3, 3.3, 3.5 and 3.6 hold and let the motion of routers satisfy the requirement in (3.39). Let $\delta$ and $\delta_G$ be defined as in (3.40) and (3.41), respectively. After the selected tree is updated by Algorithm 8, we have $\|x_{i'} - x_{j'}\| \leq R - t_d v_c - e_r$, $\forall \{i', j'\} \in \mathcal{E}_T$, here $\mathcal{E}_T$ denotes the updated edge set of the updated selected tree.*

*Proof.* First, we consider an edge $\{i', j'\} \in \{\{i_l, i_{l+1}\} | \forall i \in \mathcal{C}, \forall l \in \{1, \ldots, k_i - 1\}\}$ after the selected tree is updated by Algorithm 8. With the steering operation in Algorithm 8 (lines 10, Algorithm 8) and the control requirement (3.39), we have $\|x_i' - x_j'\| \leq R - \delta - 2e_r + 2e_r = R - \delta < R - t_d v_c - e_r$. Next, we consider the other edges. Since Algorithm 8 steers $i_{k_i+1}$ to the position $\frac{x_i + x_j}{2}$ (line 11, Algorithm 8), with Assumptions 3.3, 3.5 and 3.6 and control requirement (3.39), we have $\|x_{k_i} - x_{k_i+1}\| \leq \frac{R - t_d v_c - e_r}{2} + 2e_r < R - t_d v_c - e_r$. Moreover, with Assumption 3.6, we have $\|x_i - x_{k_i+1}\| \leq R - t_d v_c - e_r$. With the above analysis, the conclusion in this lemma is satisfied. □

*Remark:* From Assumption 3.1, we have

$$(t_x + t_m)v_c + e_r < t_m \bar{v}_r. \tag{3.47}$$

Assumption 3.5 implies

$$R - t_d v_c - e_r - \big((t_x + t_m)v_c + e_r\big) \leq R - \delta, \tag{3.48}$$

and

$$\frac{R - t_d v_c - e_r}{2} + 2e_r + \big((t_x + t_m)v_c + e_r\big) < R - \delta. \tag{3.49}$$

Let $\bar{x}_i$ denote the position of client $i$ used in the optimization algorithm right after the running of Algorithm 8. Then, Assumptions 3.3 and 3.6 , control requirements (3.47) and (3.48), yield

$$\begin{aligned}\left\| \Big( x_{k_i+1} + \frac{\bar{x}_i - x_{k_i+1}}{\|\bar{x}_i - x_{k_i+1}\|} \big((t_x + t_m)v_c + e_r\big) \Big) - \bar{x}_i \right\| &< R - t_d v_c - e_r - \big((t_x + t_m)v_c + e_r\big) \\ &\leq R - \delta,\end{aligned}$$

and

$$\begin{aligned}\left\| \Big( x_{k_i+1} + \frac{\bar{x}_i - x_{k_i+1}}{\|\bar{x}_i - x_{k_i+1}\|} \big((t_x + t_m)v_c + e_r\big) \Big) - x_{k_i} \right\| &\leq \frac{R - t_d v_c - e_r}{2} + 2e_r + \big((t_x + t_m)v_c + e_r\big) \\ &< R - \delta.\end{aligned}$$

Moreover, as in the proof of Lemma 3.3, $\|x_i - x_j\| < R - \delta, \forall \{i', j'\} \in \big\{\{i_l, i_{l+1}\} | \forall i \in \mathcal{C}, \forall l \in \{1, \dots, k_i - 1\}\big\}$. Consider optimization problem (3.11), with the above analysis, we conclude that feasible solutions exist. This implies that after running Algorithm 8, the while loop (lines $17 - 24$, Algorithm 1) terminates.

In the light of Lemmas 3.2 and 3.3, in Lemma 3.4 we show that the selected tree is connected during reinitialization stages. We will use $\mathcal{E}_T^{\kappa}(t)$ to denote the current set of edges of the selected tree at time instant $t$ during the time interval $\big(t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1}\big]$. For example, $\mathcal{E}_T^{\kappa}(t) = \mathcal{E}_{T_0^{\kappa}}$ for all $t \in \big(t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1}\big]$.

**Lemma 3.4.** *Suppose Assumptions 1.1, 1.3, 3.5 and 3.6 hold and let the motion of routers satisfies* *the control requirement in (3.39). Let $\delta$ and $\delta_G$ be defined as in (3.40) and (3.41), respectively. Then, $\|x_i(t) - x_j(t)\| \leq R$, $\forall\{i,j\} \in \mathcal{E}_T^\kappa(t)$, $\forall t \in \left(t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1}\right]$, $\forall\kappa = 0,1,\ldots$.*

*Proof.* First, we consider the time interval $(t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1} - t_d - t_m - t_c]$ (Figure 3.3). Under Assumptions 3.3 and 3.6 and Lemmas 3.2 and 3.3, we have the following two results, $\|x_i(t) - x_j(t)\| \leq R - t_d v_c - e_r$, $\forall\{i,j\} \in \left\{\{i,j\}|\{i,j\} \in \mathcal{E}_T^\kappa(t), i \in \mathcal{C} \text{ or } j \in \mathcal{C}\right\}$, $\forall t \in (t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1} - t_d - t_m - t_c]$, $\forall\kappa = 0,1,\ldots$; $\|x_i(t) - x_j(t)\| \leq R - 2e_r$, $\forall\{i,j\} \in \left\{\{i,j\}|\{i,j\} \in \mathcal{E}_T^\kappa(t), i \in \mathcal{R} \text{ and } j \in \mathcal{R}\right\}$, $\forall t \in \left(t_0^{\kappa+1} - \Delta_u^{\kappa+1}, t_0^{\kappa+1} - t_d - t_m - t_c\right]$, $\forall\kappa = 0,1,\ldots$.

Then, we consider time interval $\left(t_0^{\kappa+1} - t_d - t_m - t_c, t_0^{\kappa+1} - t_m - t_c\right]$ (Figure 3.3). Since only clients move during this time interval, for any time instant $t$ in it we have the following results. If $\{i,j\} \in \mathcal{E}_T^\kappa(t)$ with $i \in \mathcal{C} \text{ or } j \in \mathcal{C}$, then $\|x_i(t) - x_j(t)\| \leq R - t_d v_c - e_r + t_d v_c = R - e_r$. If $\{i,j\} \in \mathcal{E}_T^\kappa(t)$ with $i \in \mathcal{R} \text{ and } j \in \mathcal{R}$, then $\|x_i(t) - x_j(t)\| \leq R - 2e_r$.

Now, we consider time interval $\left(t_0^{\kappa+1} - t_m - t_c, t_0^{\kappa+1} - t_c\right]$ (Figure 3.3). Using the same arguments as in the proof of Lemma 3.1, yields $\|x_i(t) - x_j(t)\| \leq R$, $\forall\{i,j\} \in \mathcal{E}_T^\kappa(t)$, $\forall t \in \left(t_0^{\kappa+1} - t_m - t_c, t_0^{\kappa+1} - t_c\right]$.

Finally, we consider time interval $\left(t_0^{\kappa+1} - t_c, t_0^{\kappa+1}\right]$ (Figure 3.3). With the fact that active routers do not move during this time interval in reinitialization stage and Assumption 3.6, we have $\|x_i(t) - x_j(t)\| \leq R - t_d v_c - e_r$, $\forall\{i,j\} \in \left\{\{i,j\}|\{i,j\} \in \mathcal{E}_T^\kappa(t), i \in \mathcal{C} \text{ or } j \in \mathcal{C}\right\}$, $\forall t \in \left(t_0^{\kappa+1} - t_c, t_0^{\kappa+1}\right]$, $\forall\kappa = 0,1,\ldots$, and $\|x_i(t) - x_j(t)\| \leq R - 2e_r$, $\forall\{i,j\} \in \left\{\{i,j\}|\{i,j\} \in \mathcal{E}_T^\kappa(t), i \in \mathcal{R} \text{ and } j \in \mathcal{R}\right\}$, $\forall t \in \left(t_0^{\kappa+1} - t_c, t_0^{\kappa+1}\right]$, $\forall\kappa = 0,1,\ldots$.

Based on the above analysis, the conclusion in this lemma holds. □

**Theorem 3.1.** *Suppose Assumptions 3.1–3.6, 1.1 and 1.3 hold and routers satisfy control requirements in (3.36) and (3.39). Let $\delta$ and $\delta_G$ be defined as in (3.40) and (3.41), respectively. Then, by using the control policies in Algorithm 1, the conditions in (3.8) are satisfied, that is, the clients are persistently connected.*

*Proof.* Since the set of indices active and the set of inactive routers do not change other than the update procedures in Algorithms 6 and 8, with Assumptions 3.2 and 3.4 and Proposition 3.4, we conclude that the number of routers is enough for running Algorithm 1. Moreover, with Lemmas 3.1 and 3.4, we conclude that the conditions in (3.8) are satisfied for all $t \geq t_0$. □

Note that, as stated in Remark 3.4.5, under Assumptions 3.1 and 3.6, the while loop (lines $17-24$) in Algorithm 1 will be terminated in finite time.

Finally, we conclude that under Assumptions 3.1–3.5, if the movement of clients satisfies the conditions in Assumption 3.6, Algorithm 1 solves Problem 3.1.

In the above contents of this chapter, we can see that the proofs of all the results are not dependent on the agents' dynamic model (3.2). The reason why we choose this model is to make the statements simplified and clear. As long as the system satisfies the Assumptions 3.1–3.6, 1.1 and 1.3 and the motion controllers for routers satisfy the requirements (3.36) and (3.39), all the results still hold. Furthermore, in practical situations, it is not easy to find such a kind of robot with such a simple model. Therefore, in the simulation in the next section, we choose a more realistic dynamic model to verify our results.

## 3.5 Simulation

The simulation includes two parts. First, we evaluate the time costs of the two optimization algorithms with some randomly generated trees. In the second part, we run the simulation of Algorithm 1 with a specific quadrotor dynamic model.

### 3.5.1 Time Costs of Two Optimization Algorithms

Computation of the routers' desired positions takes a significant proportion of the total time cost of executing Algorithm 1. Thus, we execute Algorithms 2 and 3 in simulation to compare their time costs in this subsection. We use randomly generated trees to test the time cost of the two different optimization algorithms (Algorithms 2 and 3). Since we only consider the time cost of running optimization algorithms, the trees we use are equivalent to the selected trees, and the routers on trees are all treated as active routers. Moreover, the initial positions of the agents are generated randomly. Each coordinate of the position is generated in the range $[0, 10]$. We choose 48 trees with different number of agents for this test. The numbers of agents on the trees are $N_s = 4n,\ n = 2, \ldots, 49$, where a quarter of them are set as clients. The value of $t_m\bar{v}_r$, which is the radius of reachable sets of routers, is set as 2. With multiple trials of different stopping criteria for Algorithms 2 and 3, we choose different criteria for these two algorithms. Each stopping criteria is selected with less time cost and almost the same accuracy. The stopping criterion for Algorithm 2 is $f(x_T[k]) \leq 0.001$. The stopping criterion for Algorithm 3 is $m > 100$. In simulation there are two iteration loops in Algorithm 3. One is the loop to get $x_i^*[m], y_{ij}^*[m], t_i^*[m]$ (lines $3-8$, Algorithm 3), which is called inner loop in our simulations. The other one is the loop for iteration of $\mu_{ij}[m], \lambda_{ij}[m], \theta_{ij}[m]$ (lines $2-13$, Algorithm 3), which is called outer loop. We set the constant number of iterations as 100 for the inner loop, and 100 for the outer loop.

In the simulations, we choose constant step-length for both Algorithms 2 and 3. The step-length chosen for Algorithm 2 is $\rho[k] \equiv 0.01$. Step-lengths of Algorithm 3 are set as $0.01, 0.01$, where the first value is for outer iteration loop, the second value is for inner loop. The choices of number of iterations and step-lengths are based on multiple times of trials.

When considering Algorithm 2, in this simulation results, the relationship between time cost and the number of agents is approximately $t_2^c \approx 0.02N_s$, where $t_2^c$ denotes the time cost of executing Algorithm 2, $N_s$ is the number of agents in the simulation. For Algorithm 3, $t_3^c \approx 0.6N_s$, where $t_3^c$ denotes the time cost of executing Algorithm 3. However, in the simulation, the time cost of communication and the distributed computation among agents have not been considered. In the following, we will analyse the time cost in practical applications based on the simulation results.

Let $\delta_c$ denote the communication time cost of one time exchange of information between an agent and its neighbours. Let $K_1$ denote the number of iteration in the simulation of Algorithm 2, $K_2$ denote the number of iteration of outer loop in the simulation of Algorithm 3. By running the simulation, we calculate that the average value of $K_1$ is 1478. In Algorithm 2, $N_s - 1$ information exchanges are needed to get the longest edge in one iteration (lines $5-8$, Algorithm 2), and 1 exchange is needed to update $x_i[k]$. Moreover, in practical situations, two agents will compute the update of $x_T[k]$ concurrently at almost every $k_{th}$ iteration. Therefore, the actual time cost of Algorithm 2 is

$t_2^a \approx \frac{t_2^c}{2} + K_1 N_s \delta_c \approx 0.01N + K_1 N_s \delta_c$. In Algorithm 3, in each iteration of outer loop, 2 exchanges of information are needed. One exchange is for the inner loop iteration. The other is for the update of dual variables in the outer loop. However, in this algorithm, the computation operates concurrently among all the agents, which means the time cost is approximately equally divided among agents. The actual time cost of Algorithm 3 is $t_3^a \approx \frac{1}{N_s} t_3^c + 2K_2\delta_c \approx 0.6 + 2K_2\delta_c$. The approximate actual time costs of Algorithms 2 and 3 with respect to $N_s$ and $\delta_c$ are shown in Figure 3.4. The grey surface denotes the time cost of Algorithm 2, and the red surface denotes the time cost of Algorithm 3.

