Distributed Intermittent Connectivity Control of Mobile Robot Networks

by

Ioannis Kantaros

Department of Mechanical Engineering and Materials Science
Duke University

Date: ______________________

Approved:

___________________________
Michael M. Zavlanos, Supervisor

___________________________
Miroslav Pajic

___________________________
Kristoffer Hauser

___________________________
Mary ‘Missy’ Cummings

___________________________
George J. Pappas

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mechanical Engineering and Materials Science in the Graduate School of Duke University 2018
Abstract

Distributed Intermittent Connectivity Control of Mobile Robot Networks

by

Ioannis Kantaros

Department of Mechanical Engineering and Materials Science
Duke University

Date: __________________
Approved:

__________________________
Michael M. Zavlanos, Supervisor

__________________________
Miroslav Pajic

__________________________
Kristoffer Hauser

__________________________
Mary ‘Missy’ Cummings

__________________________
George J. Pappas

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mechanical Engineering and Materials Science in the Graduate School of Duke University
2018
Abstract

Wireless communication is known to play a pivotal role in enabling teams of robots to successfully accomplish global coordinated tasks. In fact, network connectivity is an underlying assumption in every distributed control and optimization algorithm. For this reason, in recent years, there is growing research in designing controllers that ensure point-to-point or end-to-end network connectivity for all time. Nevertheless, all these methods severely restrict the robots from accomplishing their tasks, as motion planning is always restricted by connectivity constraints on the network. Instead, a much preferred solution is to enable robots to communicate in an intermittent fashion, and operate in disconnect mode the rest of the time giving rise to an intermittently connected communication network. While in disconnect mode, the robots can accomplish their tasks free of communication constraints. The goal of this dissertation is to design a distributed intermittent connectivity framework that (i) ensures that the communication network is connected over time, infinitely often (ii) is flexible enough to account for arbitrary dynamic tasks, and (iii) can be applied to large-scale networks.

The great challenge in developing intermittent connectivity protocols for networks of mobile robots is to decide (i) which robots talk to which, (ii) where, and (iii) when, so that the communication network is connected over time infinitely often. To address these challenges, we decompose the network into small groups of robots, also called teams, so that every robot belongs to at least one team and that there
is a path, i.e., a sequence of teams, where consecutive teams have non-empty intersections, connecting every two teams of robots, so that information can propagate in the network. First, given such fixed teams, we design infinite sequences of communication events for all robots, also called communication schedules, independent of the tasks assigned to the robots, that determine when every team should communicate, so that the communication network is connected over time infinitely often. The designed communication schedules ensure that all teams communicate infinitely often, i.e., that the communication network is connected over time infinitely often. Between communication events the robots can move in the workspace free of communication constraints to accomplish their assigned tasks. Theoretical guarantees and numerical experiments corroborate the proposed framework. This is the first distributed intermittent connectivity framework that can be applied to large-scale networks and is flexible enough to account for arbitrary dynamic robot tasks.

Next, given user-specified fixed teams, we integrate the respective communication schedules with task planning. Specifically, we consider high-level complex tasks captured by temporal logic formulas, state-estimation tasks, and time-critical dynamic tasks. The proposed distributed integrated path planning and intermittent connectivity frameworks determine both where and when every team should communicate so that the assigned task is accomplished, the communication network is connected over time infinitely often, and a user-specified metric, such as total traveled distance or consumed energy, is minimized. We show that employing the proposed intermittent connectivity framework for such tasks results in significant performance gains compared to the existing solutions in the literature that maintain connectivity for all time. Theoretical guarantees, numerical and experimental studies support the proposed distributed control algorithms.

Finally, we propose a fully autonomous intermittent connectivity framework that can handle arbitrary dynamic tasks and also allows the robots to locally and online
update the structure of the teams and the communication schedules, effectively allowing them to decide *who they should talk to*, so that they can better accomplish newly assigned tasks. The structure of the teams, the associated communication locations, and the time instants when communication within teams will occur are integrated online with task planning giving rise to paths, i.e., sequences of waypoints, that ensure that the assigned task is accomplished, the communication network is connected over time infinitely often, and a user specified metric is minimized. This is the first fully autonomous, distributed, and online intermittent connectivity framework that can handle arbitrary dynamic tasks and also controls the topology of the intermittently connected robot network to better accomplish these tasks. At the same time, the proposed framework scales well with the size of the robot network. Theoretical guarantees and numerical experiments corroborate the proposed distributed control scheme.
To my family and friends
Contents

Abstract iv
List of Figures xii
Acknowledgements xvi

1 Introduction 1

1.1 Motivation ......................................................... 1

1.2 Related Work ..................................................... 5

1.2.1 Graph-based Connectivity Control ......................... 5

1.2.2 Realistic Communication Models ......................... 6

1.2.3 Intermittent Communication .............................. 6

1.3 Thesis Contents ................................................. 8

1.3.1 Chapter 2 ..................................................... 8

1.3.2 Chapter 3 ..................................................... 9

1.3.3 Chapter 4 ..................................................... 12

1.3.4 Chapter 5 ..................................................... 14

1.3.5 Chapter 6 ..................................................... 16

1.4 Contributions ................................................... 17

1.4.1 Contributions of Chapter 2 .................... 17

1.4.2 Contributions of Chapter 3 .................... 18

1.4.3 Contributions of Chapter 4 .................... 18

viii
1.4.4 Contributions of Chapter 5 .................................. 19
1.4.5 Contributions of Chapter 6 .................................. 19

2 Distributed Intermittent Connectivity Control of Mobile Robot Networks 20

2.1 Problem Formulation ............................................. 21
2.2 Construction of Communication Schedules ....................... 24
2.3 Online Asynchronous Execution of Communication Schedules .... 30
2.4 Correctness .......................................................... 31
2.5 Numerical Experiments ........................................... 34

3 Temporal Task Planning and Intermittent Connectivity Control for Mobile Robot Networks 38

3.1 Preliminaries ....................................................... 39
3.2 Problem Formulation ............................................. 40
3.3 Task Planning ..................................................... 44
3.4 Integrated Task Planning and Intermittent Communication Control 47
    3.4.1 Modification of motion plans $\tau_i$ .......................... 48
    3.4.2 Online Asynchronous Execution of Motion Plans ........... 56
3.5 Correctness and Optimality ...................................... 57
3.6 Simulation Studies ............................................... 68
3.7 Experimental Results ............................................ 73
    3.7.1 System Description ....................................... 74
    3.7.2 Experimental Results .................................... 76

4 Distributed State Estimation and Intermittent Connectivity Control of Mobile Robot Networks 79

4.1 Problem Formulation ............................................. 80
4.2 Informative Path Planning and Intermittent Communication Control 82
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1</td>
<td>Informative Planning</td>
<td>82</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Sampling-based Planning</td>
<td>86</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Computation of $\text{Cost}(v_{\text{new}})$</td>
<td>91</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Completeness and Optimality</td>
<td>93</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Integrated Path Planning and Intermittent Communication Control</td>
<td>94</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Correctness</td>
<td>95</td>
</tr>
<tr>
<td>4.3</td>
<td>Simulation Studies</td>
<td>98</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Comparative Results for Different Graphs $G_T$</td>
<td>100</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Comparison with Alternative Approaches</td>
<td>106</td>
</tr>
<tr>
<td>4.4</td>
<td>Steering function</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Distributed Intermittent Connectivity Control for Mobile Robot</td>
<td>111</td>
</tr>
<tr>
<td>5.1</td>
<td>Networks under Time-Critical Dynamic Tasks</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Problem Formulation</td>
<td>112</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Integrated Task Planning and Intermittent Communication Control</td>
<td>116</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Construction of paths $\mathcal{P}_i(t)$</td>
<td>117</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Selection of Next Communication Point</td>
<td>118</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Selection of Next Meeting Time Instant</td>
<td>122</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Online Execution of Paths $\mathcal{P}_i(t)$</td>
<td>123</td>
</tr>
<tr>
<td>5.3</td>
<td>Correctness</td>
<td>124</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation Studies</td>
<td>126</td>
</tr>
<tr>
<td>6</td>
<td>Distributed Intermittent Connectivity Control of Mobile Robot Networks under Dynamics Tasks: Who Talks to Who, Where, and When</td>
<td>131</td>
</tr>
<tr>
<td>6.1</td>
<td>Problem Formulation</td>
<td>132</td>
</tr>
<tr>
<td>6.2</td>
<td>Intermittent Connectivity Control</td>
<td>136</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Definition of Communication Schedules</td>
<td>137</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Autonomous robots. ........................................ 2
1.2 Network of autonomous cars and drones. ...................... 3
1.3 Intermittent connectivity in a surveillance task over a dispersed residential area. ........................................ 4

2.1 Graphical depiction of a connected-over-time communication graph $G_c(t)$ ........................................ 23
2.2 Team membership graph. ..................................... 28
2.3 Intermittent connectivity of a network of robots. ............... 35
2.4 Graphical depiction of consensus. ................................ 36
2.5 Graphical depiction of communication events. .................. 37

3.1 A graphical illustration of the problem formulation. ............ 44
3.2 Illustration of the proposed decentralized solution. ............. 45
3.3 Illustration of Algorithm 3. .................................... 53
3.4 Graphical illustration of the temporal task planning algorithm under intermittent connectivity constraints. .............. 54
3.5 Team membership graph. ....................................... 69
3.6 Graphical illustration of intermittent connectivity for a network of robots under temporal logic tasks. ...................... 71
3.7 Graphical depiction of consensus of numbers. .................. 72
3.8 Graphical depiction of communication events with respect to time. 73
3.9 Evolution of the total cost. .................................... 74
3.10 Experimental setup. .................................................. 75
3.11 Robot trajectories. .................................................. 77
3.12 Evolution of communication events among the robots, and among the robots and the user over time. .................................................. 78
3.13 Evolution of messages stored at each robot and the user. ........ 78
4.1 Illustration of the proposed path planning algorithm for distributed state estimation under intermittent connectivity constraints. .......... 84
4.2 Schematic representation of a sampling-based solution for the state estimation problem. .................................................. 88
4.3 Team membership graph. .......................................... 101
4.4 Graphical depiction of the estimation error and the uncertainty metric with respect to time. .................................................. 102
4.5 Comparative state estimation results. ................................ 103
4.6 Graphical depiction of propagation delays. ......................... 105
4.7 Graphical depiction of robot paths. ................................ 106
4.8 Comparative results. .................................................. 108
5.1 Graphical illustration of the problem formulation 5.1.1. .......... 115
5.2 Graphical illustration of selecting optimal communication points. 119
5.3 Graphical depiction of number of task waypoints that were visited between consecutive communication events. .................. 128
5.4 Illustration of waiting time. ........................................ 129
6.1 Graphical illustration of the problem formulation 6.1.3. .......... 136
6.2 Graphical depiction of the proposed control framework for task planning and intermittent communication control. ............... 137
6.3 Graphical illustration of selecting optimal communication points. 150
6.4 Graphical depiction of the waiting time. ........................... 170
6.5 Graphical depiction of consensus. .................................. 171
6.6 Graphical depiction of the second smallest eigenvalue and the maximum degree of team membership graph. 172
6.7 Evolution of the structure of the teams. 173
6.8 Graphical depiction of the total traveled distance. 174
## List of Algorithms

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Decentralized construction of sequence $s_i, i \in \mathcal{T}_m$</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>Online Asynchronous and Local Execution of $\text{sched}_i(t), i \in \mathcal{T}_m$</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Decentralized construction of motion plan $\tilde{\tau}_i$</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>Asynchronous execution of $\tilde{\tau}_i$</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>Sampling-based Informative Path Planning</td>
<td>87</td>
</tr>
<tr>
<td>6</td>
<td>Extend</td>
<td>89</td>
</tr>
<tr>
<td>7</td>
<td>Rewire</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>Extended Kalman Filter</td>
<td>93</td>
</tr>
<tr>
<td>9</td>
<td>Proposed Control Framework for robot $r_{ij}$</td>
<td>95</td>
</tr>
<tr>
<td>10</td>
<td>Steer</td>
<td>110</td>
</tr>
<tr>
<td>11</td>
<td>Computation of meeting time instant $t^m$</td>
<td>123</td>
</tr>
<tr>
<td>12</td>
<td>Online Execution of $\mathcal{P}_i(t), t \geq t_0$</td>
<td>124</td>
</tr>
<tr>
<td>13</td>
<td>Online Redesign of Communication Schedules</td>
<td>143</td>
</tr>
<tr>
<td>14</td>
<td>Online Execution of $\text{sched}_i(t), i \in \mathcal{T}_m(t)$</td>
<td>145</td>
</tr>
<tr>
<td>15</td>
<td>Coordination during a communication event for team $\mathcal{T}_m(t)$ at time $t$</td>
<td>149</td>
</tr>
<tr>
<td>16</td>
<td>Computation of meeting time instant $t^m$</td>
<td>153</td>
</tr>
<tr>
<td>17</td>
<td>Update of Teams</td>
<td>156</td>
</tr>
<tr>
<td>18</td>
<td>Online Execution of $\mathcal{P}_i(t), t \geq t_0$</td>
<td>160</td>
</tr>
</tbody>
</table>
Acknowledgements

First and foremost, I would like to sincerely thank my advisor, Professor Michael M. Zavlanos for his guidance, support, inspiration and the freedom he gave me to work on any research project of my interest. Michael has supported me not only by providing a research assistantship over the past five years, but also academically and emotionally through the rough road to finish this dissertation. Thanks to Michael, my Ph.D. experience at Duke has been nothing short of amazing.

My sincere thanks go to the members of my PhD committee, Professors Miroslav Pajic, Missy Cummings, and Kris Hauser from Duke University and George J. Pappas from the University of Pennsylvania for their constructive feedback and comments. Also, my deepest gratitude goes to my former advisor at the University of Patras, Professor Anthony Tzes with whom I had a very productive and enjoyable collaboration as a research assistant in his lab that sparked my interest in robotics and controls.

Besides faculty, I would like to thank all the lab-mates who have made my experience as a graduate student a wonderful journey: Nikolaos Chatzipanagiotis, Charles Freundlich, Reza Khodayi-mehr, Luke Calkins, Yan Zhang, Xusheng Luo, Soomin Lee, Meng Guo, Wann-Juin Ma, Yuankai Zhu, and Qitong Gao. A very big thank you goes to Meng and Reza with whom I worked closely to derive the results presented in Chapters 3 and 4, respectively. A very special thank you goes to Judy Rowley and Jerry Steiner for supporting me in many ways during the last five years.
Last but not least, I would like to thank all my friends in Greece and my family for their unconditional love and support.
1

Introduction

1.1 Motivation

Autonomous robots, such as drones, self-driving cars, and household robots are emerging as integral parts of our daily lives in the foreseeable future; see Figure 1.1. An essential feature of autonomous robots is that they are expected to understand complex tasks specified by non-expert users, autonomously synthesize controllers to accomplish these tasks in an optimal way, and react to any contingent changes in the environment without or with minimal human intervention. Such tasks involve area coverage Cortés et al. (2004); Kantaros et al. (2015); Kantaros and Zavlanos (2016a), area exploration and mapping Papachristos et al. (2017b,a), surveillance Ghaffarkhah and Mostofi (2012); Leahy et al. (2016); Nardi et al. (2016), environmental monitoring He et al. (2012), search and rescue missions Nourbakhsh et al. (2005), state estimation and active sensing Nerurkar et al. (2014); Huang et al. (2015); Freundlich et al. (2016), and intruder detection Spindler et al. (2012); Pasqualetti et al. (2014).

A group of coordinated autonomous robots can be more efficient in accomplishing such complex tasks than single robots due to the benefits they provide such as flexibility, adaptivity, robustness to robot failures, and scalability. Significant improvement
to scalability, robustness, and adaptivity mentioned above can be achieved if control of such robot networks is performed in a decentralized way, relying on cooperation and communication among the robots; see Figure 1.2. Cooperation is required so that the robots can accomplish collaborative tasks while making decisions that depend only on individual preferences and access to local information. On the other hand, cooperation requires information sharing between robots by means of reliable wireless communication networks that are established by the team. Indeed, network connectivity is a requirement in every distributed control Martinez et al. (2007); Bullo et al. (2009); Cortés and Egerstedt (2017) and optimization algorithm Bertsekas and Tsitsiklis (1989); Mota et al. (2013); Chatzipanagiotis et al. (2015).

As wireless communication plays a pivotal role in enabling teams of robots to successfully accomplish coordinated tasks and/or transmit gathered data to a user, in the recent years, there has been a growing research interest in developing distributed controllers that ensure all-time network connectivity. Such controllers rely on either graph-theoretic approaches or more realistic communication models that consider path loss, shadowing, and multi-path fading as well as optimal routing decisions for desired information rates. Nevertheless, in practice, limited resources, e.g., transmission power or number of wireless robots, or the effect of the ambient envi-
Communication among the vehicles is required for coordination and collision avoidance. Environment on signal strength often render impossible to ensure all-time connectivity. This is the case, e.g., in underwater applications Leonard et al. (2007), such as ocean exploration, where the robots typically communicate over aquatic acoustic channels that are not practical when the robots need to be deployed tens of kilometers apart due to limited bandwidth, or in applications that take place in cluttered and dispersed environments where long distances and environmental interference affect the quality of communications; see also Figure 1.3. Maintaining all-time connectivity in these applications is rather conservative and may prevent robots from accomplishing their goals as the robot mobility is always restricted by connectivity/proximity constraints. Therefore, a much preferred solution is to allow robots to communicate in an intermittent fashion and operate in disconnect mode the rest of the time to accomplish their assigned tasks.

In an intermittent communication setup, the goal is to remove all communication constraints and require that the robots communicate only temporarily when they are physically close to each other and construct a connected network. While in disconnect mode, the robots can accomplish their tasks free of communication constraints. A great challenge in designing intermittent communication protocols is to
Figure 1.3: Intermittent connectivity in a surveillance task over a dispersed residential area. Robots (1,2,3) communicate at communication locations (A,B,C) and operate in disconnect mode the rest of the time.

determine which robots should talk to which, where, and, more importantly, when. Equally challenging is to determine these aspects of the solution in a distributed way and in the presence of uncertainty, e.g., at the time instants at which a connected configuration among the robots needs to be formed. The reason is that distributed control requires communication, which in an intermittent communication setup is not readily available. Moreover, another challenging aspect is to integrate intermittent communication control with task planning and design robot paths in the absence of a connected communication network.

The purpose of this thesis is to design a distributed intermittent communication framework that (i) ensures that the dynamic communication network is connected over time, infinitely often (ii) is flexible enough to account for arbitrary dynamic robot tasks, and (iii) can be applied to large-scale networks.
1.2 Related Work

1.2.1 Graph-based Connectivity Control

Communication among robots has been typically modeled using proximity graphs and the communication problem is often treated as preservation of graph connectivity. For example, in Spanos and Murray (2004) a function that measures the local connectedness of a network is introduced, which under mild assumptions can provide conditions for global network connectivity. Alternative methods to control graph connectivity rely on controlling the Fiedler value of the underlying graph either in a centralized Kim and Mesbahi (2006) or distributed DeGennaro and Jadbabaie (2006); Yang et al. (2010); Montijano et al. (2011); Franceschelli et al. (2013); Sabattini et al. (2013) fashion. Also, potential fields that model loss of connectivity as an obstacle in the free space can be employed for connectivity maintenance, as shown in Zavlanos and Pappas (2007). A distributed hybrid approach to connectivity control is presented in Zavlanos and Pappas (2008) that decouples control of the discrete communication graph from continuous robot mobility via an efficient manipulation of communication links. Further distributed controllers for graph connectivity maintenance have been implemented in Ji and Egerstedt (2007); Savla et al. (2009). The relevant literature is quite extensive and a recent survey on graph-theoretic methods for connectivity control can be found in Zavlanos et al. (2011). These results have been successfully applied to multiple network coordinated robot tasks, suchs rendezvous Dimarogonas and Kyriakopoulos (2008), flocking Zavlanos et al. (2009), formation control Ji and Egerstedt (2007), and temporal task planning Filippidis et al. (2012).
1.2.2 Realistic Communication Models

In practice, the above graph-based communication models turn out to be rather conservative, since proximity does not necessarily imply tangible and reliable communication. A more realistic communication model for mobile networks compared to the above graph-theoretic models is proposed in Zavlanos et al. (2013, 2010) that takes into account the routing of packets as well as desired bounds on the transmitted rates. In this model, a weighted graph is employed to capture the inter-robot communication with weights that are associated with the packet error probability. Integration of this communication framework with area coverage control and navigation tasks in complex environments is presented in Kantaros and Zavlanos (2014b,a, 2016a,c).

A conceptually similar communication model is proposed in Fink et al. (2010, 2012) that models the communication rates among robots as random variables, while the routing of information is performed so that the uncertainty in the link rates is reduced. A communication model that accounts for multi-path fading of channels is proposed in Lindhé and Johansson (2010), where robot mobility is exploited in order to increase the throughput. Multi-path fading, shadow fading, and path loss have also been used to model channels in Mostofi et al. (2010); Malmirchegini and Mostofi (2012). In these works, a probabilistic framework for channel prediction is developed based on a small number of measurements of the received signal power. The integration of the latter communication models with robot mobility is described in Ghaffarkhah and Mostofi (2010); Yan and Mostofi (2012). A related approach pertaining to the on-line evaluation of wireless channels is presented in Le Ny et al. (2012) based on a sampling scheme for the link capacities.