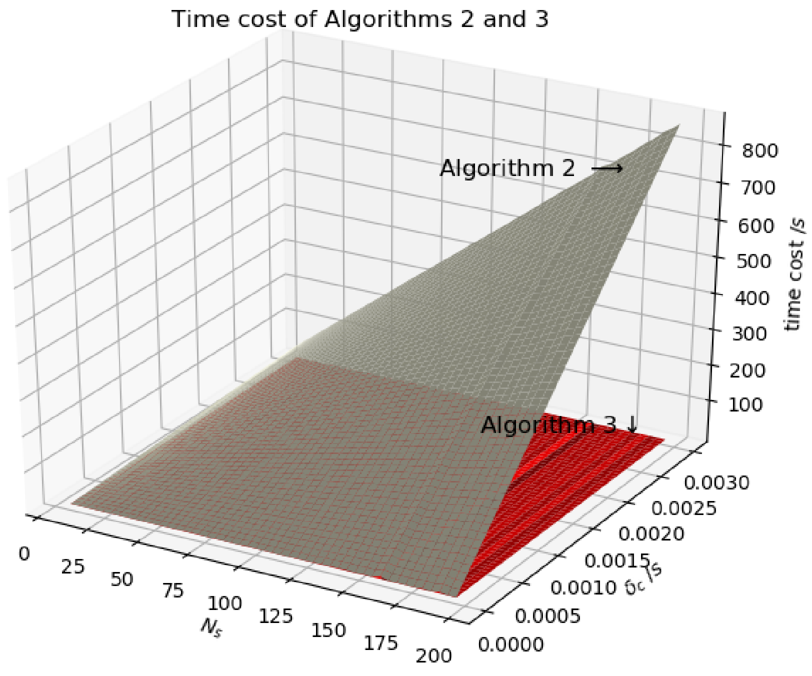


Figure 3.4: Time costs of Algorithms 2 and 3

Wireless communication protocols used in multi-agent system, such as Zigbee, are usually relatively slow. For example, the theoretical speed of Zigbee is $250kb/s$. Then, according to the information needed to be exchanged in each iteration, the order of theoretical magnitude of $\delta_c$ should be $1ms$. In practice, however, $\delta_c$ should be larger than that. In our setting, the time cost of Algorithm 2 is more than the time cost of Algorithm 3 if we consider Zigbee for communication. Based on the above analysis, the time cost of Algorithm 2 mainly comes from communication. However, the time cost of Algorithm 3 mainly comes from computation. Therefore, it is better to use Algorithm 3 if the computational capability of agents is powerful. If the communication is fast enough, Algorithm 2 is a better choice. Furthermore, with the expressions of $t_2^a$ and $t_3^a$, the time cost of Algorithm 3 does not increase with the increasing of the number of agents, but time cost of Algorithm 2 does. Thus Algorithm 3 is more scalable for systems with large number of agents.

### 3.5.2 Persistent Connectivity of Clients

The dynamic model of agents we use is a quadrotor model which comes from [33]. The controller that we use is a PID controller which is also included in [33]. The states of the dynamic model of the agent $i$ are $\xi_i = [\xi_i^1, \xi_i^2, \xi_i^3, \xi_i^4, \xi_i^5, \xi_i^6, \xi_i^7, \xi_i^8, \xi_i^9, \xi_i^{10}, \xi_i^{11}, \xi_i^{12}]^\intercal$, where $[\xi_i^1, \xi_i^2, \xi_i^3]^\intercal$ are the coordinates along longitudinal, lateral, and vertical axes, $[\xi_i^4, \xi_i^5, \xi_i^6]^\intercal$ are the velocities along the corresponding axes, $[\xi_i^7, \xi_i^8, \xi_i^9]^\intercal$ are the roll, pitch and yaw angles, $[\xi_i^{10}, \xi_i^{11}, \xi_i^{12}]^\intercal$ are the angular velocities corresponding to the angles. Thus the planar position of $x_i$ of agent $i$ is defined as $x_i = [\xi_i^1, \xi_i^2]^\intercal$. The reason why we can choose a planar position is that the controller is a hovering controller, which means the quadrotors can be controlled always on the same altitude. The details of the dynamic model and controller design are referred to [33].

*Remark:* The positions can also be chosen as the positions with three dimensions and the distances can be calculated according to this, which will not affect the results.

In this simulation, we choose $\bar{v}_r = 2m/s,\ v_c = 0.2m/s$. By running trajectory tracking with the PID controller [33] for multiple times with different desired positions, we choose $e_r$ as $0.4m$. Other parameters are chosen as follows, $\Delta = 3s,\ t_x = 1s,\ t_d = 0s,\ t_m = 2s,$ $R = 7m, D = 40m, \mathcal{V} = \{0, 1, \ldots, 136\}, \mathcal{C} = \{92, 93, 94, 95, 96, 97, 98, 99, 100, 101\}, t_0 = 0.$ Here, $t_d$ is set as $0s$ because the time cost of Algorithm 5 is too small compared to other time intervals. Router 0 is set as the base, that is $b = 0$. The constant $\delta$ is calculated as $1.6m$. The constant $\delta_G$ is calculated as 0.8. The initial positions of agents are generated randomly. In the initialization stage (lines $1-5$, Algorithm 1), the operations are based on the randomly generated initial positions of agents.

The movement of the clients are governed by the following rules. During a reinitialization stage indexed by $\kappa$, the clients remain stationary during time intervals $(t_0^{\kappa-1} + t_x, t_0^\kappa - t_c - t_m - t_d]$ and $(t_0^\kappa - t_c, t_0^\kappa]$. During the time intervals $t_x$ and $t_m$, the clients move towards randomly chosen destinations. Thus, the trajectories of the clients are set as the straight line connecting their current positions and the corresponding destinations with a constant velocity. These straight line trajectories are generated the same way as in (3.35), only with different time length which is $t_x$ or $t_m$. The random destinations and trajectories are chosen such that they satisfy the requirement of the maximal speed $v_c$ and Assumption 3.2.

With the settings of simulation above, Assumption 3.1 is satisfied with the choice of $\bar{v}_r$ and $v_c$. In this simulation, the communication between agents are set to satisfy Assumption 1.1. With the movement of clients stated above, Assumptions 3.2 and 3.6 are satisfied. Moreover, in the simulation, we assume that the information exchange satisfies Assumption 1.3 and choose $G(t_0)$ such that it satisfies Assumption 3.3. Moreover, $\mathcal{V}$ and $\mathcal{C}$ are selected so that Assumption 3.4 is satisfied and $R$ and $\delta$ satisfy Assumption 3.5. Hence, Assumptions 3.1–3.6 are satisfied.

*Remark:* In this simulation, the selected tree will be only updated using Algorithm 3.4.

Next, we show the simulation results of the control policy in Algorithm 1 with the two different optimization algorithms, that is, Algorithms 2 and 3.

The result of the length of the longest edge of the tree $T(t)$ under Algorithm 2 is shown in Figure 3.5. In this figure, the length of the longest edge on the selected tree is always less than the communication range $R$, which means the clients are persistently connected. The time instants when the curve is not continuous mean when the selected tree is updated. The selected trees and extended trees at some time instants are shown in Figure 3.6. The two axes denote the axes of the coordinate axes of agents' positions with meter as unit. The trees with red edges stand for tree $T(t)$, the trees with green dashed edges are extended trees $T_E(t)$, and the blue dashed circles stand for the boundaries of $B(t)$ as in Assumption 3.2. With the trees in the figure, we can see some inactive routers move to update the selected trees and become active routers.

In Figure 3.7, we show the length of the longest edge on the selected tree when Algo-

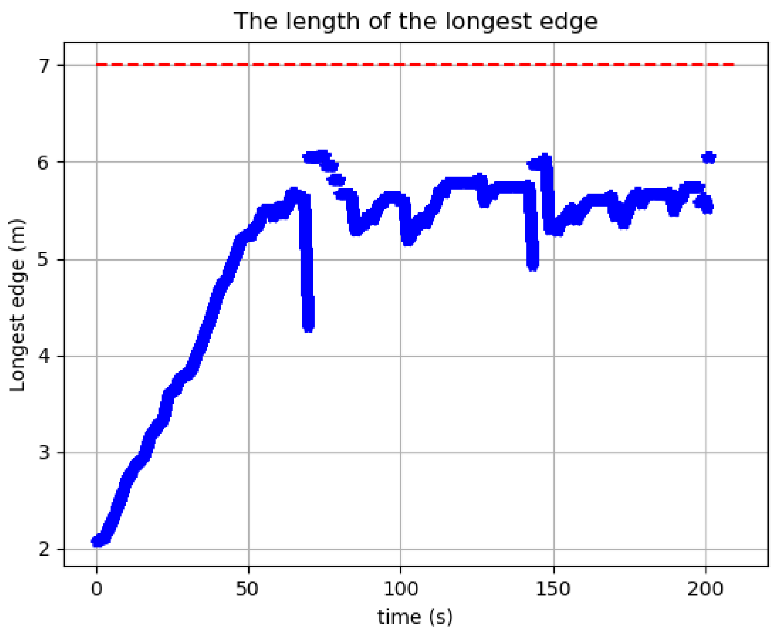


Figure 3.5: Length of the longest edge of $T$ under Algorithm 2

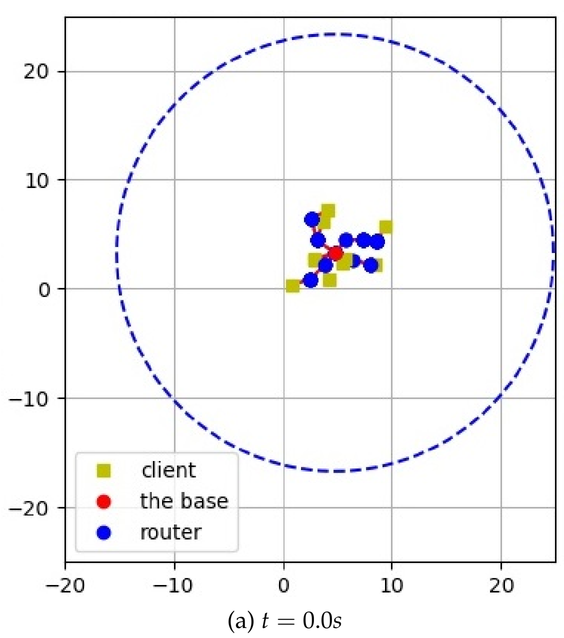
rithm 1 runs with the optimization Algorithm 3. The length is also always less than $R$, which means the clients are persistently connected.

The trees $T$ and $T_E$ at some time instants with Algorithm 3 are shown in Figure 3.8, which shows the varying structure of the trees when Algorithm 1 runs with optimization Algorithm 3.

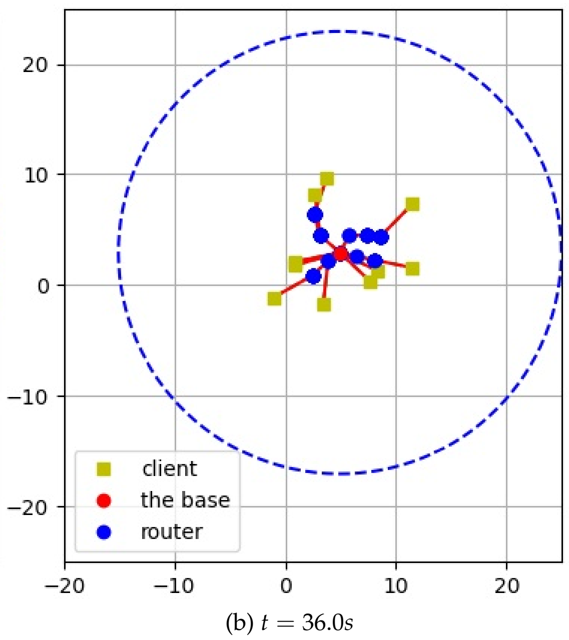
## 3.6 Conclusion

In this work, maintaining persistent connectivity of clients is achieved by running control policy Algorithm 1 for routers. One important feature of our work is that the clients can move without worrying about the communication between each other except some possible reinitialization stages. This makes the clients more suitable for their own tasks in multi-agent systems.

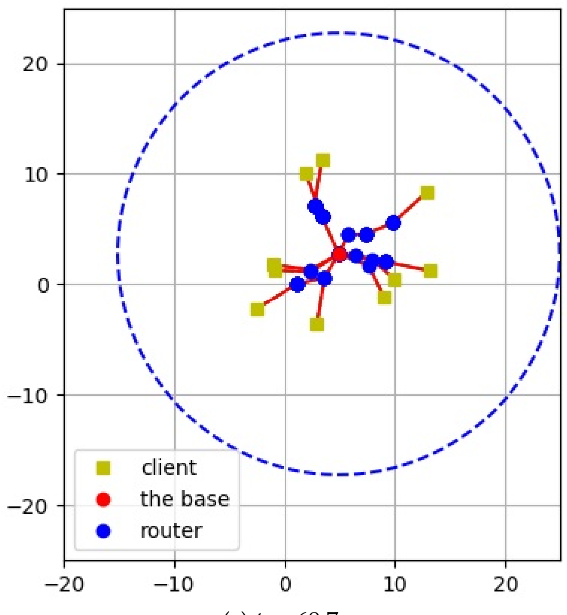
In the proposed strategy, first, a tree is selected from the graph induced by the distances between agents. Then, the optimization algorithms minimize the length of the edges of the selected tree in the worst case, which gives the desired positions of routers