1.2.3 Intermittent Communication

Although controllers that ensure all-time connectivity has been extensively studied using either graph-based communication models or more realistic ones, intermittent
connectivity is still widely considered an assumption. The reason is that connectivity is a global network property that requires coordination between the robots to control, which is unavailable when the robots operate in disconnect mode. Specifically, in multi-agent systems, intermittent communication has been studied in consensus problems Wen et al. (2014b,a), task assignment Zavlanos et al. (2008), state estimation Leung et al. (2012); Freundlich et al. (2016), coverage problems Wang and Hussein (2010), and in delay-tolerant networks Daly and Haahr (2007); Lindgren et al. (2003); Zhang (2006); Jones et al. (2007); Costa et al. (2008). The common assumption in these works is that the communication network is connected over time infinitely often. Relevant is also the work on event-based network control Dimarogonas et al. (2012); Tabuada (2007); Wang and Lemmon (2011) where, although the network is assumed to be connected for all time, messages between the agents are exchanged intermittently when certain events take place.

The connectivity-over-time assumption is lifted in Guo and Zavlanos (2017); Hollinger and Singh (2010); Hollinger et al. (2012); Zavlanos (2010). Specifically, Guo and Zavlanos (2017) controls groups of data-gathering source robots and relay robots that receive information from the source robots and transmit it to a data center, while respecting buffer size constraints. Note that this framework allows information to flow only to a root/user and not among the robots. On the other hand, in Hollinger and Singh (2010); Hollinger et al. (2012); Zavlanos (2010) information can flow intermittently between any pair of robots and possibly a user. In particular, Hollinger and Singh (2010) proposes a receding horizon framework for periodic connectivity under information gathering tasks that ensures recovery of end-to-end connectivity within a given time horizon. Experimental validation of this framework is presented in Hollinger et al. (2012). However, as the number of robots or the size of the time horizon grows, this approach can become computationally expensive. Also, when connectivity is recovered, the whole network needs to be connected which is
not the case in Zavlanos (2010). Specifically, Zavlanos (2010) proposes a distributed synchronization scheme that allows robots that move along the edges of a bipartite mobility graph to meet periodically at the vertices of this graph ensuring that the communication network is connected over time infinitely often.

1.3 Thesis Contents

In Chapter 1.1, we motivated the need to relax the requirement of all-time connectivity in multi-robot systems and, instead, control communication intermittently. The goal is to design a distributed intermittent communication protocol that ensures that the communication network is connected over time infinitely often, is flexible enough to account for various multi-robot missions and can be applied to large-scale networks. The main challenges in controlling communication in any multi-agent system in an intermittent fashion are to enable the agents to decide locally, online, and optimally (iii) which agents/robots they should talk to which, (ii) where, and (iii) when, so that the communication network is connected over time, infinitely often. Integration of intermittent connectivity protocols with task planning must also be explored.

1.3.1 Chapter 2

In Chapter 2, we propose a distributed intermittent connectivity framework for mobile multi-robot systems that is agnostic to the assigned task and can be applied to large-scale networks. Specifically, we consider situations where the robot communication capabilities are not sufficient to form reliable and connected networks while the robots move to accomplish their tasks. Since the robots have limited communication capabilities, they can communicate only sporadically when they are physically close to each other. To define a communication network, we decompose the network into small groups of robots, also called teams, that are required to form connected
sub-networks infinitely often giving rise to an intermittently connected dynamic communication network. These teams are designed so that every robot belongs to at least one team and that there is a path, i.e., a sequence of teams, where consecutive teams have non-empty intersections, connecting every two teams of robots, so that information can propagate in the network. Given such fixed teams, we design infinite sequences of communication events for all robots, also called communication schedules, independent of their assigned tasks, that determine when every team should communicate, so that the communication network is connected over time infinitely often. The designed communication schedules ensure that all teams communicate infinitely often, i.e., that the communication network is connected over time infinitely often. Between communication events the robots can move in the workspace free of communication constraints to accomplish their assigned tasks. Theoretical guarantees and numerical experiments corroborate the proposed framework. In Chapters 3-5, we integrate the proposed intermittent communication protocol with task planning. These results have been reported in Kantaros and Zavlanos (2017a, 2016d); Kantaros et al. (2017b). Relevant publications that have motivated this work but are not included in this thesis are presented in Kantaros and Zavlanos (2014a,b, 2016a,c).

1.3.2 Chapter 3

In Chapter 3, we propose an offline, decentralized, task planning and intermittent communication framework for mobile robot networks. Specifically, the goal of the robots is to accomplish complex, high-level tasks, captured by local Linear Temporal Logic formulas (LTL), and share the collected information with all other robots and possibly also with a user. We also consider situations where the robot communication capabilities are not sufficient to form reliable and connected networks while the robots move to accomplish their tasks. Therefore, we assume that the robots can
only communicate with each other when they meet at common locations in space. To define a communication network, we decompose the network into smaller teams of robots, as discussed in Chapter 2, that are fixed with time. Then, we require robots in every team to meet infinitely often at locations in space, also called communication locations, that are common for each team, but possibly different across teams giving rise to an intermittently connected dynamic communication network. Then, our goal is to design motion plans for every robot that ensure that the local LTL tasks are satisfied, the communication network is connected over time infinitely often, i.e., robots in every team meet and communicate infinitely often at one of the available communication locations, and a user-specified metric, such as total traveled distance, is minimized.

To address this problem, we first design optimal motion plans that satisfy the assigned local tasks ignoring the intermittent connectivity requirement using existing optimal control synthesis algorithms for multi-robot systems under local LTL specifications Smith et al. (2011); Guo and Dimarogonas (2013, 2015). Then, we integrate these local discrete plans with the schedules designed in Chapter 2 by introducing communication locations in the discrete motion plans of the robots so that the local tasks are satisfied, the communication network is intermittently connected as per the designed schedules, and a user-specified metric is minimized. We provide theoretical guarantees supporting the proposed framework, as well as numerical simulations and experiments showing its ability to solve very large and complex planning problems that relevant approaches cannot solve.

We also show that the considered problem can be cast as an optimal control synthesis problem under global temporal logic specifications Ulusoy et al. (2013, 2014). Specifically, we show that the intermittent connectivity requirement can be modeled by a global LTL statement that forces the teams to meet infinitely often at communication locations that minimize a user-defined metric, such as total traveled distance.
Then, the considered problem is equivalent to synthesizing motion plans that satisfy both the local LTL tasks and the global LTL formula capturing the intermittent connectivity requirement while minimizing a user-specified metric. Such a problem can be solved by existing optimal control synthesis algorithms that consider global temporal tasks Ulusoy et al. (2013, 2014) that employ tools from model checking and concurrency theory Baier and Katoen (2008); Clarke et al. (1999). Nevertheless, these approaches are centralized and rely on the construction of a synchronous product automaton among all robots and application of graph search methods to synthesize optimal plans. Therefore, such methods are resource demanding and scale poorly with the number of robots. A more tractable optimal control synthesis algorithm is presented in Schillinger et al. (2016) that identifies independent parts of the LTL formula and builds a local product automaton for each agent. Nevertheless, this approach can be applied only to finite LTL missions and, therefore, it cannot capture the intermittent connectivity requirement. Also, a new logic, called counting linear temporal logic, is proposed in Sahin et al. (2017) that can be applied to coordination of large collections of agents. However, this logic assumes that the identity of the agents is not important for the successful accomplishment of the task, which is not the case in our work due to the intermittent connectivity requirement. Also, our synthesized controllers can be executed in an asynchronous fashion, which is not the case in most relevant literature in control synthesis as, e.g., in Kloetzer and Belta (2010); Pola et al. (2016). Note that the proposed framework can be extended to any robot task that is accomplished by periodic trajectories, as e.g., Lan and Schwager (2016); Nardi et al. (2016). These results have been reported in Kantaros et al. (2017b). Relevant publications that motivated this work but are not reported in this thesis are Kantaros and Zavlanos (2015, 2017b,c, 2018c); Kantaros et al. (2017a).
1.3.3 Chapter 4

In Chapter 4 we propose an online, distributed state estimation and intermittent connectivity framework for mobile robot networks. Specifically, we consider networks of robots that are tasked with collectively estimating a collection of hidden states. As in Chapter 3, we assume that the robots have limited communication capabilities and, therefore, communicate their measurements only sporadically, when they are physically close to each other. To define a communication network, we decompose the network of robots into smaller teams, as discussed in Chapter 2, and we require that the robots in every team form a connected configuration somewhere in the workspace giving rise to an intermittently connected dynamic communication network. Our goal in this chapter is to design robot paths that ensure that the uncertainty and the estimation error of the hidden states is minimized and the dynamic communication network is connected over time infinitely often. Note that unlike Chapter 3, in this chapter, the teams of robots do not meet at a priori known locations in space to communicate, but instead they form connected sub-networks that are unknown and they are decided by the robots considering their objective to explore their environment and gather information.

To address this problem, given fixed teams of robots, we first design communication schedules, as in Chapter 2, that ensure that the dynamic communication network is connected over time infinitely often. When the robots in a team form a connected configuration, they exchange the information they have collected along their respective paths and update their beliefs using an estimation filter. The connected subnetworks formed by each team and the corresponding informative robot paths are determined by an online, distributed, sampling-based motion planner that takes into account the robots’ objective to gather information in their environment. The proposed sampling-based algorithm is built upon the RRT* algorithm Karaman
and Frazzoli (2011). Specifically, we have modified this algorithm so that all robots in a subgroup form a connected subnetwork somewhere in the workspace (depending on the estimation task) and they arrive at this location the same time.