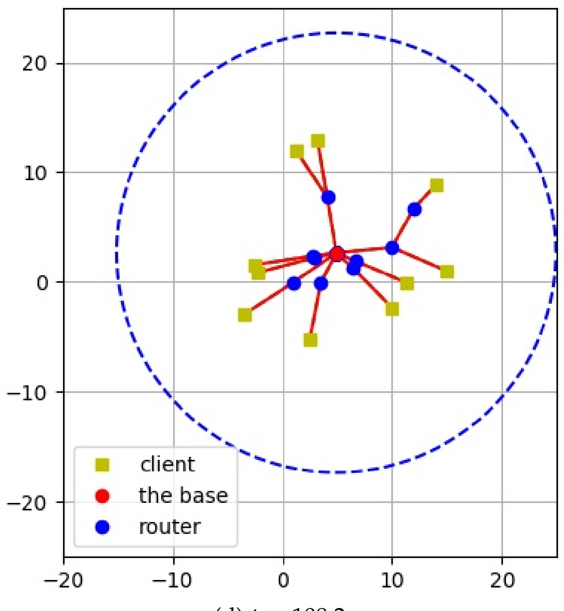
(a) $t = 0.0s$



(b) $t = 36.0s$



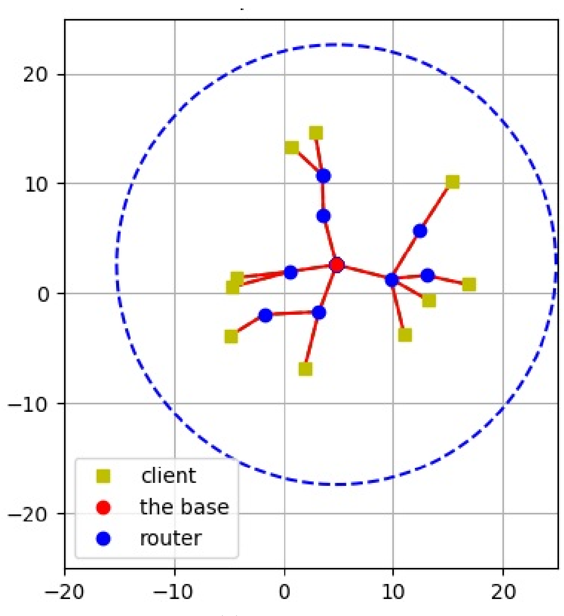
(c) $t = 69.7s$



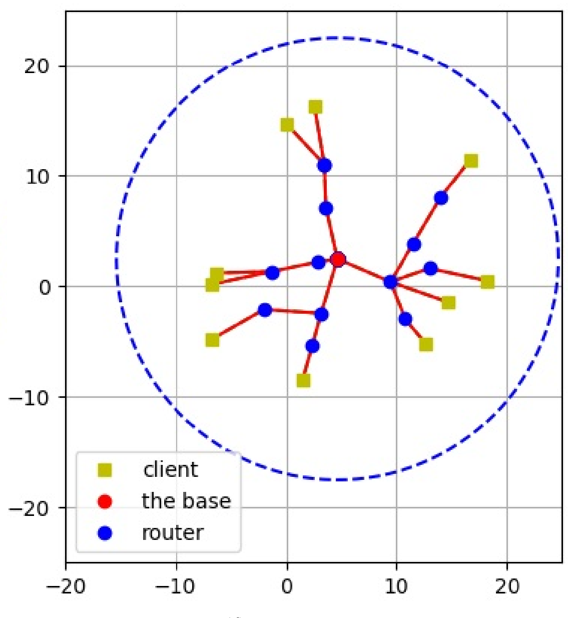
(d) $t = 108.2s$

Figure 3.6: $T(t)$ and $T_E(t)$ during the process of simulation under Algorithm 2

to maintain communication connectivity of clients. When the structure of the selected tree cannot support communication for clients, update of the selected tree handles these situations. After proposing the control policy, we analyse it to show that it guarantees the persistent connectivity of clients. Finally, a simulation example is given to demonstrate



(e) $t = 154.8s$



(f) $t = 200.0s$

Figure 3.6: $T(t)$ and $T_E(t)$ during the process of simulation under Algorithm 2

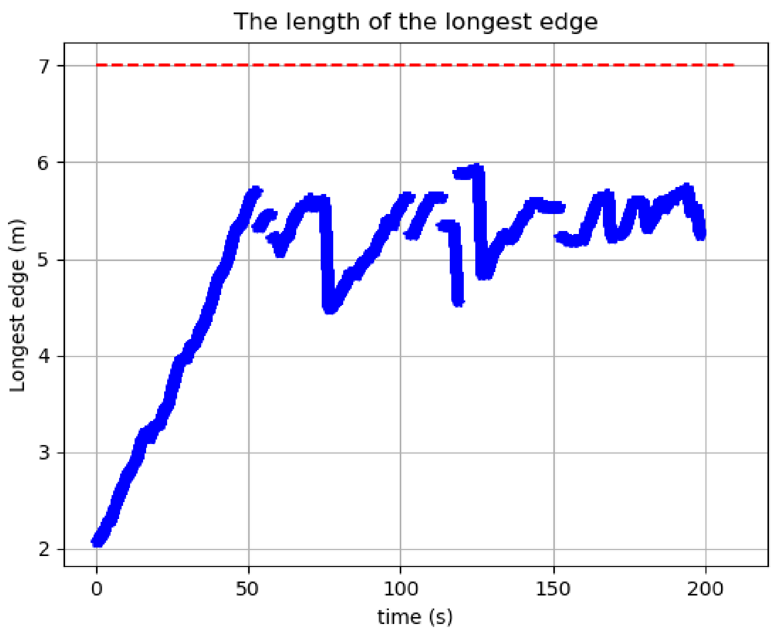
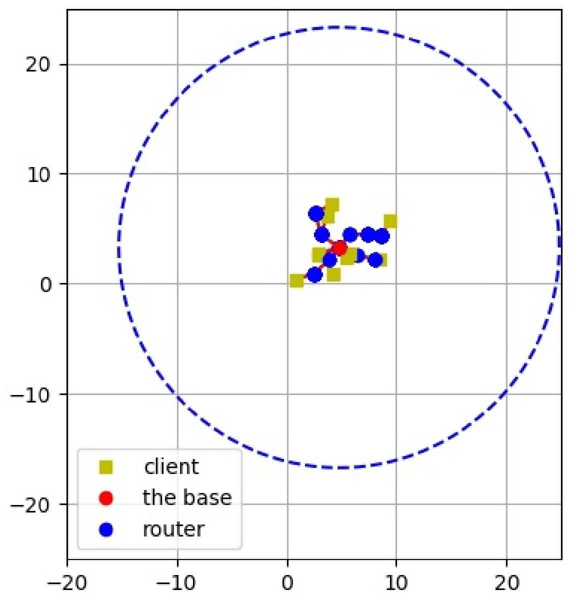
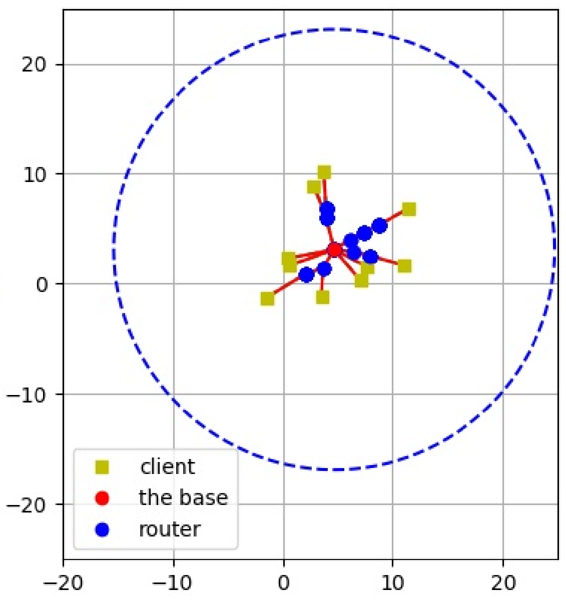


Figure 3.7: Length of the longest edge of $T$ under Algorithm 3

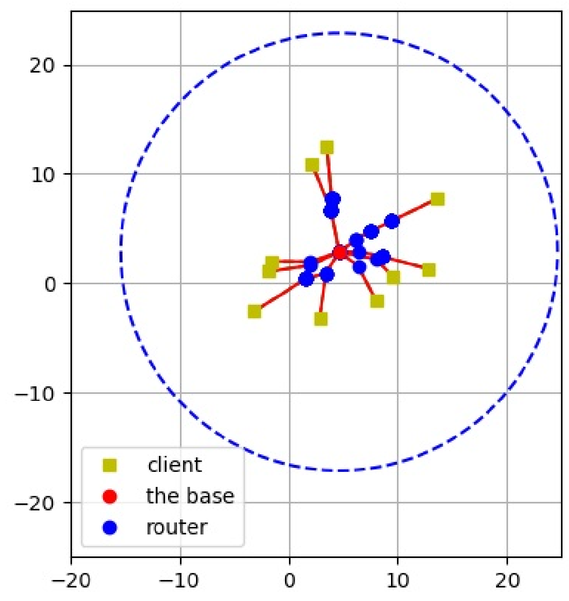
the performance of our control policy.



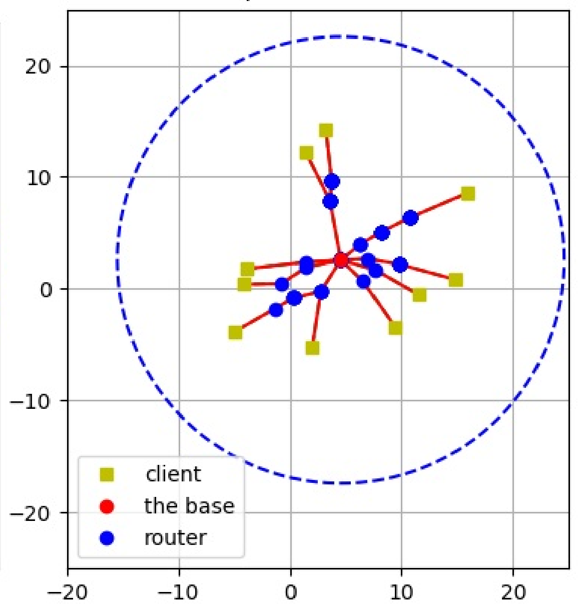
(a) $t = 0.0s$



(b) $t = 36.0s$



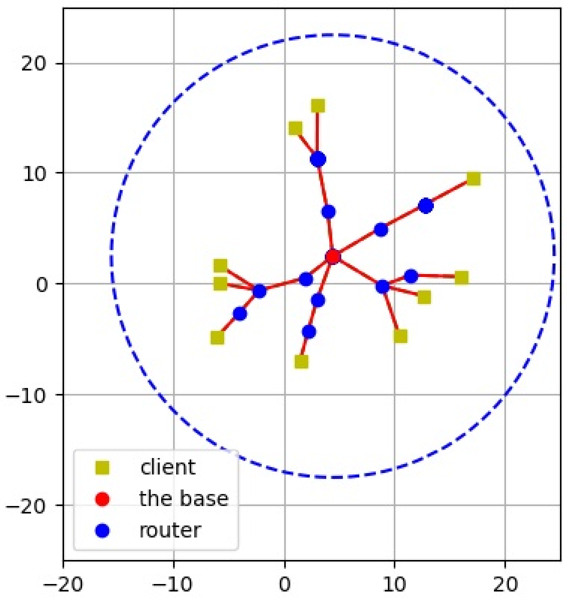
(c) $t = 73.4s$

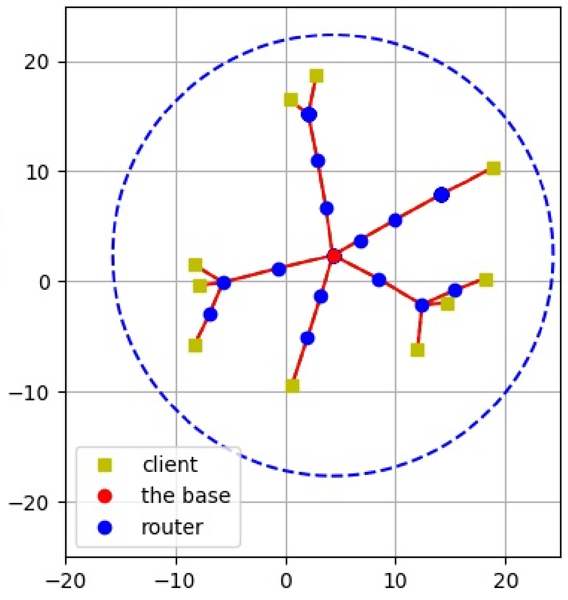


(d) $t = 111.0s$

Figure 3.8: $T(t)$ and $T_E(t)$ during the process of simulation under Algorithm 3

As a future direction, the case where the condition for the existence of a communication link between a pair of agents depends on other factors other than the distance alone will be considered. Moreover, the question of choosing a tree from the induced graph needs further attention.





(e) $t = 195.1s$

Figure 3.8: $T(t)$ and $T_E(t)$ during the process of simulation under Algorithm 3

# Chapter 4

# Experiment with Differential Wheeled Robots

*A multi-agent system with multiple ground robots is used to test Algorithm 1. The experiment set up and motion controller design of the robots are presented. With the results of the experiment, we show that the persistent connectivity of clients is maintained all the time.*

## 4.1 Introduction

EXPERIMENT is set up with multiple e-puck robots to test the effectiveness of the control policy in Chapter 3. First, we describe the motion controller design for one robot by using an existing controller in the literature. Then, we integrate the motion controller in an multi-agent system by modifying the existing controller. Moreover, in this chapter, we prove that persistent connectivity of clients is guaranteed under the settings of our experiment. Finally, the experimental results verify the effectiveness of the control policy in Algorithm 1 with the modified motion controller for routers.

## 4.2 Robot Motion Controller Design

The robot we use for experiment is e-puck robot [6], which is a differential wheeled robot. The e-puck robot is shown in Figure 4.1 [5].

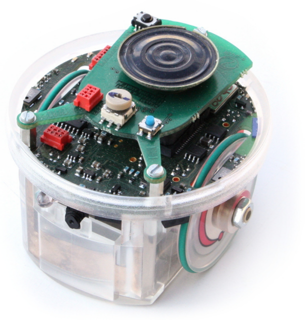


Figure 4.1: E-puck robot version 1

### 4.2.1 Dynamic Model of E-puck Robot

We use $\xi_x, \xi_y$ to denote the Cartesian position of a robot, and $\phi$ to denote its orientation angle, as shown in Figure 4.2. The kinematic model of the differential drive robots is as

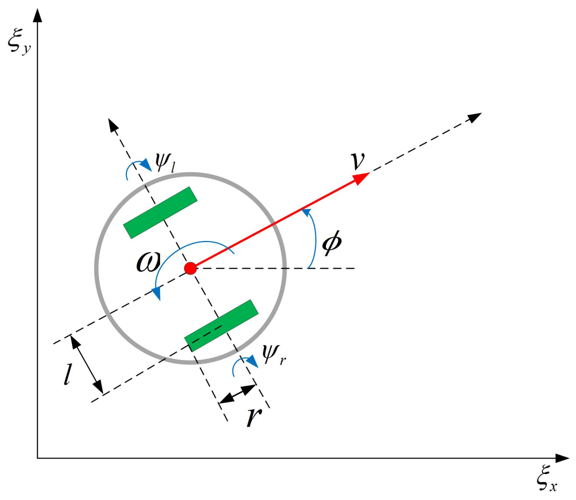


Figure 4.2: Dynamic model of e-puck robot

follows

$$\dot{\xi}_x = v\cos\phi, \tag{4.1a}$$

$$\dot{\xi}_y = v\sin\phi, \tag{4.1b}$$

$$\dot{\phi} = \omega, \tag{4.1c}$$

where $v$ is the linear velocity and $\omega$ is the angular velocity of orientation of the robot. The assumed control inputs $v$ and $\omega$ satisfy the following conditions

$$v = \frac{r\dot{\psi}_r}{2} + \frac{r\dot{\psi}_l}{2}, \tag{4.2a}$$

$$\omega = \frac{r\dot{\psi}_r}{2l} - \frac{r\dot{\psi}_l}{2l}, \tag{4.2b}$$

where $r$ is the radius of the wheels of the robot, $l$ is the distance between two wheels, $\dot{\psi}_r, \dot{\psi}_l$ are the angular velocities of the right and left wheels. Here, $\dot{\psi}_r, \dot{\psi}_i$ are the actual control inputs for the robot. Based on the relationship in (4.2), $v, \omega$ should satisfy

$$\dot{\psi}_r = \frac{v + l\omega}{r} \tag{4.3a}$$

$$\dot{\psi}_l = \frac{v - l\omega}{r}. \tag{4.3b}$$

### 4.2.2 Trajectory Tracking Controller

Assume a reference state trajectory $q_d(t) = [\xi_{xd}(t), \xi_{yd}(t), \phi_d(t)]^\intercal$ is given. The state of an e-puck robot at time instant $t$ is denoted by $q(t) = [\xi_x(t), \xi_y(t), \phi(t)]^\intercal$. Define $\tilde{q}, \tilde{v}, \tilde{\omega}$ as $\tilde{q} := q_d - q$, $\tilde{v} = v_d - v$, $\tilde{\omega} = \omega_d - \omega$. Linearisation of the dynamic model of the robot at $q_d(t)$ leads to

$$\dot{\tilde{q}} = \begin{bmatrix} 0 & 0 & -v_d \sin\phi_d \\ 0 & 0 & v_d \cos\phi_d \\ 0 & 0 & 0 \end{bmatrix} \tilde{q} + \begin{bmatrix} \cos\phi_d & 0 \\ \sin\phi_d & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\omega} \end{bmatrix}. \tag{4.4}$$