Sampling-based algorithms for informative path planning are presented in Lan and Schwager (2016); Hollinger and Sukhatme (2013, 2014). Specifically, a variation of the RRT* algorithm Karaman and Frazzoli (2011) is proposed in Lan and Schwager (2016) that can design offline periodic trajectories to estimate the state of a dynamic field. The works in Hollinger and Sukhatme (2013, 2014) build upon the RRG algorithm Karaman and Frazzoli (2011) to design motion plans that maximize an information theoretic metric subject to budget constraints associated with the traveled distance. Nevertheless, this approach cannot be used when the length of the designed path, i.e., the budget, is not known a priori, as in the problem considered in this chapter. Common in Hollinger and Sukhatme (2013, 2014) is that they address single-agent motion planning problems. Applying these algorithms to multi-agent problems would require exploration of the joint space of all robots rendering them computationally intractable for large-scale networks. On the other hand, the algorithm proposed here is online and more computationally efficient, since the network is divided into smaller teams and the sampling-based algorithm is performed on the joint space of robots in each team. The authors in Hollinger et al. (2015) propose an informative path planning algorithm and a set of data fusion methods for mobile robot networks that can disconnect to search for mobile targets. The proposed fusion methods are computationally efficient since they avoid overcounting information and they ensure that combining data from robots that have been disconnected for a long time will not decrease the performance of the path planning algorithm. Nevertheless, they require a discretization of the environment which is not the case in this chapter. Also, in Hollinger et al. (2015), robots broadcast their information only when they arrive at a replanning location. Similarly, in Kemna
et al. (2017) a method to learn a spatial field through informative adaptive sampling is proposed for underwater vehicles that requires the robots to occasionally surface in order to communicate and coordinate their actions. On the other hand, the algorithm proposed here decomposes the network into smaller teams and determines online informative paths that lead to connected configurations that can change with time for each team by taking into account the objective to explore the environment and gather information. These results have been reported in Khodayi-mehr et al. (2017)

1.3.4 Chapter 5

In this chapter, we develop a distributed and online intermittent communication framework for teams of mobile robots that are responsible for accomplishing high-level, time-critical, dynamic tasks while sharing the collected information with all other robots and possibly also with a user. The assigned tasks are dynamic, i.e., the task specifications can change over time, which can be the case in unknown or dynamic environments; see, e.g., Guo and Dimarogonas (2015); Alonso-Mora et al. (2017). Moreover, the assigned tasks are time-critical in the sense that the information collected by the robots as they navigate the workspace is time-critical and, as result, they should not hold onto the gathered data for a long time. Instead, they have to communicate with other robots frequently enough, according to desired specifications. Moreover, we assume that the robots have limited communication capabilities that are not sufficient to maintain reliable and connected networks while they move to accomplish their tasks. Therefore, we assume that the robots can only communicate with each other when they meet at common locations in space. To define a communication network, we decompose the network into smaller teams of robots, as discussed in Chapter 2, that are fixed with time. Then, we require robots in every team to meet infinitely often at communication locations, that are common
for each team, but possibly different across teams giving rise to an intermittently connected dynamic communication network, as in Chapter 3. Then, our goal is to synthesize paths, i.e., sequences of waypoints, for every robot that ensure that the assigned time-critical dynamic tasks are satisfied, the communication network is connected over time infinitely often, i.e., robots in every team communicate infinitely often at one of the available communication locations, and a user-specified metric, such as total traveled distance, is minimized.

To address this problem, we propose a novel integration of the communication schedules that are designed offline, as discussed in Chapter 2, with task-oriented paths. Specifically, we design online new paths by introducing communication points in the task-oriented paths as per the designed schedules so that the assigned tasks are satisfied, the communication network is connected over time infinitely often, the robots share the time-critical information frequently enough according to the desired specifications, and a user specified metric is minimized. These paths are designed by solving local integer programs. We show that the proposed framework can be applied to several robot tasks accomplished by finite, infinite, periodic, or aperiodic paths. Tasks that are accomplished by periodic (infinite) paths include surveillance Nardi et al. (2016), informative path planning Lan and Schwager (2016), or temporal logic tasks, such as coverage Fainekos et al. (2005) and data gathering Guo and Zavlanos (2017). Infinite and aperiodic tasks include reactive motion planning in unknown or dynamic environments as, e.g., in Guo and Dimarogonas (2015); Alonso-Mora et al. (2017), or receding horizon planning approaches Hollinger and Singh (2010); Ulusoy and Belta (2014). Finite tasks comprise point-to-point navigation tasks Karaman and Frazzoli (2011); Kantaros and Zavlanos (2016c); Stephan et al. (2017) or co-safe temporal logic tasks Bhatia et al. (2010), such as sequencing tasks Fainekos et al. (2005) that require visitation of regions of interest in a specific order. These results have been reported in Kantaros and Zavlanos (2018b).
In this chapter, we develop a distributed, online, and fully autonomous intermittent communication framework for teams of mobile robots that are responsible for accomplishing arbitrary dynamic tasks while sharing the collected information with all other robots and possibly also with a user. Similar to the previous chapters, we assume that the robots have limited communication capabilities and can only communicate with each other when they meet at common locations in space. To define a communication network, we decompose the network into smaller teams of robots that, unlike the previous chapters, can change with time. Then, we require robots in every team to meet infinitely often at communication locations, that are common for each team, but possibly different across teams giving rise to an intermittently connected dynamic communication network, as in Chapter 3. Our goal in this paper is to construct optimal teams and select optimal respective communication locations for every team, design paths that accomplish the assigned dynamic tasks and ensure that the dynamic communication network is connected over time infinitely often so that a user-specified metric, such as total traveled distance, is minimized.

To solve this problem, we propose a distributed control framework that decouples the task planning and the intermittent communication problem. Specifically, first we design schedules of communication events for all robots, independently of the assigned task, that ensure that the communication graph is connected over time infinitely often. These schedules are initialized given the initial user-specified teams and are updated locally and online as the robots design new teams. Second, we integrate the schedules of communication events with the dynamic tasks giving rise to paths, i.e., sequence of waypoints, that the robots have to follow. The teams and the associated communication points, the time instants that the teams communicate, and the resulting paths are updated online so that the assigned task is satisfied, the
communication network is connected over time infinitely often, and a user-specified metric, such as total traveled distance or consumed energy, is minimized. These results have been reported in Kantaros and Zavlanos (2018a).

1.4 Contributions

The contributions of this dissertation for each chapter can be summarized as follows:

1.4.1 Contributions of Chapter 2

- We propose the first distributed intermittent communication framework that ensures connectivity over time, infinitely often and does not require that the communication network is ever connected at once nor imposes restrictions on the communication pattern that needs to be achieved, as opposed to the relevant works discussed in Section 1.2.3. Specifically, the proposed framework determines when the robots should communicate and allows them to move free of communication constraints the rest of the time to accomplish their tasks.

- The proposed intermittent connectivity framework is agnostic to the assigned task and is flexible enough to account for arbitrary dynamic tasks, such as consensus, complex high-level tasks (e.g., surveillance, coverage, sequencing, and data gathering) captured by temporal logic specifications, state estimation, and time-critical dynamic tasks,

- The proposed intermittent connectivity framework scales with the number of teams and not with the size of the network. Therefore, it can be applied to large-scale robot networks.

- We provide theoretical guarantees that show that that the dynamic communication network is connected over time infinitely often.
1.4.2 Contributions of Chapter 3

- We propose the first decentralized framework for concurrent temporal task planning and intermittent connectivity control for mobile robot networks that can determine both when and where the robots should communicate.

- We show that synthesized controllers can be executed in an asynchronous fashion, which is not the case in most relevant literature in control synthesis as, e.g., in Kloetzer and Belta (2010); Pola et al. (2016).

- We provide theoretical guarantees about the convergence, correctness, and optimality of the proposed decentralized algorithm.

- We provide numerical studies and experiments showing that the proposed algorithm can solve very large and complex planning problems that relevant centralized optimal control synthesis algorithms or off-the-shelf model checkers Cimatti et al. (2002); Holzmann (2004) cannot solve.

1.4.3 Contributions of Chapter 4

- We propose the first distributed framework for concurrent state estimation and intermittent connectivity control for mobile robot networks that can determine both when and where the robots should communicate.

- We show that the proposed sampling-based solution to the informative path planning problem is probabilistically complete and guarantees that the robots form a connected subnetwork at the end at the same time.

- Moreover, we characterize the delay in propagating information across the network caused due to intermittent communication.

- We show through comparative simulation studies that state estimation for target tracking using intermittent communication outperforms methods that
maintain network connectivity for all time. We also present simulation studies that show the effect of the structure of the teams in the propagation of delays and in the estimation performance of our algorithm.

1.4.4 Contributions of Chapter 5

- We propose the first, distributed, and online intermittent connectivity control framework for networks of mobile robots under high-level, dynamic, time-critical tasks.

- We provide theoretical results that support the proposed framework.

- We provide simulation studies illustrating its efficiency and capability to account for arbitrary dynamic tasks, such as simple point-to-point navigation tasks, temporal task planning, estimation tasks, and reactive path planning.

1.4.5 Contributions of Chapter 6

- We propose the first, distributed, online, and fully autonomous intermittent connectivity control framework for networks of mobile robots under arbitrary dynamic tasks. The proposed framework allows the robot to locally and online decide both when and where they should communicate, and which robots should talk to which. All these decisions are integrated online with robot paths designed online to accomplish the assigned tasks.

- We provide theoretical results that corroborate the efficiency of the the proposed framework.

- Finally, as in Chapter 5, we provide simulation studies illustrating the efficiency and capability of the proposed control scheme to account for arbitrary dynamic robot tasks, such as simple point-to-point navigation tasks, temporal task planning, estimation tasks, and reactive path planning.
Distributed Intermittent Connectivity Control of Mobile Robot Networks

Many coordinated tasks performed by teams of mobile robots e.g., area coverage and exploration, environmental monitoring, and synchronization, critically depend on the ability of the robots to exchange information with each other through a communication network that is connected either for all time, or intermittently but infinitely often. This requirement for network connectivity has recently led to a broad range of techniques to control it. Typically these methods focus on all-time connectivity, while intermittent connectivity is still widely considered an assumption. In this chapter, we design distributed discrete controllers that ensure that the dynamic communication network is connected over time infinitely often. The proposed controllers scale well with the size of the network and their construction is agnostic to the assigned task.
2.1 Problem Formulation

Consider \( N > 1 \) mobile robots operating in a workspace \( \mathcal{W} \subset \mathbb{R}^d \), \( d = 2, 3 \), and let
\[
\dot{x}_i(t) = f_i(x_i(t), u_i(t)),
\]
de note the equations of motion of robot \( i \), where \( x_i(t) \in \mathbb{R}^d \) and \( u_i(t) \in \mathbb{R}^d \) are the position and control input of robot \( i \), respectively, at time \( t \geq 0 \). Let \( \mathcal{N} = \{1, \ldots, N\} \) denote the set of all robots. Also, the robots possibly have to accomplish a task \( \Gamma(x(t)) \), where \( x(t) \) is a vector that stacks the positions of all robots at time \( t \geq 0 \).

As the robots navigate the workspace to accomplish the assigned task they collect data that need to be shared with all other robots and possibly with a user. The shared data can possibly be used to design robot paths associated with the assigned task.

To define a communication network among the robots, we first partition the robot team into \( M \geq 1 \) robot subgroups, called also teams, and require that every robot belongs to at least one subgroup. The indices \( i \) of the robots that belong to the \( m \)-th subgroup are collected in a set denoted by \( \mathcal{T}_m \), for all \( m \in \mathcal{M} := \{1, 2, \ldots, M\} \). We define the set that collects the indices of teams that robot \( i \) belongs to as \( \mathcal{M}_i = \{m|i \in \mathcal{T}_m, m \in \mathcal{M}\} \).

Also, for robot \( i \) we define the set that collects the indices of all other robots that belong to common teams with robot \( i \) as \( \mathcal{N}_i = \{j|j \in \mathcal{T}_m, \forall m \in \mathcal{M}_i\} \setminus \{i\}, \forall i \in \mathcal{N} \). Given the robot teams \( \mathcal{T}_m \), for all \( m \in \mathcal{M} \), we can define the graph over these teams as follows.

**Definition 2.1.1** (Team Membership Graph \( \mathcal{G}_\mathcal{T} \)). The graph over the teams \( \mathcal{T}_m \), \( m \in \mathcal{M} \) is defined as \( \mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T}) \), where the set of nodes \( \mathcal{V}_\mathcal{T} = \mathcal{M} \) is indexed by the teams \( \mathcal{T}_m \) and set of edges \( \mathcal{E}_\mathcal{T} \) is defined as \( \mathcal{E}_\mathcal{T} = \{(m, n)|\mathcal{T}_m \cap \mathcal{T}_n \neq \emptyset, \forall m, n \in \mathcal{M}, m \neq n\} \).

Given the team membership graph \( \mathcal{G}_\mathcal{T} \), we can also define the set
\[
\mathcal{N}_{\mathcal{T}_m} := \{e \in \mathcal{V}_\mathcal{T}|(m, e) \in \mathcal{E}_\mathcal{T}\},
\]
(2.2)
that collects all neighboring teams of team $\mathcal{T}_m$ in $\mathcal{G}_T$. Since the robots have limited communication capabilities compared to the size of the workspace they live in, we assume that the robots in every subgroup $\mathcal{T}_m$ can only communicate if all of them are physically close to each other. When all robots in a team $\mathcal{T}_m$ are close to each other, we assume that communication in team $\mathcal{T}_m$ happens and the robots leave to accomplish their tasks or communicate with other teams. This way, a dynamic robot communication network is constructed, defined as follows; see also Figure 2.1.

**Definition 2.1.2 (Communication Network $\mathcal{G}_c$).** The communication network among the robots is defined as a dynamic undirected graph $\mathcal{G}_c(t) = (\mathcal{V}_c, \mathcal{E}_c(t))$, where the set of nodes $\mathcal{V}_c$ is indexed by the robots, i.e., $\mathcal{V}_c = \mathcal{N}$, and $\mathcal{E}_c(t) \subseteq \mathcal{V}_c \times \mathcal{V}_c$ is the set of communication links that emerge among robots in every team $\mathcal{T}_m$, when all robots in $\mathcal{T}_m$ are physically close to each other constructing a connected sub-network, given their communication capabilities.

To ensure that information is continuously transmitted across the network of robots, we require that the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often. For this, it is necessary to assume that the graph of teams $\mathcal{G}_T$ is connected. Specifically, if $\mathcal{G}_T$ is connected, then information can be propagated intermittently across teams through robots that are common to these teams and, in this way, information can reach all robots in the network. Connectivity of $\mathcal{G}_T$ and the fact that robots can be members of only a few teams means that information can be transferred over long distances, possibly to reach a remote user, without requiring that the robots leave their assigned regions of interest defined by their assigned tasks and communication points corresponding to the teams they belong to. Moreover, we assume that the teams $\mathcal{T}_m$ are a priori known and can be selected arbitrarily as long as the graph of teams $\mathcal{G}_T$ is connected.

The problem that is addressed in this chapter can be stated as:
Problem 2.1.3. Compute schedules of communication events that determine the order in which every robot $i$ should communicate with teams $\mathcal{T}_m$, for all $m \in \mathcal{M}_i$, so that the dynamic communication graph $\mathcal{G}_c(t)$, defined in Definition 2.1.2, is connected over time infinitely often.

To solve Problem 2.1.3, in Section 2.2, we construct offline and in a decentralized way communication schedules that determine when every robot $i$ should communicate with the robots in teams $\mathcal{T}_m$, for all $m \in \mathcal{M}_i$. In Section 2.3, we describe how the robots execute these communication schedules online, i.e., as they navigate the workspace to accomplish their tasks.

Remark 2.1.4 (Transportation networks). Although the proposed framework is developed and motivated with the objective of ensuring intermittent communication within teams of robots in a distributed way, other applications are also possible. For example, instead of collecting sensor data, the robots can collect and transfer sup-
plies from designated depots that fall along their motion paths. Then, during meeting
events, the robots can deliver supplies from one another, giving rise to a distributed
transportation system.

2.2 Construction of Communication Schedules

In this section we construct infinite sequences of communication events (also called
communication schedules) so that the communication network $\mathcal{G}_c(t)$, defined in Def-
nition 2.1.2, is connected over time infinitely often. Construction of the communi-
cation schedules occurs in a decentralized way and offline i.e., before the robots are
deployed in the workspace to accomplish their tasks, and requires that the robots
are connected so that they can share information with each other.

Since every robot can be a member of more than one team, the objective in
designing these schedules is that no teams that share common robots communicate
at the same time, as this would require that the shared robots are present at more
than one possibly different locations at the same time. We call such schedules conflict-
free. To construct such conflict-free schedules of communication events we rely on
a sequence $S$ of teams that defines the order in which the robots construct their
schedules.

**Definition 2.2.1 (Sequence S).** The finite sequence $S$ is a sequence of teams defined
as $S = T_n, T_m, \ldots$. The sequence $S$ satisfies two requirements: (i) all teams $T_m,
m \in \mathcal{M}$ appear in $S$; and (ii) consecutive teams $T_n$ and $T_m$ that appear in $S$ are
neighboring nodes in the graph $\mathcal{G}_T$, i.e., $m \in \mathcal{N}_{T_n} := \{ e \in \mathcal{E}_T | (n, e) \in \mathcal{E}_T \}$.

In what follows, we assume that the sequence $S$ is a user-defined input to the
system that is known by all robots. Moreover, we denote by $S(k)$ the $k$-th team
in $S$, $\forall k \in \{1, \ldots, |S|\}$ where $|S|$ stands for the length of $S$. Using the sequence $S$
we construct communication schedules for all robots $i$, denoted by $\text{sched}_i$. In what
follows, we define communication schedules that determine the order in which the robots in every team \( T_m \) for \( m \in M_i \) should communicate with each other.

**Definition 2.2.2 (Schedule of Communication Events).** The schedule of communication events of robot \( i \), denoted by \( \text{sched}_i \), is defined as an infinite repetition of the finite sequence

\[
s_i = X, \ldots, X, M_i(1), X, \ldots, X, M_i(2), X, \ldots, X,
\]

\[
M_i(|M_i|), X, \ldots, X,
\]

(2.3)

i.e., \( \text{sched}_i = s_i, s_i, \cdots = s_\omega \), where \( \omega \) stands for the infinite repetition of \( s_i \).

In (2.3), \( M_i(e), e \in \{1, \ldots, |M_i|\} \) stands for the \( e \)-th entry of \( M_i \) and represents a communication event for team with index \( M_i(e) \), and the discrete states \( X \) indicate that there is no communication event for robot \( i \). The length of the sequence \( s_i \) is

\[ \ell = \max \{d_{T_m} \}_{m=1}^M + 1 \] for all \( i \in \mathcal{N} \), where \( d_{T_m} \) is the degree of node \( m \in \mathcal{V}_T \). It is shown in Proposition 2.4.1 that this length \( \ell \) is sufficient for the construction of conflict-free communication schedules as per the algorithm described bellow. The schedule \( \text{sched}_i \) defines the order in which robot \( i \) participates in the communication events for the teams \( m \in M_i \), for all robots \( i \in \mathcal{N} \). Specifically, robot \( i \) either communicates with all robots that belong to team \( T_m \), for \( m \in M_i \) if \( \text{sched}_i(z_i) = m \), or does not need to participate in any communication event if \( \text{sched}_i(z_i) = X \), where \( \text{sched}_i(z_i) \) stands for the entry in \( \text{sched}_i \) with index \( z_i \in \mathbb{N}_+ \).

In what follows we present a decentralized process that relies on two rules that the robots execute in order to construct the schedules \( \text{sched}_i \). These schedules are constructed sequentially across the teams \( T_m, m \in \mathcal{M} \), in an order that is determined by the sequence \( S \). In other words, robots in team \( S(k) \) will construct their respective schedules, only if all robots in team \( S(k - 1) \) have already designed their schedules. Assume that according to the sequence \( S \), robots in team \( S(k) = T_m \), for some \( k \geq 1 \)
are due to construct their schedules. By construction of the sequence $S$, consecutive teams in $S$ are always neighboring teams, which means that there exists a team $T_n$ with $n \in \mathcal{N}_{T_m}$ such that $S(k-1) = T_n$ and $T_m \cap T_n \neq \emptyset$. Consequently, there exist also robots $j \in T_m \cap T_n$ that previously constructed their sequences $s_j$. These robots $j$ never re-construct their schedules. Instead, one of the robots $j \in S(k) \cap S(k-1)$ is tasked with providing information to the other robots $i \in S(k) = T_m$ that is necessary to construct their sequences $s_i$.

Specifically, this robot $j \in S(k) \cap S(k-1)$ first notifies the robots in team $S(k) = T_m$ that it is their turn to construct their communication schedules.\(^1\) Second, robot $j$ transmits to robots $i \in T_m$ all sequences $s_b$ that were have been constructed so far by the robots in teams $S(1), \ldots, S(k-1)$. Among all those sequences $s_b$, robots $i \in T_m$ use only the sequences of robots $b \in \mathcal{N}_i$ to construct their sequences $s_i$.\(^2\) As a result, all robots $i \in T_m$ that have not constructed $s_i$ yet, are aware of the indices $n_b^{T_m}$ that point to entries in $s_b$ associated with some communication events $g$. These indices satisfy $s_b(n_b^{T_m}) = g$, $b \in \mathcal{N}_i$.\(^3\) Notice that this means that robots $i \in T_m$ are also aware of the indices $n_b^{T_m}$. Using this information, every robot $i \in T_m$ constructs the sequence $s_i$ based on the following two rules that determine the indices $n_i^{T_m}$ that point to entries in $s_i$ where the communication event $g$ will be placed, i.e., $s_i(n_i^{T_m}) = g$, for all $g \in \mathcal{M}_i$.

1. **First rule:** Let $n_i^{T_m}$ denote the index of the entry at which the communication event $g \in \mathcal{M}_i$ will be placed into $s_i$. If there exists a robot $b \in \mathcal{N}_i$ that has

\(^1\) Note that if the teams in $S$ were not necessarily neighboring teams, then robot $j \in S(k-1) = T_n$ would have to know who the members of team $S(k) = T_m$, $m \notin \mathcal{M}_j$, are in order to notify them that its their turn to construct the communication schedules. Due to the fact that $S$ connects neighboring teams, every robot $j$ needs to know only the structure of teams $T_m$, $m \in \mathcal{M}_j$.