The trajectory tracking controller design presented in the following is from [18]. Readers may refer to Section 5 in [18] for more details. Tracking error $\epsilon = [\epsilon_1, \epsilon_2, \epsilon_3]^\top$ is defined as

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{q}. \tag{4.5}$$

The control inputs $v, \omega$ are designed as

$$\begin{aligned} v &= v_d \cos \epsilon_3 + k_1(v_d, \omega_d)\epsilon_1, \\ \omega &= \omega_d + \bar{k}_2 v_d \frac{\sin \epsilon_3}{\epsilon_3} \epsilon_2 + k_3(v_d, \omega_d)\epsilon_3, \end{aligned} \tag{4.6}$$

where $k_1(v_d(t), \omega_d(t)) = k_3(v_d(t), \omega_d(t)) = 2\sigma\sqrt{\omega_d^2(t) + b v_d^2(t)}$, $\bar{k_2} = b$ with $b > 0$ and $\sigma \in (0,1)$. Theorem 4.1 presents the stabilization of $\epsilon$ under the controller presented in 4.6.

**Theorem 4.1.** *[18] Assuming $v_d$ and $\omega_d$ are bounded with bounded derivatives, and that $v_d(t) \nrightarrow 0$ or $\omega_d(t) \nrightarrow 0$ when $t \to \infty$, the control law (4.6) globally asymptotically stabilizes the origin $\epsilon = 0$.*

Note that, with the relationship in (4.5), $\epsilon = 0$ implies $\tilde{q} = 0$. Once $v$ and $\omega$ are calculated, $\dot{\psi}_r, \dot{\psi}_l$ will be determined according to (4.3).

With our sensing system, the values of $\phi$ are in the interval $[-\pi, \pi]$, which will cause discontinuity of $\epsilon_3$ under some conditions. For example, when $\phi = \pi$, and $\omega > 0$, in the next time instant $\phi$ might become $-\pi$, which makes $\epsilon_3$ change discontinuously. To avoid such situations, we redefine $\epsilon_3$ as

$$\epsilon_3(t) = \begin{cases} \phi_d(t) - \phi(t) & \text{if } -\pi \leq \phi_d(t) - \phi(t) \leq \pi \\ \phi_d(t) - \phi(t) + 2\pi & \text{if } \phi_d(t) - \phi(t) < -\pi \\ \phi_d(t) - \phi(t) - 2\pi & \text{if } \phi_d(t) - \phi(t) < \pi \end{cases} \tag{4.7}$$

### 4.2.3 Controller for Steering an E-puck Robot in the Multi-agent System

We use $x$ to denote the planar position of a robot, which is $x := [\xi_x, \xi_y]^\top$. Because we only address one robot in this part, we will omit the indices of agents. When we consider controllers of robots in the multi-agent system settings, we will use indices to differentiate agents. As stated in Subsection 3.4.4, the desired trajectory of a router is a straight line. The trajectory is defined based on the desired position $x^*$, which is determined by Algorithm 2 or 3. Therefore, testing the controller for one robot, we will only consider straight line trajectories. Assume the desired positions of a robot is $x^* = [\xi_x^*, \xi_y^*]^\top$. Here we use $t_a$ to denote the time interval for steering a robot to its desired positions. Then the desired trajectory of planar position is defined as

$$r(t) := x(\bar{t}_0) + \frac{t - \bar{t}_0}{t_a}\big(x^* - x(\bar{t}_0)\big),\ \forall t \in (\bar{t}_0, \bar{t}_0 + t_a], \tag{4.8}$$

where $\bar{t}_0$ denotes the initial time instant of steering. Note that $r(t) = [\xi_{xd}(t), \xi_{yd}(t)]^\top$, $\forall t \in (\bar{t}_0, \bar{t}_0 + t_a]$. Since the desired trajectory $r(t)$ is a straight line, the desired rotation angle $\phi_d(t)$ is a constant such that

$$\phi_d(t) \equiv \arctan \frac{\xi_y^* - \xi_y(\bar{t}_0)}{\xi_x^* - \xi_x(\bar{t}_0)}, \forall t \in (\bar{t}_0, \bar{t}_0 + t_a]. \tag{4.9}$$

Thus, the reference state trajectory of the robot with desired position $x^*$ is set as

$$q_d(t) = \big[\xi_{xd}(t), \xi_{yd}(t), \phi_d(t)\big]^\top = \big[r(t)^\top, \phi_d(t)\big]^\top. \tag{4.10}$$

According to the definition of $e(t)$, we have $e(t) = [\xi_x^i(t) - \xi_{xd}^i(t), \xi_y^i(t) - \xi_{yd}^i(t)]^\top$. Thus the norm of $e(t)$ is $\|e_i(t)\| = \sqrt{(\xi_x(t) - \xi_{xd}(t))^2 + (\xi_y(t) - \xi_{yd}(t))^2}$.

### 4.2.4 Experiment Result of Controller for One Robot

#### Experiment set up for motion control of one robot

The robot we use for experiment is e-puck1 [1]. The radius of the wheel is $r = 0.0205m$, and the distance between two wheels is $2l = 0.053m$. The angular velocities of the wheels satisfy $-2\pi/s \le \dot{\psi}_r \le 2\pi/s, -2\pi/s \le \dot{\psi}_l \le 2\pi/s$. The tracking system we use to capture the position of robots is a camera tracking system OptiTrack Flex13 [4]. The main program to calculate the control inputs to control the robot is written in *python*, which is running on a HP Elitebook 840 laptop with Windows system. In the program, we use the package *optirx* [3] to receive the position of robots from the Motive [2], which is the software environment corresponding to OptiTrack Flex13. The main control program will send the control inputs $\dot{\psi}_l$ and $\dot{\psi}_r$ to the e-puck1 robot. The firmware on the micro-controller of the robot is written based on the standard firmware of e-puck robots.

Readers may refer to [6] for details of the firmware. The communication between the Python program and the e-puck robot is via bluetooth.

Because of the restricted area in our lab, we set $\bar{v}_r$ in Assumption 3.1 as $0.02m/s$, which is slower than the maximal speed $v_r$ the e-puck1 robot can reach. We also set $-1.5rad/s \leq \omega \leq 1.5rad/s$. Note that these constraints on the movement of robots are used in designing the desired trajectories of robots.

**Experimental results for one e-puck robot**

We first test the trajectory tracking controller for a robot with a straight line desired trajectory. As we only use one robot in this test, we still use $x = [\xi_x, \xi_y]^T$ without the index $i$ to denote the position of the robot. Same simplification rules apply for other notations in this test. We use $x_0$ to denote the initial position of the robot, which is $x_0 = [\xi_{x_0}, \xi_{y_0}]^\intercal = [0.35, 0.21]^\intercal$ in our implementation. The destination is chosen as

$$x^* = [\xi_{x_0} + 0.4, \xi_{y_0} + 0.4]^\intercal. \tag{4.11}$$

The desired trajectory is set as $r(t) = x_0 + \frac{t-\bar{t}_0}{30}(x^* - x_0)$, where the time length for the test is set as $30s$. We show how the tracking error $e(t)$ varies with different initial orientation angle $\phi_0$ in Figure 4.3. As shown in this figure, when the difference between $\phi(\bar{t}_0)$ and $\phi_d$ is large, there is a large tracking error $e(t)$ at the start of the movement of the robot. We conclude that the tracking error upper bound $e_r$ increases with the difference between initial orientation angle and the orientation angle of the desired trajectory. Note that at the end of the movement, we can see that the magnitude of the error is still small even with a large difference between $\phi(\bar{t}_0)$ and $\phi_d$.

If we expect a small magnitude of $e_r$, which is presented in (3.36) and (3.39), the magnitude of $e(t)$ is required to be small all the time during the steering. The method we use to deal with this requirement is addressed in the next section.

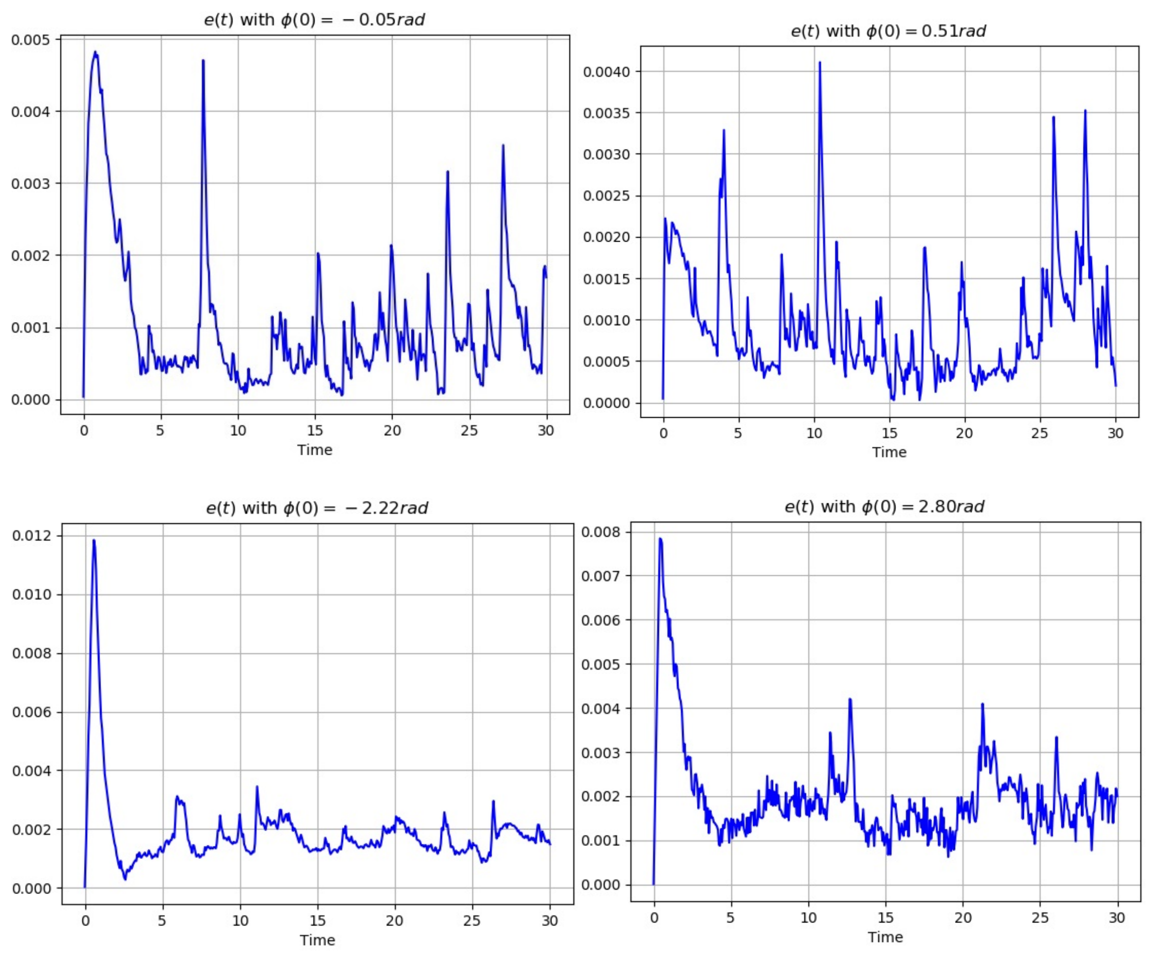


Figure 4.3: $e(t)$ with different $\phi(\bar{t}_0)$ with $\bar{t}_0 = 0$

## 4.3 Modified Trajectory Tracking Controller for an E-puck Robot

As shown in Figure 4.3, in order to decrease the value of $e_r$, we need to make $e(t)$ small at the start of the movement of the robot. To achieve this, we split the steering process into two stages, rotation stage and translation stage. During the rotation stage, the robot rotates to the desired angle which is the angle of the desired straight line trajectory. During the translation stage, the robot will track the straight line trajectory connecting its current position and the desired position. A time interval with $t_p$ seconds is allocated to the rotation stage, and $t_r$ seconds is allocated to the translation stage. The process of steering a router to its corresponding desired position is presented in Algorithm 10.