\(^2\) Note that robot $j$ is not aware of the sets $\mathcal{N}_i$ and, therefore, it transmits all the sequences $s_b$ that have already been constructed to robots $i \in T_m$.

\(^3\) Note that the indices of entries in $\text{sched}_i$, where the communication event $g \in \mathcal{M}_i$ is placed are $n_i^{T_m} + z\ell$, where $z \in \mathbb{N}_+$, by definition of $\text{sched}_i$. 
selected \( n^T_g \) so that \( s_b(n^T_g) = g \), then \( n^T_g = n^T_b \). In this way, all robots \( b \in T_g \), including robot \( i \in T_m \cap T_g \) will select the same index \( n^T_g \) for the communication event \( g \); see line 4, Alg. 1.

2. Second rule: If there do not exist robots \( b \in N_i \) that have selected indices \( n^T_g \), for communication event \( g \in M_i \), then the communication event \( g \) can be placed at any available entry \( n^T_i \) of \( s_i \) that satisfies the following requirement. The entry \( n^T_i \) in all sequences \( s_j \) of robots \( j \in N_i \) that have already been constructed should not contain communication events \( h \) such that \( h \in N_{T_g} \); see line 6, Alg. 1.

Note that the index \( n^T_m \) will always be determined by the first rule, since robot \( j \in S(k) \cap S(k - 1) \) has already constructed its sequence \( s_j \) by placing the event \( m \) at an entry of \( s_j \) with index \( n^T_m \). To highlight the role of the second rule assume that \( h \in N_{T_g} \). Then, this means that there exists at least one robot \( r \in T_h \cap T_g \). Notice that without the second rule, at a subsequent iteration of this procedure, robot \( r \in T_h \cap T_g \) would have to place communication events for teams \( T_g \) and \( T_h \) at a common entry of \( s_r \), i.e., \( n^T_r = n^T_r \), due to the first rule and, therefore, a conflicting communication event in schedule \( \text{sched} \) would occur. In all the remaining entries of \( s_i \), \( X \)'s are placed; see line 7, Alg. 1. By construction of \( s_i \), there are \( \ell - |M_i| \) \( X \)'s in \( s_i \).

Once all robots \( i \) in team \( S(k) \) have constructed the sequences \( s_i \), a robot \( j \in S(k) \cap S(k + 1) \) will notify all robots in team \( S(k + 1) \) that its their turn to compute their respective schedules. The procedure is repeated sequentially over the teams in \( S \) until all robots have computed their respective schedules of meeting events. Execution of Algorithm 1 is also illustrated in Example 2.2.5.

Remark 2.2.3 (Initialization of Algorithm 1). At the first iteration of Algorithm 1, robots of team \( S(1) \) have to construct their sequences \( s_i \), \( i \in S(1) \). In this case,
Algorithm 1: Decentralized construction of sequence $s_i$, $i \in \mathcal{T}_m$

**Input:** Already constructed sequences $s_b$, $\forall b \in \mathcal{N}_i$.

**Output:** Schedule of meeting events: $\text{sched}_i = [s_i]_{\ell}^\omega$

1. Construct an empty finite sequence $s_i$ of length $\ell$.
2. for $g \in \mathcal{M}_i$ do
   3. if there exist constructed sequences $s_b$, $b \in \mathcal{T}_g$ then
      4. $s_i(n_{i}^{T_g}) := g$, where $n_{i}^{T_g} := n_{b}^{T_g}$, $\forall b \in \mathcal{T}_g$; $\triangleright$ First rule
   5. else
      6. Choose an available $n_{i}^{T_g} \in \{1, \ldots, \ell\}$ such that it holds either
         $s_j(n_{i}^{T_g}) := X$, or $s_j(n_{i}^{T_g}) := h$ with $h \notin \mathcal{N}_{T_g}$, $\forall j \in \mathcal{N}_i$. Then set
         $s_i(n_{i}^{T_g}) := g$. $\triangleright$ Second rule
7. Put $X$ in the remaining entries;

---

**Figure 2.2:** Team membership graph $G_T$ for Example 2.2.5.

---

A randomly selected robot $j \in S(1)$ creates arbitrarily its sequence $s_j$ by placing the teams $m \in \mathcal{M}_j$ at the $n_{j}^{T_m}$-th entry of $s_j$. Then the procedure described in Algorithm 1 follows.

**Remark 2.2.4** (Sequence $S$ and Algorithm 1). Due to the requirement that consecutive teams in $S$ need to be neighbors in $G_T$, it is possible that a team $\mathcal{T}_m$ may appear more than once in $S$, depending on the structure of the graph $G_T$. In this case, robots $i \in \mathcal{T}_m$ construct the sequences $s_i$ only the first time that team $\mathcal{T}_m$ appears in $S$.

**Example 2.2.5** (Algorithm 1). To illustrate Algorithm 1, consider the network of $N = 3$ robots, where the teams of robots are designed as $\mathcal{T}_1 = \{1, 2\}$, $\mathcal{T}_2 = \{2, 3\}$, and
$T_3 = \{3,1\}$ giving rise to the team membership graph shown in Figure 2.2. Let the sequence $S$ be $S = T_1, T_2, T_3$. Hence, initially the robots 1 and 2 in team $T_1$ coordinate to construct their respective sequences $s_1$. Assume that initially robot 1 constructs the sequence $s_1$ of length equal to $\ell = \max \{d_{T_m}\}_{m=1}^3 + 1 = 3$. Robot 1 belongs to teams $T_1$ and $T_2$ and it arbitrarily constructs $s_1$ as follows: $s_1 = 1,3,X$. Then the sequence $s_1$ is transmitted to robot 2 that belongs to teams $T_1$ and $T_2$. Now robot 2 is responsible for constructing the sequence $s_2$. To construct $s_2$, according to the first rule, team $T_1$ is placed at the first entry of $s_2$, i.e., $n_{T_1}^{T_2} = n_{T_1}^{T_1} = 1$. Next, the index $n_{T_2}^{T_2}$ is determined by the second rule. Specifically, notice that among the two available entries in $s_2$ for team $T_3$ the entry $n_{T_3}^{T_2} = 2$ is invalid, since robot 1 $\in T_1$ has already constructed its sequence $s_1$ so that $n_{T_1}^{T_3} = 2$ and for teams $T_3$ and $T_2$ it holds that $3 \in N_{T_2}$. Therefore, robot 2 selects $n_{T_2}^{T_2} = 2$ and constructs the sequence $s_2 = 1,X,2$. At the next iteration of Algorithm 1 the robots 2 and 3 in team $T_2$ coordinate to construct their sequences $s_1$. Robot 2 has already constructed the sequence $s_2$ at the previous iteration and it transmits its constructed sequence $s_2$ and the previously constructed sequence $s_1$ to robot 3. Thus robot 3 has now access to all already constructed sequences $s_e$, for $e \in N_3 = \{1,2\}$. Robot 3 constructs $s_3 = X,3,2$ using the first rule. Finally, the robots in the third team $T_3 = \{3,1\}$ have already constructed their finite paths at previous iterations.

Remark 2.2.6 (Discrete states $X$). In the schedules $sched_i$, defined in Definition 2.2.2 and constructed using Algorithm 1, the states $X$ indicate that no communication events are placed at these entries in $sched_i$. These states are used to synchronize the communication events over the indices $z_i \in N_+$, i.e., to ensure that the index $z_i$ of the entry in $sched_i$ at which the communication event for a team $T_m$ has been placed is the same for all robots $i \in T_m$; see also Example 2.2.5. Nevertheless, as it will be shown in Theorem 2.4.4, in Section 2.4, it is the order of communication events
Algorithm 2: Online Asynchronous and Local Execution of \( \text{sched}_i(t) \), \( i \in \mathcal{T}_m \)

\[
\begin{array}{l}
\text{for } z_i = 1 : \infty \text{ do} \\
\quad \text{if } \text{sched}_i(z_i) = m \text{ then} \\
\quad \quad \text{Wait until } \text{sched}_j(z_j) = m, \text{ for all } j \in \mathcal{T}_m; \\
\quad \quad \text{Communication within team } \mathcal{T}_m \text{ occurs;}
\quad z_i = z_i + 1; \\
\text{else if } \text{sched}_i(z_i) = X \text{ then} \\
\quad z_i = z_i + 1; \\
\text{Next communication event is } \text{sched}_i(z_i);
\end{array}
\]

in \( \text{sched}_i \) that is critical to ensure intermittent communication, not the indices of entries in \( \text{sched}_i \) at which these communication events have been placed. This is due to communication policy proposed in Section 2.3.

2.3 Online Asynchronous Execution of Communication Schedules

In this section, we describe the online execution of the schedules \( \text{sched}_i \), i.e., how the robots locally determine the next communication event they need to participate using their respective schedules \( \text{sched}_i \), as they navigate the workspace to accomplish their assigned tasks. Note that the communication schedules \( \text{sched}_i \) are executed asynchronously among the robots. The execution of \( \text{sched}_i \) is described in Algorithm 2. The index \( z_i \) is initialized as \( z_i = 1 \), for all robots \( i \in \mathcal{N} \) [line 1, Alg. 2] and is updated so that it points to the next event in \( \text{sched}_i \) that robot \( i \) needs to participate. Specifically, assume that the current the event for robot \( i \) is \( \text{sched}_i(z_i) \), where \( \text{sched}_i(z_i) \) stands for the \( z_i \)-th entry in the infinite \( \text{sched}_i \). If \( \text{sched}_i(z_i) = m, m \in \mathcal{M}_i \), then robot \( i \) needs to communicate with robots in team \( \mathcal{T}_m \) and performs the following control policy [lines 2-3, Alg. 2]:

**Definition 2.3.1** (Communication Control Policy). If \( \text{sched}_i(z_i) = m, m \in \mathcal{M}_i \), then robot \( i \) waits (i.e., does not update the index \( z_i \)), indefinitely or until \( \text{sched}_j(z_j) = m, \text{ for all } j \in \mathcal{T}_m \).
Once $\text{sched}_j(z_j) = m$, for all $j \in T_m$, then communication in team $T_m$ occurs [line 4, Alg. 2]. In case $\text{sched}_i(z_i) = X$, then robot $i$ does not need to participate in any communication event. In both cases the index $z_i$ is updated as $z_i = z_i + 1$ [lines 5 & 7, Alg. 2] and the next communication event for robot $i$ is is $\text{sched}_i(z_i)$. This process is executed indefinitely over $z_i \in \mathbb{N}_+$ [line 1, Alg. 2].

**Remark 2.3.2 (Paths and Communication Locations).** Note that the designed communication schedules do not determine either the locations where communication in teams $T_m$ should occur or the paths that the robots should follow to communicate with other robots. Instead, they only determine the order at which robots should participate at communication events. Design of robot paths and communication locations depend on the assigned task $\Gamma(x(t))$ and is discussed in the following chapters.

### 2.4 Correctness

**Proposition 2.4.1 (Conflict-free schedules).** Algorithm 1 can always construct sequences $s_i$, for all $i \in \mathcal{N}$, if the length $\ell$ of $s_i$ is selected as $\ell = \max\{d_{T_m}\}_{m=1}^M + 1$.

**Proof.** The proof is based on contradiction. Assume that a robot $i$ requires a sequence $s_i$ of length greater than $\ell = \max\{d_{T_m}\}_{m=1}^M + 1$ when Algorithm 1 is applied. This means that there is team $T_m$, $m \in M_i$, which cannot be placed at any of the first $\ell$ entries of $s_i$. By construction of Algorithm 1, this means that the team $T_m$ has at least $\ell$ neighbors in graph $\mathcal{G}_T$, i.e., $d_{T_m} \geq \ell$, which can never happen, which completes the proof. 

Next, to show that the communication network $\mathcal{G}_e(t)$ is connected over time infinitely often we first need to show that the network is deadlock-free when the communication schedules $\text{sched}_i$ are executed according to Algorithm 2. Specifically, we assume that there is a deadlock, if the current communication event for some
robots \( i \in \mathcal{T}_m \) is \( \text{sched}_i(z_i) = m \) and are waiting forever until \( \text{sched}_j(z_j) = m \) for the other robots \( j \in \mathcal{T}_m \) due to the control policy defined in Definition 2.3.1.

**Proposition 2.4.2** (Deadlock-free schedules). The mobile robot network is deadlock-free when the communication schedules \( \text{sched}_i \) generated by Algorithm 1 are executed according to Algorithm 2.

**Proof.** Let \( \mathcal{W}_m \subset \mathcal{T}_m \) denote the set of robots \( i \) that satisfy \( \text{sched}_i(z_i) = m \) and are waiting until \( \text{sched}_j(z_j) = m \) for the other robots \( j \in \mathcal{T}_m \setminus \mathcal{W}_m \) due to the control policy defined in Definition 2.3.1. Assume that the robots \( j \in \mathcal{T}_m \setminus \mathcal{W}_m \) never satisfy \( \text{sched}_j(z_j) = m \). This means that the robots in \( j \in \mathcal{T}_m \setminus \mathcal{W}_m \) are waiting indefinitely to communicate with robots in team \( \mathcal{T}_n \). The fact that there are robots that wait indefinitely means that a communication within team \( \mathcal{T}_n \) never occurs by construction of Algorithm 2. Following an argument similar to the above, we conclude that the robots in \( \mathcal{T}_n \setminus \mathcal{W}_n \) are waiting indefinitely to communicate with robots that belong to a team \( \mathcal{T}_f \), \( f \in \mathcal{N}_{\mathcal{T}_n} \). Therefore, if a communication event never occurs for team \( \mathcal{T}_m \), then all robots \( i \in \mathcal{N} \) need to be waiting forever. Consequently, there is no team within which communication will ever occur. Throughout the rest of the proof we will refer to this network configuration as a *stationary configuration*.

In what follows, we show by contradiction that the network can never reach a stationary configuration when the communication schedules \( \text{sched}_i \) are executed asynchronously as per Algorithm 2. Specifically, we will show that if the network gets trapped at a stationary configuration during the asynchronous execution of the schedules \( \text{sched}_i \), then this means that there exist robots of some team \( \mathcal{T}_m \) that missed a communication event for a team \( \mathcal{T}_m \) at a previous time instant, which cannot happen by construction of Algorithm 2. Consider that there is an arbitrary time instant \( t_0 \) at which the network is at a stationary configuration. Then, this means that the current communication event for all robots \( i \) is \( \text{sched}_i(n^T_m(t_0)) = m \).
for some $m \in \mathcal{M}_i$, where the indices $n_i^{T_m}$ were defined in Algorithm 1. Define also the set $\mathcal{N}_{\min}(t_0) = \left\{ n_i^{T_m}(t_0) \mid n_i^{T_m}(t_0) = \min_{e=1}^{N} \{ n_e^{T_m}(t_0) \} , g \in \mathcal{M}_e \right\}$ that collects the minimum indices $n_i^{T_m}(t_0)$ among all robots. Also let $n_e^{T_g}(t_0)$ be an index such that $n_e^{T_g}(t_0) \in \mathcal{N}_{\min}(t_0)$. By assumption there are robots $e \in \mathcal{T}_g$ and $r \in \mathcal{T}_z$, $g \in \mathcal{N}_{T_z}$, such that $e \in \mathcal{W}_g(t_0)$ and $r \in \mathcal{W}_z(t_0)$ and, therefore, the events that are taking place for these two robots according to their assigned schedules of meeting events are $\text{sched}_e(n_e^{T_g}(t_0)) = g$ and $\text{sched}_r(n_r^{T_z}(t_0)) = z$. Since $n_e^{T_g}(t_0) \in \mathcal{N}_{\min}(t_0)$ we have that $n_e^{T_g}(t_0) \geq n_r^{T_z}(t_0)$, which along with the fact that $g \in \mathcal{N}_{T_z}$ results in $n_e^{T_g}(t_0) > n_r^{T_z}(t_0)$ by construction of Algorithm 1. This leads to the following contradiction. The fact that $n_e^{T_g}(t_0) > n_r^{T_z}(t_0)$ means that there exists a time instant $t < t_0$ at which the event that took place for robots $a \in \mathcal{T}_g \cap \mathcal{T}_z$ was $\text{sched}_a(n_a^{T_z}(t)) = g$ and at least one of these robots did not wait for the arrival of all other robots in team $\mathcal{T}_g$, since at the current time instant $t_0$ there are still robots in team $\mathcal{T}_g$ waiting for the arrival of other robots. However, such a scenario is precluded by construction of Algorithm 2. Consequently, the network is deadlock-free when all robots $i$ execute their respective schedules $\text{sched}_i$, asynchronously, as per Algorithm 2, which completes the proof.  

Remark 2.4.3 (Bounded waiting times). Proposition 2.4.2 shows also that the waiting times of robots at the communication points that were introduced by Algorithm 4 are bounded.

Theorem 2.4.4 (Intermittent Communication). The asynchronous execution of the communication schedules $\text{sched}_i$ as per Algorithm 2 ensures that the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often.

Proof. To show that the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often, it suffices to show that the communication within every team $\mathcal{T}_m$ occurs infinitely often, since the graph $\mathcal{G}_\mathcal{T}$ is connected by assumption. Due to Proposition
it holds that the waiting times of robots induced by the communication control policy 2.3.1 are bounded. Also, since by construction of Algorithm 1, the schedules \(\text{sched}_i\) contain communication events for all teams \(\mathcal{T}_m, m \in \mathcal{M}\); and the schedules \(\text{sched}_i\) are executed indefinitely, communication within any team \(\mathcal{T}_m\) occurs infinitely often, which completes the proof.

**Remark 2.4.5** (Execution of Communication Schedules). Theorem 2.4.4 also implies that it is the order of communication events that is critical to ensure intermittent communication, not the time instants that they take place. In other words, the discrete states \(X\) can be completely ignored during the execution of the communication schedules due to the control policy defined in Definition 2.3.1. Moreover, Proposition 2.4.2 and Theorem 2.4.4 imply that the execution of the communication schedules as per Algorithm 2 is robust to uncertainty in traveling times between consecutive communication events due to the control policy defined in Definition 2.3.1.

### 2.5 Numerical Experiments

In this section, a simulation study is provided that illustrates our approach for a network of \(N = 42\) robots that is partitioned into 32 teams so that the resulting graph \(\mathcal{G}_T\) is connected. Also, we assign to each team \(\mathcal{T}_m\) a single pre-determined location, denoted by \(v_m\), where the robots in \(\mathcal{T}_m\) should be physically present to communicate. Moreover, for simplicity we assume that there are paths that are straight lines that connect these communication locations giving rise to a mobility graph, as shown in Figure 2.3. The teams are designed so that every robot belongs to two teams with communication points that are connected through a straight line in the mobility graph. Hereafter, we denote by \(r_{mn}\) a robot that belongs to teams \(\mathcal{T}_m\) and \(\mathcal{T}_n\). To model uncertainty in robot mobility, we assume that values for the travel times \(T_{mn}\) are generated by a uniform distribution \(U(1,4)\) every time robot
Figure 2.3: Intermittent connectivity of $N = 42$ robots moving along the edges of an underlying mobility graph. Black squares represent communication points and red circles stand for robots. Red arrows show the direction along which robots are moving. Robots without arrows are waiting at the meeting points for the arrival of robots.

$r_{mn}$ reaches the communication point $v_m$. The same holds for travel times $T_{nm}$, as well. Therefore, the controller $u_{mn}$ that drives robot $r_{mn}$ from $v_m$ to $v_n$ is given by:

$$u_{mn} = \frac{1}{T_{mn}} \frac{v_m - v_n}{\|v_m - v_n\|}.$$  \hspace{1cm} (2.4)

The schedules constructed by Algorithm 1 for robots $r_{12}$, $r_{13}$, $r_{23}$ have the following structure

$$\tau_{12} = [1, 2, X]^\omega,$$ \hspace{1cm} (2.5)

$$\tau_{13} = [1, X, 3]^\omega,$$ \hspace{1cm} (2.6)

$$\tau_{23} = [X, 2, 3]^\omega.$$ \hspace{1cm} (2.7)

To illustrate that under the proposed motion plans connectivity is ensured over time, we implement a consensus algorithm over the dynamic network $G_c(t)$. Specifi-
cally, we assume that initially robots generate a random number $u_{mn}(t_0)$ and when all robots $r_{mn}$ meet at $v_m$ they perform the following consensus update:

$$ u_{mn}(t) = \frac{1}{|T_m|} \sum_{r_{mn} \in T_m} u_{mn}(t). \quad (2.8) $$

In Figure 2.3 we observe that there are robots that either wait at the communication points for the arrival of other robots or depart from a communication point in order to communicate with other robots. Figure 2.4 shows that eventually all robots reach a consensus on the numbers $u_{mn}(t)$, which means that communication takes place infinitely often, as expected due to Proposition 2.4.4. The communication events over time for teams $T_1$ and $T_3$ are depicted in Figure 2.5. Observe also in Figure 2.5 that the time instances at which a communication event occurs do not depend linearly on time, which means that communication for these teams is aperiodic. Note also that the distributed intermittent communication framework presented in Zavlanos (2010) cannot be applied in this simulation case, since the mobility graph
Figure 2.5: Graphical depiction of communication events at nodes $T_1$ (Figure 2.5(a)) and $T_3$ (Figure 2.5(b)) with respect to time.

shown in Figure 2.3 is not bipartite.
In this chapter, we consider robots that are responsible for accomplishing independent tasks that are specified by local Linear Temporal Logic (LTL) formulas which can capture, e.g., data gathering and surveillance tasks. As the robots navigate the workspace to accomplish their tasks they collect information that needs to reach all other robots and possibly a user. We assume that the robots have limited communication capabilities and can only communicate when they are physically close to each other. As a result, intermittent communication protocols are necessary that allow the robots to temporarily disconnect from the network in order to accomplish their tasks free of communication constraints. Our goal in this chapter is to design controllers that ensure that the assigned local LTL tasks are satisfied, the communication network is connected over time infinitely often, and a user-specified metric, such as total traveled distance, is minimized.
3.1 Preliminaries

In this section we formally describe Linear Temporal Logic (LTL) by presenting its syntax and semantics. Also, we briefly review preliminaries of automata-based LTL model checking, while a detailed overview of this theory can be found in Baier and Katoen (2008).