**Algorithm 10** Process of steering a router to its desired position

```
1: Steer the robot to a desired rotation angle $\arctan \frac{\xi_y^{i*} - \xi_y(t_n)}{\xi_x^{i*} - \xi_x(t_n)}$ with a constant angular velocity $\omega_d$
2: Wait until the current time instant is $\bar{t}_0 + t_p$
3: Steer the robot to the desired position along the trajectory $r_i(t)$
```

In the rotation stage, we set the reference trajectory as follows. First we define $\delta_\phi$ as,

$$\delta_\phi = \begin{cases} \phi_d - \phi(\bar{t}_0), & \text{if } -\pi \le \phi_d - \phi(\bar{t}_0) \le \pi \\ \phi_d - \phi(\bar{t}_0) + 2\pi, & \text{if } \phi_d - \phi(\bar{t}_0) < -\pi \\ \phi_d - \phi(\bar{t}_0) - 2\pi, & \text{if } \phi_d - \phi(\bar{t}_0) > \pi, \end{cases} \tag{4.12}$$

where $\phi_d$ is defined in (4.9). Then we define $\phi_o^i(t)$ as $\phi(t) = \phi(\bar{t}_0) + \frac{\delta_\phi}{|\delta_\phi|}\omega_d(t - \bar{t}_0),\ \bar{t}_0 \le t \le \bar{t}_0 + t_p$, where $\bar{t}_0$ is the initial time instant of movement. In the rotation stage, the desired trajectory of $\phi(t)$ is defined as follows for all $t \in [\bar{t}_0, \bar{t}_0 + t_p]$

$$\phi_d(t) = \begin{cases} \phi_o(t), & \text{if } -\pi \le \phi_o(t) \le \pi \\ \phi_o(t) + 2\pi, & \text{if } \phi_o(t) < -\pi \\ \phi_o(t) - 2\pi, & \text{if } \phi_o(t) > \pi. \end{cases} \tag{4.13}$$

The desired trajectory of the other two states are set as $\xi_{xd}(t) = \xi_x(\bar{t}_0)$, $\xi_{yd}(t) = \xi_y(\bar{t}_0)$. Thus the desired planar trajectory during this time interval is $r(t) = [\xi_x(\bar{t}_0), \xi_y(\bar{t}_0)]^\top,\ t \in [\bar{t}_0, \bar{t}_0 + t_p]$. Then the desired trajectory $q_d(t)$ during the rotation stage is set as

$$q_d(t) = [\xi_x(\bar{t}_0), \xi_y(\bar{t}_0), \phi_d(t)]^T,\ t \in [\bar{t}_0, \bar{t}_0 + t_p]. \tag{4.14}$$

During the translation stage, the desired planar position trajectory $r(t)$ is set in the way as in (4.8) as follows

$$r(t) = x(\bar{t}_0) + \frac{t - \bar{t}_0 - t_p}{t_a}\big(x^* - x(\bar{t}_0)\big),\ t \in (\bar{t}_0 + t_p, \bar{t}_0 + t_p + t_m]. \tag{4.15}$$

Moreover, the desired rotation angle is $\phi_d$, which is defined in (4.9). Then the desired

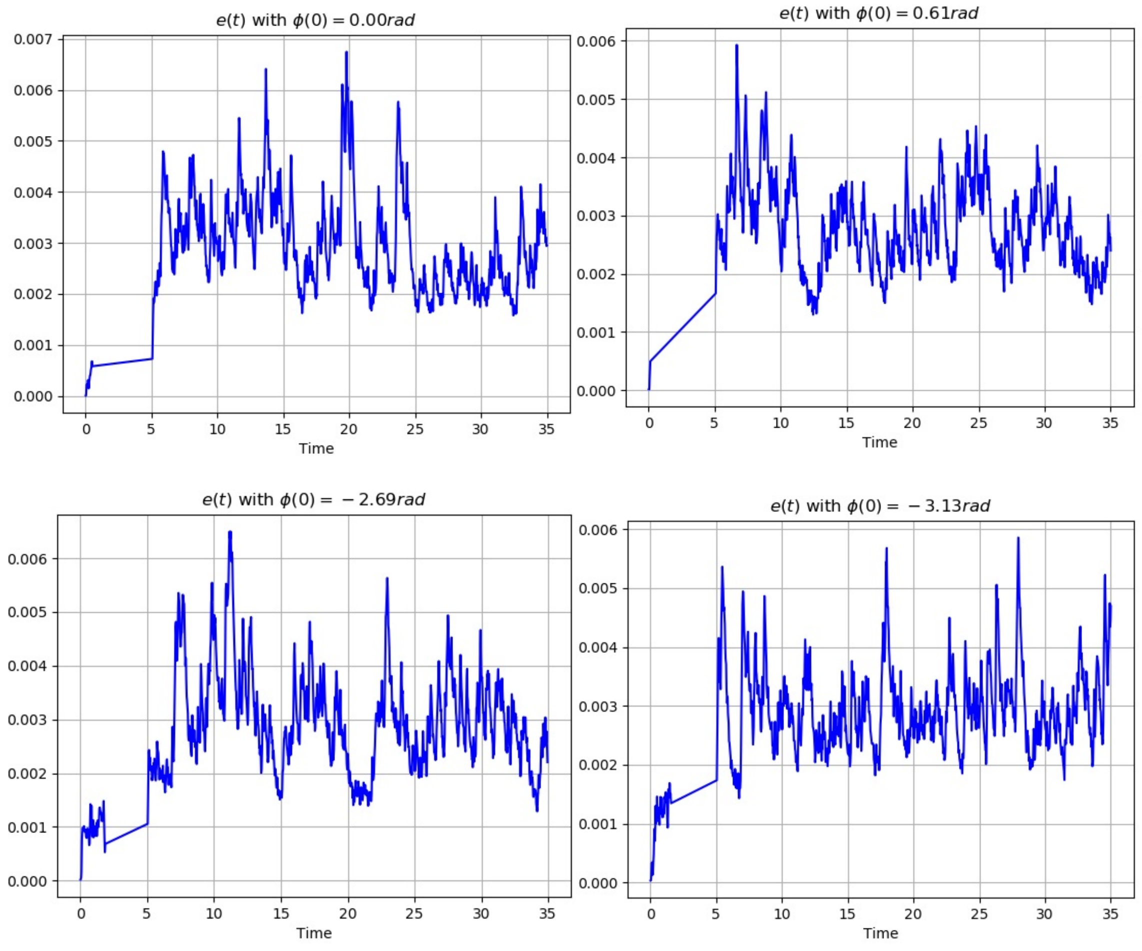


Figure 4.4: $e(t)$ with different $\phi(\bar{t}_0)$ with $\bar{t}_0 = 0$

trajectory $q_d(t)$ during the translation stage is

$$q_d(t) = [r(t)^\top, \phi_d]^\top, \ t \in (\bar{t}_0 + t_p, \bar{t}_0 + t_p + t_m]. \tag{4.16}$$

Next, we present experimental results for one robot with different initial orientation angles by using the modified motion controller. In Figure 4.4, $e(t)$ is shown with this rotation and tracking control strategy. The destination is still set the same as in (4.11). Moreover, the desired trajectory is set the same as in Subsection 4.2.4. It shows that the magnitude upper bounds of $e(t)$ are almost the same for different $\phi(\bar{t}_0)$. Furthermore, we can conclude the upper bound is smaller by using this control strategy. Thus, in the implementation of our control strategy in Algorithm 1, we will choose this rotation-tracking movement strategy to steer the rotors when the steering is in time interval $t_m$. From multiple times of trials, we set the upper bound of the magnitude of $e(t)$ as $0.01m$ with the controller inputs determined by (4.3) and (4.6). That is, $\|e(t)\| \leq 0.01m$, $\forall t \in [\bar{t}_0, \bar{t}_0 + t_p + t_m]$.

## 4.4 Modified Routers' Desired Trajectories and Control Requirements

With the modified controller design in Section 4.3, the desired trajectories and control requirements for routers in multi-agent systems are presented. In this section, we use superscript or subscript $i$ to denote the variables corresponding to router $i$. Previous steering of routers is split into two stages as presented in Section 4.3. The first one is preparation stage, which costs $t_p$ seconds, during which the robots rotate to their corresponding desired orientation angles. The other stage is the translation stage, during which the robots follow the desired straight line trajectories. In this section, we implement the modified motion controller for routers in the multi-agent system considered in Chapter 3.

First, we consider implementing the modified motion controller for routers in the time interval $t_m$ (Figures 3.2 and 3.3). We appoint the preparation stage to the time interval $t_d$ (Figures 3.2 and 3.3) right before the time interval $t_m$. Translation stage is still appointed in the time interval $t_m$. Now there are two tasks to be completed in the time interval $t_d$. The first one is to determine the desired positions of inactive routers during time interval $t_r$. The second task is to rotate the routers to the desired orientation angles determined by the straight line desired trajectories during time interval $t_p$. The time interval $t_d$ is split as in Figure 4.5.

Now, the start time instant of movement for our multi-agent system is $t_n - t_p$, where $t_n$ is the start time of the time interval $t_m$. The desired orientation angle is

$$\phi_d^i = \arctan \frac{\xi_x^{i*} - \xi_x^i(t_n - t_p)}{\xi_y^{i*} - \xi_y^i(t_n - t_p)}, \tag{4.17}$$

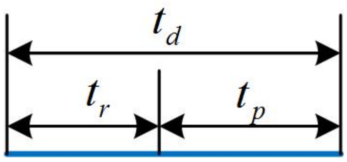


Figure 4.5: Split of $t_d$

where $\left[\xi_x^{i*}, \xi_y^{i*}\right]^\intercal = x_i^*$. Then $\delta_\phi^i$ is defined as

$$\delta_\phi^i = \begin{cases} \phi_d^i - \phi^i(t_n - t_p),\ if\ -\pi \le \phi_d^i - \phi_i(t_n - t_p) \le \pi \\ \phi_d^i - \phi^i(t_n - t_p) + 2\pi,\ if\ \phi_d^i - \phi_i(t_n - t_p) < -\pi\ , \\ \phi_d^i - \phi^i(t_n - t_p) - 2\pi,\ if\ \phi_d^i - \phi_i(t_n - t_p) > -\pi \end{cases} \tag{4.18}$$

and $\phi_o^i(t) := \phi^i(t_n - t_p) + \frac{\delta_\phi^i}{|\delta_\phi^i|}\omega_d(t - t_n + t_p)$. Thus, during the rotation stage, the desired trajectory of orientation angle is defined as in (4.13)

$$\phi_r^i(t) = \begin{cases} \phi_o^i(t), & \text{if } -\pi \le \phi_o^i(t) \le \pi, \\ \phi_o^i(t) + 2\pi, & \text{if } \phi_o^i(t) < -\pi, \qquad t \in (t_n - t_p, t_n] \\ \phi_o^i(t) - 2\pi, & \text{if } \phi_o^i(t) > \pi. \end{cases} \tag{4.19}$$

The desired trajectory of planar position during the rotation stage is

$$r_i(t) := [\xi_x^i(t_n - t_p), \xi_y^i(t_n - t_p)]^\top,\ t \in (t_n - t_p, t_n]. \tag{4.20}$$

Thus, the desired trajectory $q_d(t)$ during this stage is,

$$q_d^i(t) = [\xi_{xd}^i(t), \xi_{yd}^i(t), \phi_r^i(t)]^\top = [\xi_x^i(t_n - t_p), \xi_y^i(t_n - t_p), \phi_r^i(t)]^\top,\ t \in (t_n - t_p, t_n] \tag{4.21}$$

In the translation stage, the desired orientation angle is $\phi_d^i$, which is defined in (4.17). The desired trajectory of the planar position of a router $i$ is

$$r_i(t) := x_i(t_n - t_p) + \frac{t - t_n}{t_m}\left(x_i^* - x_i(t_n - t_p)\right),\ t \in (t_n, t_n + t_m]. \tag{4.22}$$

Therefore, the desired trajectory used for controller design of robot $i$ is set as,

$$q_d^i(t) = [\xi_{xd}^i(t), \xi_{yd}^i, \phi_d^i]^\top = [r_i(t)^\top, \phi_d^i]^\top,\ t \in (t_n, t_n + t_m]. \tag{4.23}$$

With the modified desired trajectories of routers above, the tracking error $e_i(t)$ during the movement satisfies

$$|e_i(t)| = \|x_i(t) - r_i(t)\| \leq e_r,\ t \in (t_n - t_p, t_n + t_m]. \tag{4.24}$$

We deal with the inactive routers when $\|x_i^* - x_i(t_n)\| > t_m v_r$ in the same as the original motion control design.

Then, we consider implementing the modified controller for routers in time intervals $t_u$ and $t_c$ (Figures 3.2 and 3.3). In these situations, we appoint preparation stage to time interval $(t_s, t_s + t_p]$ and translation stage to time interval $\left(t_s, t_s + \frac{\|x_i^* - x_i(t_s)\|}{\bar{v}_r}\right]$. The desired trajectory in the preparation stage is the same as the trajectory in time interval $(t_n - t_p, t_n]$ above. The desired trajectory in the translation stage is the same as the trajectory in (3.38). As the method of deriving the desired trajectory is in the same way as in time interval $(t_n - t_p, t_n + t_m]$ above, we will not give details here.

By using this modified controller design, we will show next that the persistent connectivity of clients can be maintained.

### 4.4.1 Persistent Connectivity of Clients with Modified Desired Trajectories

In this subsection, we will prove that the persistent connectivity of clients still holds with the modified controllers design. We give the details that the conclusion in Lemma 3.1 still hold with the modified controllers design for routers. The other conclusions in Lemmas 3.2–3.4 and Theorem 3.1 can be proved in the same way, we will not give details. During every optimization and control stage (lines $10 - 13$, $35 - 38$, Algorithm 1), Lemma 3.1 presents that the corresponding selected tree is always connected.

**Lemma 4.1.** *Suppose Assumptions 3.1, 1.1, 1.3, 3.3, and 3.6 hold and let the motion of routers satisfy the requirement in (4.24). Let $\delta$ and $\delta_G$ be defined in (3.40) and (3.41), respectively. Then,*

$\|x_i(t) - x_j(t)\| \leq R,\ \forall\{i,j\} \in \mathcal{E}_T^\kappa,\ \forall t \in \big(t_0^\kappa + n\Delta, t_0^\kappa + (n+1)\Delta\big],\ \forall\kappa = 0,1,\ldots,\ \forall n = 0,\ldots,n^\kappa - 1$.

*Proof.* In this proof, we will use $x_i^*(t_0^\kappa + n\Delta)$ to denote the desired planar position of the active router $i$ calculated during time interval $(t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ and $x_b^*(t_0^\kappa + n\Delta)$ to denote $x_c(t_0^\kappa + n\Delta)$.

*Case 1*: First, we consider the edges $\{i,j\}$ connecting a router and a client, $i \in \mathcal{R}, j \in \mathcal{C}$.

Consider $n = 0$, with Assumptions 3.3 and 3.6, we have

$$\|x_i(t) - x_j(t)\| \leq R - (t_x + t_d)v_c - e_r + (t_x + t_d)v_c + e_r = R,\ \forall t \in (t_0^\kappa, t_0^\kappa + t_x + t_d].$$

Especially, for all $t \in (t_0^\kappa, t_0^\kappa + t_x + t_r]$, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta + e_r + (\Delta + t_x + t_r)v_c \leq R - t_q v_c - e_r \tag{4.25}$$

The above inequalities hold because the routers stay static and only clients move during these time intervals.

Moreover, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ where $n > 0$, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta + e_r + (\Delta + t_x + t_d)v_c + e_r \leq R.$$

Especially, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_r]$, we still have

$$\|x_i(t) - x_j(t)\| \leq R - \delta + e_r + (\Delta + t_x + t_r)v_c \leq R - t_q v_c - e_r \tag{4.26}$$

Let $x_i^*(t_0^\kappa + n\Delta)$ be a solution to the optimization problem considered in time interval $(t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x]$. With the modified controllers design for routers, we have

$$\|x_i^*(t_0^\kappa + n\Delta) - x_j(t_0^\kappa + n\Delta + t_x + t_r)\| \leq R - \delta + (t_x + t_r)v_c \leq R - (\Delta + t_q)v_c - 2e_r.$$

During any time interval $(t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, with the trajectory $r_i(t)$ in (4.22) and control requirement in (4.24), the length of edge $\{i,j\}$ satisfies the following condition for all $t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$.

$$\begin{aligned}
\|x_i(t) - x_j(t)\| =& \left\| x_i(t_0^\kappa + n\Delta + t_x + t_d) + \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m}(x_i^*(t_0^\kappa + n\Delta) - x_i(t_0^\kappa + n\Delta)) \right. \\
& \left. + e_i(t) - x_j(t) \right\| \\
=& \left\| (1-\alpha)(x_i(t_0^\kappa + n\Delta + t_x + t_d) - x_j(t)) + \alpha(x_i^*(t_0^\kappa + n\Delta) - x_j(t)) + e_i(t) \right\| \\
=& \left\| (1-\alpha)(x_i(t_0^\kappa + n\Delta + t_x + t_d) - x_j(t)) \right\| + \left\| \alpha(x_i^*(t_0^\kappa + n\Delta) - x_j(t)) \right\| \\
& + \left\| e_i(t) \right\| \\
\leq& (1-\alpha)(R - t_q v_c - e_r + t_p v_c + \alpha t_m v_c) + \alpha\big(R - (\Delta + t_q)v_c - 2e_r + t_q v_c \\
& + \alpha t_m v_c\big) + e_r \\
\leq& (1-\alpha)(R - e_r + \alpha t_m v_c) + \alpha(R - \Delta v_c - 2e_r + \alpha t_m v_c) + e_r \\
=& R + \alpha(t_m v_c - \Delta v_c - e_r) \\
\leq& R
\end{aligned},$$

where $\alpha = \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m}$ and $0 \leq \alpha \leq 1$. The largest value on the right hand side of this inequality is achieved when $\alpha = 0$.

Therefore, $\|x_i(t) - x_j(t)\| \leq R$, $\forall t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, $\forall \{i,j\} \in \big\{\{i,j\} | i \in \mathcal{R}, j \in \mathcal{C}, \{i,j\} \in \mathcal{E}_{T^\kappa}\big\}$, $\forall \kappa = 0, 1, \ldots$, $\forall n = 0, \ldots, n^\kappa - 1$.

*Case 2*: Now, we consider the edges $\{i,j\}$ with $i,j \in \mathcal{R}$. Consider $n = 0$, with Assumptions 3.3 and 3.6, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta_G + 2e_r \leq R, \ \forall t \in (t_0^\kappa, t_0^\kappa + t_x + t_d]. \tag{4.27}$$

Especially, for all $t \in (t_0^\kappa, t_0^\kappa + t_x + t_r]$, because the routers do not move, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta_G \leq R - 2e_r. \tag{4.28}$$

Moreover, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$ with $n > 0$, we have

$$\|x_i(t) - x_j(t)\| \leq R - \delta + 2e_r + 2e_r \leq R. \tag{4.29}$$

Especially, for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_r]$, we still have

$$\|x_i(t) - x_j(t)\| \leq R - \delta_G \leq R - 2e_r. \tag{4.30}$$

The desired positions of routers calculated by the optimization problem satisfy the following conditions for any $\kappa$, $n$

$$\|x_i^*(t_0^\kappa + n\Delta) - x_j^*(t_0^\kappa + n\Delta)\| \leq R - \delta \leq R - 4e_r.$$

During any time interval $(t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta]$, with the trajectory $r_i(t)$ in (4.22) and controller requirement in (4.24), the length of edge $\{i,j\}$ satisfies the following condition for all $t \in (t_0^\kappa + n\Delta, t_0^\kappa + n\Delta + t_x + t_d]$

$$\begin{aligned}
\|x_i(t) - x_j(t)\| =& \Bigg\| \Big[ x_i(t_0^\kappa + n\Delta + t_x + t_r) + \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m} \Big( x_i^*(t_0^\kappa + n\Delta) \\
& - x_i(t_0^\kappa + n\Delta + t_x + t_r) \Big) \Big] - \Big[ x_j(t_0^\kappa + n\Delta + t_x + t_r) \\
& + \frac{t - t_0^\kappa - n\Delta - t_x - t_d}{t_m} \Big( x_j^*(t_0^\kappa + n\Delta) - x_j(t_0^\kappa + n\Delta + t_x + t_r) \Big) \Big] \Bigg\| \\
=& \| [x_i(n\Delta + t_x + t_r) + \alpha(x_i^*(n\Delta) - x_i(n\Delta + t_x + t_r)) + e_i(t)] \\
& - [x_j(n\Delta + t_x + t_r) + \alpha(x_j^*(n\Delta) - x_j(n\Delta + t_x + t_r)) + e_j(t)] \| \\
\leq& (1-\alpha)\|x_i(n\Delta + t_x + t_r) - x_j(n\Delta + t_x + t_r)\| \\
& + \alpha\|x_i^*(n\Delta) - x_j^*(n\Delta)\| + \|e_i(t)\| + \|e_j(t)\| \\
\leq& (1-\alpha)(R - 2e_r) + \alpha(R - 4e_r) + 2e_r \\
\leq& R
\end{aligned}.$$

Therefore, $\|x_i(t) - x_j(t)\| \leq R, \forall t \in (t_0^\kappa + n\Delta + t_x + t_d, t_0^\kappa + (n+1)\Delta], \forall \{i,j\} \in \big\{\{i,j\} | i \in \mathcal{R}, j \in \mathcal{R}, \{i,j\} \in \mathcal{E}_{T^\kappa}\big\}, \forall \kappa = 0, 1, \ldots, \forall n = 0, \ldots, n^\kappa$.

With analysis of *Case 1* and *Case 2*, we conclude that $\|x_i(t) - x_j(t)\| \leq R$, $\forall \{i,j\} \in \mathcal{E}_T^\kappa$, $\forall t \in (t_0^\kappa + n\Delta, t_0^\kappa + (n+1)\Delta]$, $\forall \kappa = 0, 1, \ldots$, $\forall n = 0, \ldots, n^\kappa - 1$. $\square$

Note that in time intervals $t_s, t_c$, it is easy to check that the modified motion controller for routers does not affect the results in Lemmas 3.2 and 3.3. Then, Lemma 3.4 still holds.

Therefore, Theorem 3.1 follows from these lemmas.

### 4.4.2 Experiment Setup

Limited by the area in our lab, we choose $R = 0.9m$, $\bar{v}_r = 0.02m/s$, $v_c = 0.002m/s$. The time intervals are chosen as follows, $t_x = 25s$, $t_d = 5s$, $t_m = 10s$, $t_r = 0s$, $t_p = 5s$, $\Delta = 40s$. With these choices and $e_r = 0.01m$, we calculate that $\delta = 0.16m$, $\delta_G = 0.07m$. The set of indices of agents is $\mathcal{V} = \{1,2,3,4,5,6\}$, the set of indices of routers is $\mathcal{R} = \{1,2,3\}$ and the set of indices of clients is $\mathcal{C} = \{4,5,6\}$. Thus $N = 6$, $N_r = 3$ and $N_c = 3$. Limited by the number of e-puck robots we have, in the experiment, clients are paper boards with markers on them for position tracking while routers are e-puck robots. Same as the settings for one robot control, OptiTrack Flex13 [4] camera system is used to track the positions of agents, and a HP Elitebook 840 laptop running Windows is used to support the experiment.

The wireless communication on e-puck1 robots is bluetooth 2.0. The bluetooth chip on e-puck1 cannot support a real ad-hoc communication which supports communication between robots. Therefore, the communication between two robots need to be handled through a computer. Another fact is, the speed of this type of wireless communication is slow. Considering these factors, we simulate the distributed algorithms, especially the optimization algorithms, on a computer by using *python*. We implement the algorithms in a distributed fashion by using class objects in python. Each class object is used to emulate a micro-controller on a robot and they exchange local information through interfaces during the execution of the algorithms. Therefore, it can still test that our distributed algorithms. When we have new platform that can support ad-hoc communication, such as, wireless communication with zigbee protocol, the algorithms will be applied distributedly on the robots. At this stage, we simulate the exchange of information of robots with python class objects. Similarly, the control inputs are also calculated distributedly with python on a computer. After the control inputs are calculated, they will be sent to corresponding robots. The structure of our experiment setting is illustrated in Figure 4.6.

*Remark:* In our experiment, as stated before, we do not consider the collision between

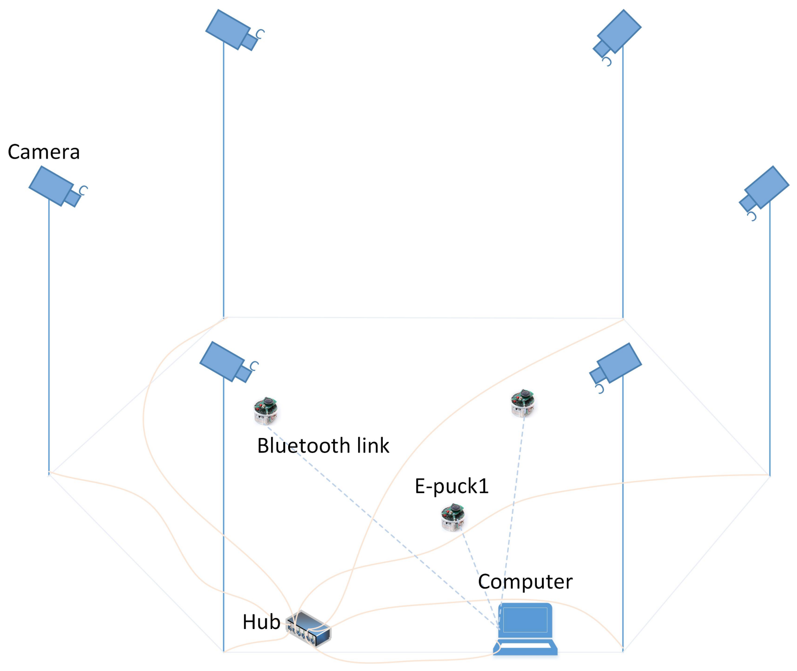


Figure 4.6: Experiment setting in the lab

robots. However, because the area in the lab is limited, if the inactive routers are steered to the same desired position as their neighbours on the extended tree, it will cause collision. To avoid this collision situation, we will not steer an inactive robot if the distance between its position and the neighbour's position is less than $0.035m$. This distance is selected according to the size of the e-puck1 robot we use. In future works, when there are many more robots in the system, we need to consider the collision between robots. Note that this treatment of avoiding collision will not change the connectivity of inactive routers to their corresponding neighbours on the extended tree.

First, we show some results of our control policy (Algorithm 1) with primal subgradient method (Algorithm 2). The result of the length of the longest edge on the selected tree is shown in Figure 4.7. In this figure, we can see that the length of the longest edge on the selected tree is always less than $R$, which means the persistent connectivity of the clients is maintained all the time.

In Figure 4.8, the structure of the selected trees and extended trees at different time instants are shown. This figure shows the update progress of the selected tree. The two axes are the coordinate axes of agents with meter as unit. The blue dashed lines are the

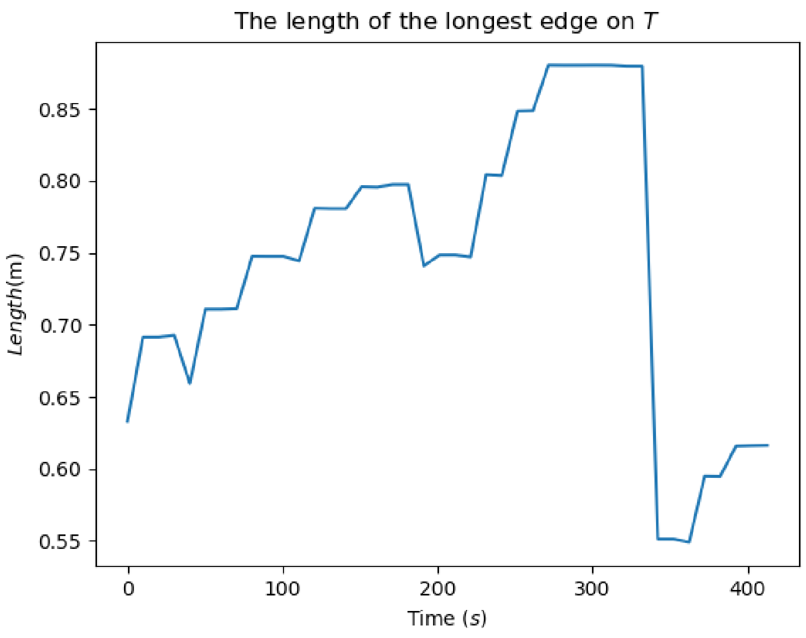
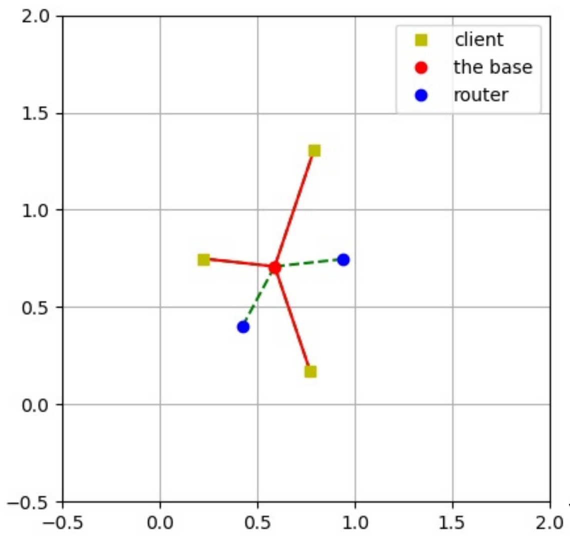


Figure 4.7: The length of the longest edge on $T$

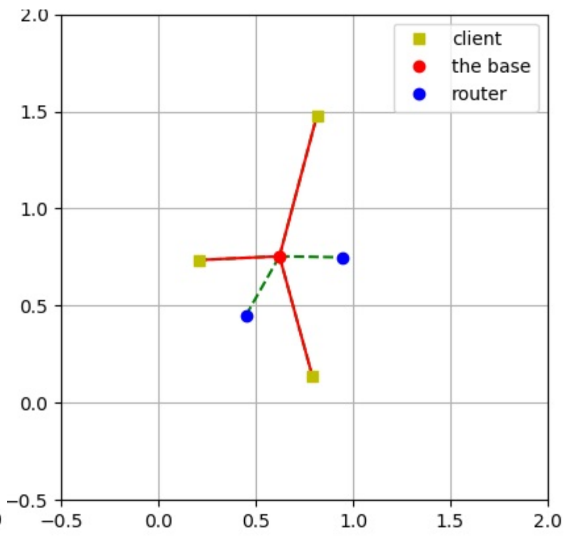
edges of extended trees, and the red lines are the edges of selected trees.

Next, we show some results of the control policy (Algorithm 1) with dual sub-gradient method (Algorithm 3). The result of the length of the longest edge on the selected tree is shown in Figure 4.9. In this figure, we can see that the length of the longest edge on the selected tree is always less than $R$, which means the persistent connectivity of the clients is maintained all the time.

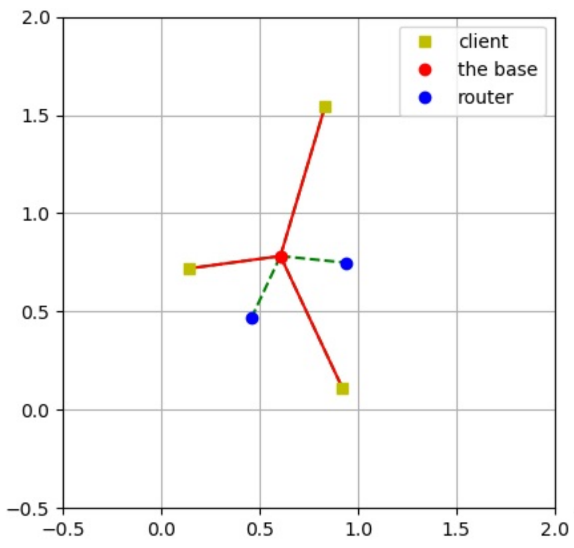
In Figure 4.10, the structure of the selected trees and extended trees at different time instants are shown. This figure shows the update progress of the selected tree. These two experiments confirm that the control policy in Chapter 3 (Algorithm 1) ensures persistent connectivity of clients with the modified motion controller for routers in this chapter.



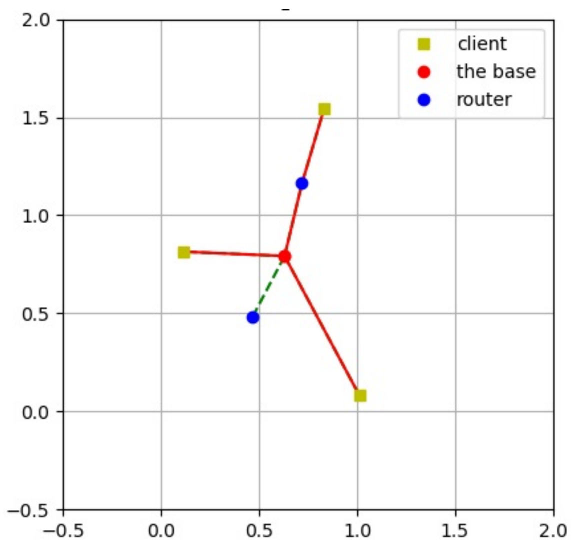
(a) $t = 0.0s$



(b) $t = 80.4s$

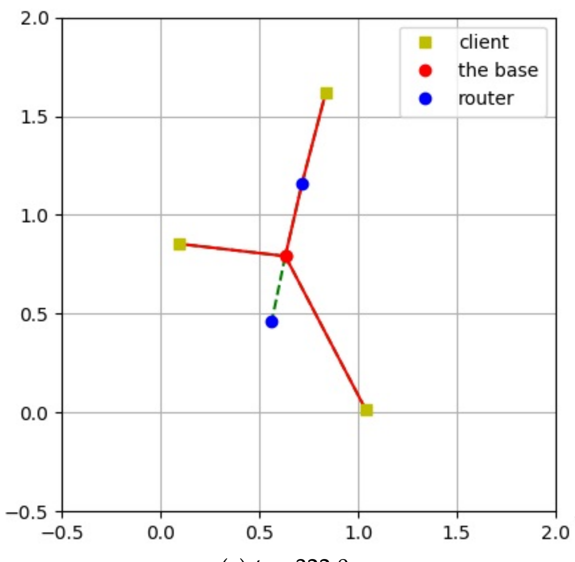


(c) $t = 161.0s$

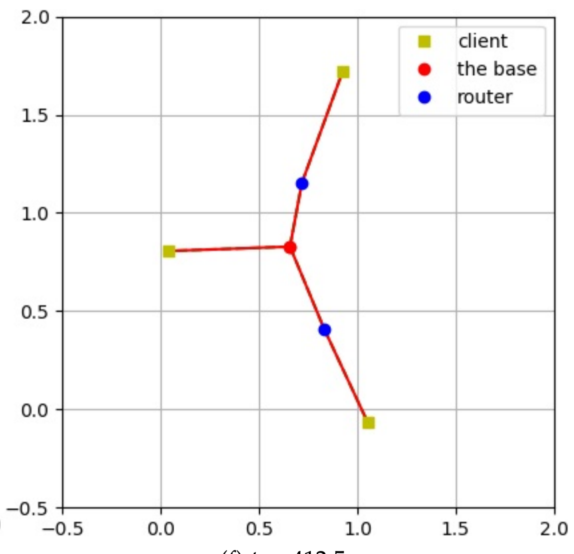


(d) $t = 241.3s$

Figure 4.8: Selected tree and extended tree



(e) $t = 322.0s$



(f) $t = 412.5s$

Figure 4.8: Selected tree and extended tree

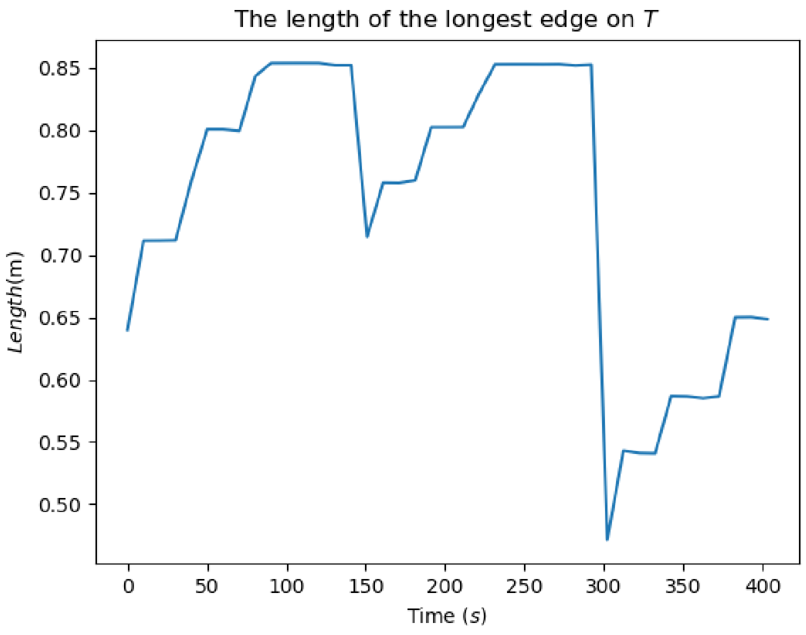
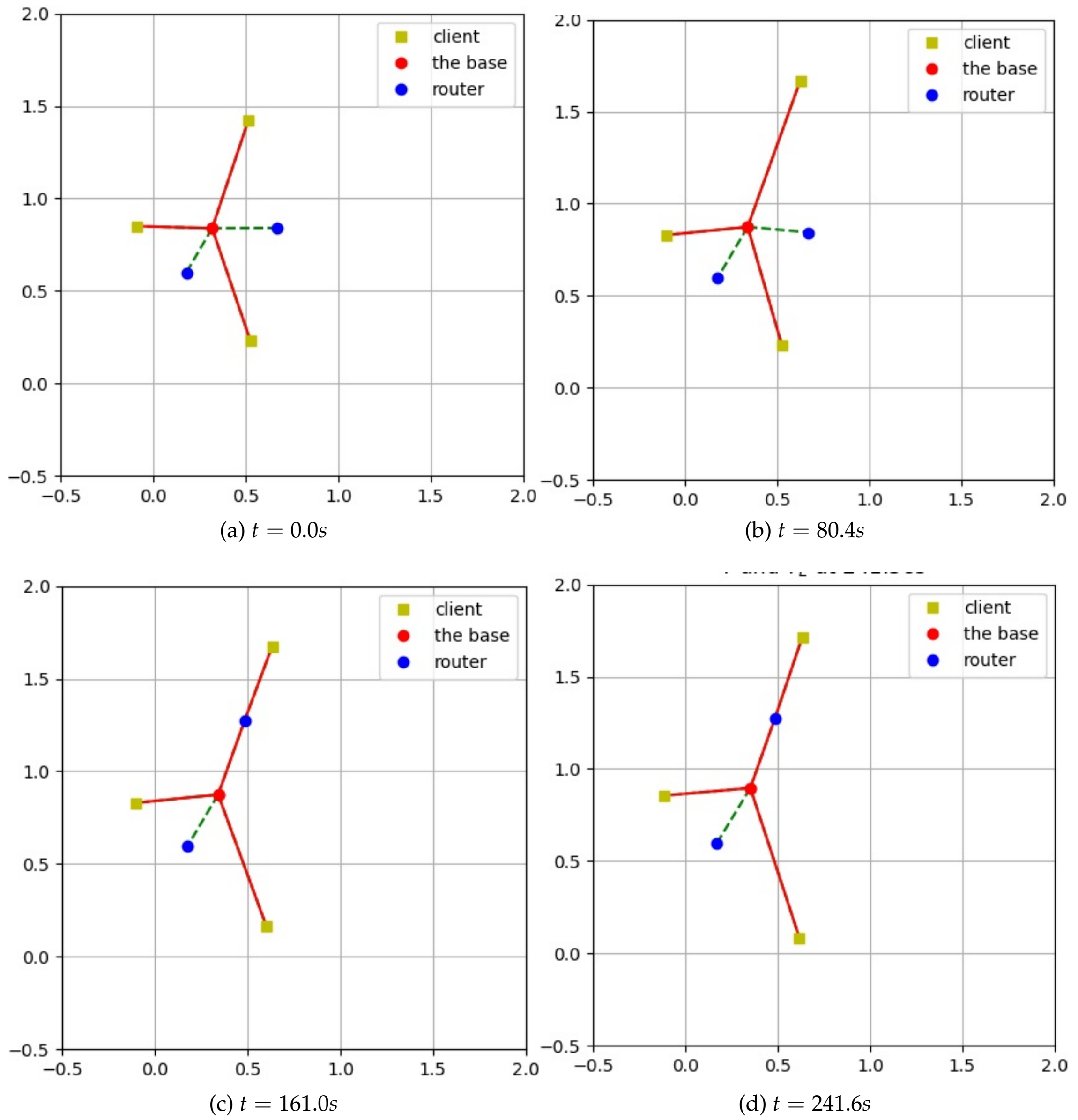


Figure 4.9: The length of the longest edge on $T$



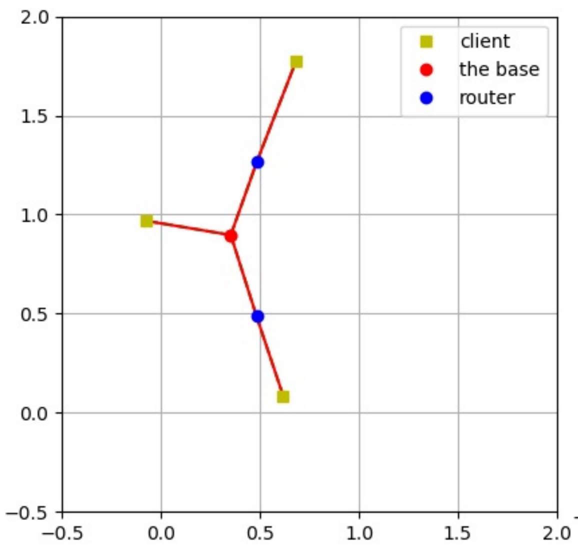
(a) $t = 0.0s$

(b) $t = 80.4s$

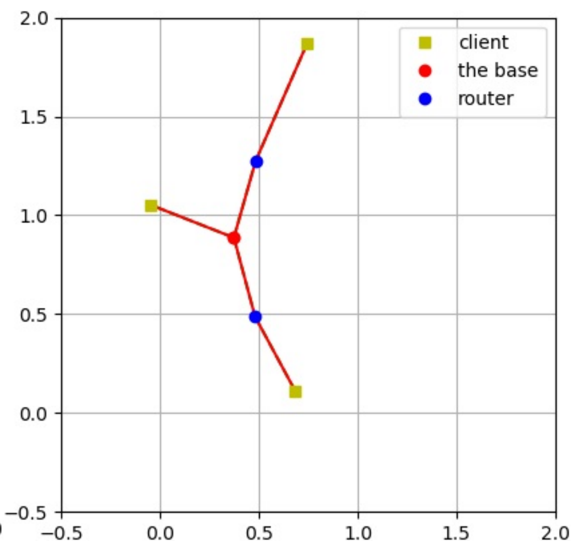
(c) $t = 161.0s$

(d) $t = 241.6s$

Figure 4.10: Selected tree and extended tree



(e) $t = 322.2s$



(f) $t = 402.9s$

Figure 4.10: Selected tree and extended tree

# Chapter 5
# Conclusions

*In this thesis, we propose control policies of routers to maintain persistent connectivity of clients while the clients have their own corresponding objectives. These control policies are suitable for situations where client tasks have higher priority. For example, rescue after disasters and tracking fast moving targets. We address the persistent connectivity of clients in multi-agent systems in two cases. One case is simple with two clients, the other is with an arbitrary number of clients. Moreover, we consider general dynamic models of agents, which makes our control policies fit most of the common types of robots used in multi-agent systems.*

## 5.1 Conclusions

There are three main contributions in this thesis, which are persistent connectivity of clients with different numbers of clients in two cases and the experiment test of the corresponding results.

First, in Chapter 2, we consider a type of multi-agent systems with two clients. This makes the structure of the system simple, that is, the corresponding graph structure is simple. With this structure, we do not need to change the structure of the graph. We also consider a specific dynamic model of the agents in this chapter, which is a quadrotor model. Unlike the simple general model used in most of existing literature, this practical model makes our control strategy more practical when we consider real applications. Furthermore, as mentioned in this chapter, the method of designing control policy for this specific dynamic agent model can be adapted to other robot models by using the same idea. As presented in Chapter 2, we prove that the initial structure of the multi-agent system is always maintained aided by an auxiliary system. That is, the persistent connectivity of the two clients is ensured. Finally, we test the persistent connectivity results in the simulation by using practical parameters of the quadrotor model.

In Chapter 3, a general case of multi-agent systems with multiple clients is considered. This situation leads to a more complex structure, which makes the control policy design more complicated. In this case, we consider the system and control policy in a hybrid form instead of the continuous one in Chapter 2. The control policy to address the persistent connectivity of clients in this case switches between optimization and control. First, optimization algorithms are executed to derive the desired positions of routers based on the current positions of clients. Then, the routers are steered to their corresponding desired positions by using controllers satisfying some specific requirements. If the structure of the corresponding graph cannot support connectivity of clients, the graph will be updated by some spare routers. Because the execution of optimization algorithms and update of the graph structure cost computation time, it is more convenient to address the problem in a hybrid form. At the end of this chapter, we still use a quadrotor model as the dynamic model of agents in the simulation to verify the designed control policy.

In Chapter 4, we use e-puck robots as practical routers to implement our control policy in Chapter 3 and test the results of Chapter 4. In the setting of the experiment, we use a camera system to capture the positions of agents and use the python program on a computer to simulate the computation of agents and communication between robots. This restrictive setting is because the bluetooth 2.0 communication on e-puck robots is slow and the number of e-puck robots we have is limited. As shown in the experiment results, the lab experiment verifies the effectiveness of the control policy we design.

## 5.2 Future Work

There are several future research directions that follow from this work. These directions include theoretical and experimental parts, as stated as follows.

When considering the general case in Chapter 3, we assume that the routers are sufficiently faster than the clients, which limits the capability of clients in performing their main tasks. Moreover, as in the experiment in Chapter 4, the setting that $\bar{v}_r$ is much larger than $v_c$ reflects this limitation in applications. Increasing the speed of clients is very important for a range of applications. For example, in searching and rescue, faster speed means less time to complete the tasks. Therefore, relaxing such an assumption, such as Assumption 3.1, is a direction to improve the applicability of our control policy.

As in Assumption 1.1, the communication model we use in this chapter is only based on the distances between agents. However, the practical wireless communication is also affected by other factors, such as shadowing, fading and routing protocols. In order to make control policies more applicable, we may need to use a more practical communication model when considering communication connectivity in multi-agent systems. Moreover, we might need to have a new definition of persistent connectivity according to improved communication models. This direction requires that we combine the theory in control and system field with the theory in wireless communication field.

Another aspect we may need to explore deeply is how to make use of the structure of the graph corresponding to a multi-agent system. In our control policy in Chapter 3, no criterion is given to choose the selected tree. Future work in this direction may include defining and finding a good tree. We might combine the graph structure with real communication protocols. For example, when we consider the routing protocol, we might choose a more complex structure than a tree from the graph in order to make the communication reliable when considering real wireless communication protocols.

The optimization algorithms, especially Algorithm 3, need synchronous iterations among agents during their execution. These might be improved to be asynchronous algorithms which might be more suitable for multi-agent systems without a centre. In multi-agent systems, the communication and computation might not occur in a synchronous fashion. For example, there might exist delays in communication. Thus, asynchronous algorithms might need less time than synchronous ones which need some time to synchronize information. To explore in this direction, we need to address the convergence of distributed optimization algorithms in asynchronous fashion.

We may also want to improve the experiment platform. Because the communication between agents is simulated on the computer as stated in Chapter 4, we might need to choose a platform that can support ad-hoc wireless communication between robots. When using a real wireless communication, such as zigbee, we might need to schedule the information exchange between agents in experiments. This, in turn, might help improve the theoretical control policy design for practical situations.

In conclusion, there still exist interesting directions in communication connectivity of multi-agent systems and they deserve more efforts.

# Bibliography

[1] "e-puck education robot," http://http://www.e-puck.org/.

[2] "Motive 1.5.7," http://optitrack.com/products/motive/.

[3] "optirx," https://pypi.org/project/optirx/.

[4] "OptiTrack Flex 13," http://tracklab.com.au/products/hardware/optitrack-flex-13/.

[5] "Photo of e-puck," https://en.wikipedia.org/wiki/E-puck_mobile_robot.

[6] "Standard firmware of e-puck," http://www.gctronic.com/doc/index.php/E-Puck#Standard_firmware.

[7] A. B. Alcherio Martinoli, "Underwater robotics and water quality," https://transport.epfl.ch/underwater-robotics-water-quality.

[8] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[9] M. Bangura, R. Mahony *et al.*, "Real-time model predictive control for quadrotors," 2014.

[10] D. P. Bertsekas, *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[11] B. Bethke, J. Redding, J. P. How, M. A. Vavrina, and J. Vian, "Agent capability in persistent mission planning using approximate dynamic programming," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 1623–1628.

[12] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[13] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms.* Princeton University Press, 2009.

[14] Y. Cao and W. Ren, “Distributed coordinated tracking via a variable structure approach-part i: Consensus tracking,” in *American Control Conference (ACC), 2010.* IEEE, 2010, pp. 4744–4749.

[15] J. Chen and Q. Zhu, “Resilient and decentralized control of multi-level cooperative robotic networks to maintain connectivity under adversarial attacks,” *arXiv preprint arXiv: 1505.07158*, 2015.

[16] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, “Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft,” *Journal of Field Robotics*, vol. 30, no. 3, pp. 415–438, 2013.

[17] E. Commission, “Robot fleets for highly effective agriculture and forestry management,” http://www.rhea-project.eu/index.php.

[18] A. De Luca, G. Oriolo, and M. Vendittelli, “Control of wheeled mobile robots: An experimental overview,” *Ramsete*, pp. 181–226, 2001.

[19] R. Delpoux and T. Floquet, “On-line parameter estimation via algebraic method: An experimental illustration,” *Asian Journal of Control*, vol. 17, no. 1, pp. 315–326, 2015.

[20] F. Ducatelle, G. A. Di Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, F. Mondada, R. OGrady, C. Pinciroli, P. Rétornaz *et al.*, “Cooperative navigation in robotic swarms,” *Swarm Intelligence*, vol. 8, no. 1, pp. 1–33, 2014.

[21] R. Fabbiano, C. C. De Wit, and F. Garin, “Source localization by gradient estimation based on poisson integral,” *Automatica*, vol. 50, no. 6, pp. 1715–1724, 2014.

[22] M. F. Fallon, G. Papadopoulos, J. J. Leonard, and N. M. Patrikalakis, “Cooperative auv navigation using a single maneuvering surface craft,” *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1461–1474, 2010.

[23] S. Gil, D. Feldman, and D. Rus, “Communication coverage for independently moving robots,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4865–4872.

[24] S. Gil, S. Kumar, D. Katabi, and D. Rus, “Adaptive communication in multi-robot systems using directionality of signal strength,” *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 946–968, 2015.

[25] S. Gil, M. Schwager, B. J. Julian, and D. Rus, “Optimizing communication in air-ground robot networks using decentralized control,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1964–1971.

[26] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2013, vol. 207.

[27] Google, “Project loon,” https://x.company/loon/.

[28] A. Gunatilaka, B. Ristic, A. Skvortsov, and M. Morelande, “Parameter estimation of a continuous chemical plume source,” in *Information Fusion, 2008 11th International Conference on*. IEEE, 2008, pp. 1–8.

[29] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, “Maintaining network connectivity and performance in robot teams,” *Journal of field robotics*, vol. 25, no. 1-2, pp. 111–131, 2008.

[30] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein, “Graph-based distributed cooperative navigation for a general multi-robot measurement model,” *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1057–1080, 2012.

[31] M. Ji and M. Egerstedt, “Distributed coordination control of multiagent systems while preserving connectedness,” *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[32] T. Jiřinec, “Stabilization and control of unmanned quadcopter,” 2011.

[33] C. Luis and J. L. Ny, “Design of a trajectory tracking controller for a nanoquadcopter,” *arXiv preprint arXiv:1608.05786*, 2016.

[34] I. V. Melnyk, J. A. Hesch, and S. I. Roumeliotis, "Cooperative vision-aided inertial navigation using overlapping views," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 936–943.

[35] L. Merino, F. Caballero, J. R. Martínez-De-Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 533–548, 2012.

[36] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[37] N. Michael, M. Schwager, V. Kumar, and D. Rus, "An experimental study of time scales and stability in networked multi-robot systems," in *Experimental Robotics*. Springer, 2014, pp. 631–643.

[38] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Maintaining connectivity in mobile robot networks," in *Experimental Robotics*. Springer, 2009, pp. 117–126.

[39] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1093–1099.

[40] R. M. Murray, "Recent research in cooperative control of multivehicle systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 2007.

[41] A. Nedić and D. P. Bertsekas, "The effect of deterministic noise in subgradient methods," *Mathematical programming*, vol. 125, no. 1, pp. 75–99, 2010.

[42] T. Nestmeyer, A. Franchi, H. H. Bülthoff, and P. R. Giordano, "Decentralized multi-target exploration and connectivity maintenance with a multi-robot system," in *RSS 2015 Workshop: Reviewing the review process*, 2015, pp. 1–8.

[43] G. E. Newstadt, B. Mu, D. Wei, J. P. How, and A. O. Hero III, "Resource-constrained adaptive search for sparse multi-class targets with varying importance," *arXiv preprint arXiv:1409.7808*, 2014.

[44] N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," in *Aerospace Conference, 2008 IEEE*. IEEE, 2008, pp. 1–14.

[45] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining limited-range connectivity among second-order agents," in *American Control Conference, 2006*. IEEE, 2006, pp. 6–pp.

[46] U. S. D. of Transportation, "Automated car platoon," https://www.volpe.dot.gov/news/how-automated-car-platoon-works.

[47] J. Ondrácek, O. Vanek, and M. Pechoucek, "Solving infrastructure monitoring problems with multiple heterogeneous unmanned aerial vehicles," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1597–1605.

[48] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 5, pp. 1068–1080, 2006.

[49] Z. Petroulias, "Multiple ground robots platform," http://monash-wsrnlab.blogspot.com/2014/08/network-formation-control-methods.html.

[50] I. H. Potential, "How robots will save your life when disaster strikes," http://increasinghumanpotential.org/how-robots-will-save-your-life-when-disaster-strikes/.

[51] J. Redding, T. Toksoz, N. K. Ure, A. Geramifard, J. P. How, M. A. Vavrina, and J. Vian, "Distributed multi-agent persistent surveillance and tracking with health management," in *Proceedings of the AIAA Guidance Navigation and Control Conference, Portland, OR, USA*, 2011, pp. 8–11.

[52] W. Ren and Y. Cao, *Distributed coordination of multi-agent networks: emergent problems, models, and issues*. Springer Science & Business Media, 2010.

[53] F. Research, "The connectivity lab," https://research.fb.com/category/connectivity/.

[54] L. Sabattini, N. Chopra, and C. Secchi, “On decentralized connectivity maintenance for mobile robotic systems,” in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 988–993.

[55] A. C. Sanderson, “A distributed algorithm for cooperative navigation among multiple mobile robots,” *Advanced Robotics*, vol. 12, no. 4, pp. 335–349, 1997.

[56] M. Schwager, N. Michael, V. Kumar, and D. Rus, “Time scales and stability in networked multi-robot systems,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3855–3862.

[57] I. Shames, S. Dasgupta, B. Fidan, and B. D. Anderson, “Circumnavigation using distance measurements,” in *Control Conference (ECC), 2009 European*. IEEE, 2009, pp. 2444–2449.

[58] Y. Shen, S. Mazuelas, and M. Z. Win, “Network navigation: Theory and interpretation,” *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 9, pp. 1823–1834, 2012.

[59] D. P. Spanos and R. M. Murray, “Robust connectivity of networked vehicles,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 3. IEEE, 2004, pp. 2893–2898.

[60] ——, “Motion planning with wireless network constraints,” in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 87–92.

[61] J. Stephan, J. Fink, B. Charrow, A. Ribeiro, and V. Kumar, “Robust routing and multi-confirmation transmission protocol for connectivity management of mobile robotic teams,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3753–3760.

[62] J. O. Swartling, I. Shames, K. H. Johansson, and D. V. Dimarogonas, “Collective circumnavigation,” *Unmanned Systems*, vol. 2, no. 03, pp. 219–229, 2014.

[63] V. Trianni, “Swarm robotics for agricultural applications,” http://insideunmannedsystems.com/swarming-the-skies/.

[64] N. K. Ure, G. Chowdhary, T. Toksoz, J. P. How, M. A. Vavrina, and J. Vian, "An automated battery management system to enable persistent missions with multiple aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 1, pp. 275–286, 2015.

[65] N. K. Ure, T. Toksoz, G. Chowdhary, J. Redding, J. How, M. Vavrina, and J. Vian, "Experimental demonstration of multi-agent learning and planning under uncertainty for persistent missions with automated battery management," in *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4622.

[66] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, 2005, pp. 154–165.

[67] A. R. Vetrella, G. Fasano, A. Renga, and D. Accardo, "Cooperative uav navigation based on distributed multi-antenna gnss, vision, and mems sensors," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 1128–1137.

[68] M. Z. Win, A. Conti, S. Mazuelas, Y. Shen, W. M. Gifford, D. Dardari, and M. Chiani, "Network localization and navigation via cooperation," *IEEE Communications Magazine*, vol. 49, no. 5, 2011.

[69] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[70] S.-k. Yun and D. Rusy, "Distributed coverage with mobile robots on a graph: Locational optimization," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 634–641.

[71] M. M. Zavlanos and G. J. Pappas, "Controlling connectivity of dynamic graphs," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 6388–6393.

[72] ——, "Potential fields for maintaining connectivity of mobile networks," *IEEE Transactions on robotics*, vol. 23, no. 4, pp. 812–816, 2007.

[73] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Mobility & routing control in networks of robots," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7545–7550.

[74] A. Zenonos, S. Stein, and N. R. Jennings, "Coordinating measurements for air pollution monitoring in participatory sensing settings," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 493–501.

[75] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware design experiences in zebranet," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 227–238.

[76] Y. Zhang and Y. Meng, "Dynamic multi-robot task allocation for intruder detection," in *Information and Automation, 2009. ICIA'09. International Conference on*. IEEE, 2009, pp. 1081–1086.