The basic ingredients of Linear Temporal Logic are a set of atomic propositions $\mathcal{AP}$, the boolean operators, i.e., conjunction $\land$, and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar:

$$\phi ::= \text{true} \mid \pi \mid \phi_1 \land \phi_2 \mid \neg \phi \mid \bigcirc \phi \mid \phi_1 \mathcal{U} \phi_2,$$

where $\pi \in \mathcal{AP}$. For the sake of brevity we abstain from presenting the derivations of other Boolean and temporal operators, e.g., always $\mathcal{A}$, eventually $\mathcal{E}$, implication $\Rightarrow$, which can be found in Baier and Katoen (2008). An infinite word $\sigma$ over the alphabet $2^{\mathcal{AP}}$ is defined as an infinite sequence $\sigma = \pi_0 \pi_1 \pi_2 \cdots \in (2^{\mathcal{AP}})^\omega$, where $\omega$ denotes infinite repetition and $\pi_k \in 2^{\mathcal{AP}}, \forall k \in \mathbb{N}$. The language

$$\text{Words}(\phi) = \{\sigma \in (2^{\mathcal{AP}})^\omega | \sigma \models \phi\}$$

is defined as the set of words that satisfy the LTL formula $\phi$, where $\models \subseteq (2^{\mathcal{AP}}) \times \phi$ is the satisfaction relation.

Any LTL formula $\phi$ can be translated into a Nondeterministic Büchi Automaton (NBA) over $2^{\mathcal{AP}}$ denoted by $B$ Vardi and Wolper (1986), which is defined as follows:

**Definition 3.1.1 (NBA).** A Nondeterministic Büchi Automaton (NBA) $B$ over $2^{\mathcal{AP}}$ is defined as a tuple $B = (Q_B, Q_B^0, \Sigma, \rightarrow_B, F_B)$, where

- $Q_B$ is the set of states;
- $Q_B^0 \subseteq Q_B$ is a set of initial states;
• $\Sigma = 2^\text{AP}$ is an alphabet;

• $\to_B \subseteq Q_B \times \Sigma \times Q_B$ is the transition relation; and

• $F_B \subseteq Q_B$ is a set of accepting/final states.

An infinite run $\rho_B$ of $B$ over an infinite word $\sigma = \pi_0 \pi_1 \pi_2 \ldots$, $\pi_k \in \Sigma = 2^\text{AP}$ $\forall k \in \mathbb{N}$ is a sequence $\rho_B = q^0_B q^1_B q^2_B \ldots$ such that $q^0_B \in Q^0_B$ and $(q^k_B, \pi_k, q^{k+1}_B) \in \to_B$, $\forall k \in \mathbb{N}$. An infinite run $\rho_B$ is called accepting if $\text{Inf}(\rho_B) \cap F_B \neq \emptyset$, where $\text{Inf}(\rho_B)$ represents the set of states that appear in $\rho_B$ infinitely often. The words $\sigma$ that result in an accepting run of $B$ constitute the accepted language of $B$, denoted by $\mathcal{L}_B$. Then it is proven Baier and Katoen (2008) that the accepted language of a NBA $B$, associated with an LTL formula $\phi$, is equivalent to the words of $\phi$, i.e., $\mathcal{L}_B = \text{Words}(\phi)$.

3.2 Problem Formulation

Consider $N \geq 1$ mobile robots operating in a workspace $\mathcal{W} \subset \mathbb{R}^d$, $d = 2, 3$, containing $W > 0$ locations of interest denoted by $\mathbf{v}_j$, $j \in \mathcal{I} := \{1, \ldots, W\}$. Mobility of robot $i \in \mathcal{N} := \{1, \ldots, N\}$ in $\mathcal{W}$ is captured by a weighted Transition System (wTS) that is defined as follows:

**Definition 3.2.1** (weighted Transition System). A weighted Transition System for robot $i$, denoted by $wTS_i$, is a tuple $wTS_i = (\mathcal{Q}_i, q^0_i, \to_i, w_i, \text{AP}, L_i)$ where

• $\mathcal{Q}_i = \{q^j_i, j \in \mathcal{I}\}$ is the set of states, where a state $q^j_i$ indicates that robot $i$ is at location $\mathbf{v}_j \in \mathcal{W}$;

• $q^0_i \in \mathcal{Q}_i$ is the initial state of robot $i$;

• $\to_i \subseteq \mathcal{Q}_i \times \mathcal{Q}_i$ is a given transition relation such that $(q^j_i, q^e_i) \in \to_i$ if there exists a controller that can drive robot $i$ from location $\mathbf{v}_j$ to $\mathbf{v}_e$ in finite time without going through any other location $\mathbf{v}_c$;
\( w_i : Q_i \times Q_i \rightarrow \mathbb{R}_+ \) is a function that assigns a cost/weight to each transition \((q_i^v, q_i^{v_e}) \in \rightarrow_i\) that can capture, e.g., the time, energy, or distance required for robot \( i \) to move from \( v_j \) to \( v_e \);

- \( \mathcal{AP} = \{\{\pi_i^{v_j}\}_{i=1}^N\}_{j \in \mathcal{I}} \) is the set of atomic propositions associated with each state; and

- \( L_i : Q_i \rightarrow \mathcal{AP} \) is defined as \( L_i(q_i^v) = \pi_i^{v_j} \), for all \( i \in \mathcal{N} \) and \( j \in \mathcal{I} \).

Every robot \( i \in \mathcal{N} \) is responsible for accomplishing high-level tasks associated with some of the locations \( v_j, j \in \mathcal{I} \). Hereafter, we assume that the tasks assigned to the robots are independent from each other. Specifically, we assume that the task assigned to robot \( i \) is captured by a local LTL\(_{-\bigcirc}\) formula \( \phi_i \) Clarke et al. (1999) specified over the set of atomic propositions \( \mathcal{AP} = \{\{\pi_i^{v_j}\}_{i=1}^N\}_{j \in \mathcal{I}} \), where \( \pi_i^{v_j} = 1 \) if \( \|x_i - v_j\| \leq \epsilon \), for a sufficiently small \( \epsilon > 0 \), and 0 otherwise, for all \( i \in \mathcal{N} \) and \( j \in \mathcal{I} \). Namely, the atomic proposition \( \pi_i^{v_j} \) is true if robot \( i \) is sufficiently close to location \( v_j \). For example, an LTL\(_{-\bigcirc}\) task for robot \( i \) can be: \( \phi_i = (\Box \pi_i^{v_4}) \land ((-\pi_i^{v_4}) \bigcup \pi_i^{v_8}) \land (\Diamond \pi_i^{v_5}) \land (\Box \pi_i^{v_3}) \land (\Diamond \pi_i^{v_1}) \), which requires robot \( i \) to (i) visit location \( v_4 \) infinitely often, (ii) never visit location \( v_4 \) until location \( v_8 \) is visited, (iii) eventually visit location \( v_5 \), (iv) always avoid an obstacle located at \( v_3 \), and (v) visit location at \( v_1 \) infinitely often. Together with accomplishing local tasks, robots are also responsible for communicating with each other so that any information that is collected as part of these tasks is propagated in the network and, possibly, eventually reaches a user.

To define a communication network among the robots, we partition the robot team into \( M \geq 1 \) robot teams \( T_m, m \in \{1, \ldots, M\} \), so that the resulting team membership graph \( G_T \), defined in Definition 2.1.1, in Chapter 2, is connected. In\(^1\) The syntax of LTL\(_{-\bigcirc}\) is the same as the syntax of LTL excluding the ‘next’ operator. The choice of LTL\(_{-\bigcirc}\) over LTL is motivated by the fact that we are interested in the continuous time execution of the synthesized plans, in which case the next operator is not meaningful. This choice is common in relevant works, see, e.g., Kloetzer and Belta (2008) and the references therein.
this chapter, we assume that the teams are *a priori* known and can be selected arbitrarily as long as the graph of teams $G_T$ is connected. Since the robots have limited communication capabilities, we assume that the robots in every team $T_m$ can only communicate if all of them are simultaneously present at a common location $v_j \in \mathcal{W}$, hereafter called a communication point. We assume that there are $R \geq 1$ available communication points in the workspace at locations $v_j \in \mathcal{W}$, where $j \in C \subset \mathcal{I}$. The indices $j$ of the communication points that are specifically available to the robotic team $T_m$ are collected in a finite set $C_m \subseteq C$, where the sets $C_m$ are not necessarily disjoint. When all robots in a team $T_m$ have arrived at a communication location, we assume they form a connected sub-network. Then, communication within team $T_m$ happens and its robot members leave to accomplish their tasks or communicate with other teams. This way, a dynamic robot communication network $G_c(t)$ is constructed, as defined in Definition 2.1.2, in Chapter 2.

Intermittent connectivity of the communication network $G_c(t)$ can be captured by the global LTL formula

$$
\phi_{\text{com}} = \bigwedge_{m \in \{1,2,\ldots,M\}} \left( \Box \left( \bigvee_{j \in C_m} \left( \bigwedge_{i \in T_m} \pi_i^{v_j} \right) \right) \right), \quad (3.1)
$$

specified over the set of atomic propositions $\{\pi_i^{v_j}\}_{j=1}^N_{i \in C}$. Composing $\phi_{\text{com}}$ with the local LTL formulas $\phi_i$, yields the following global LTL statement

$$
\phi = \left( \bigwedge_{i} \phi_i \right) \land \phi_{\text{com}}, \quad (3.2)
$$

that captures the local tasks assigned to every robot and intermittent connectivity of the communication network $G_c(t)$.

Given the wTS$_i$, for all robots $i \in \mathcal{N}$, and the global LTL formula (3.2), the goal is to synthesize motion plans $\tau_i$, for all $i \in \mathcal{N}$, whose execution satisfies the global LTL formula (3.2). Typically, such motion plans are infinite paths in wTS$_i$, Clarke et al. (1999), i.e., infinite sequences of states in wTS$_i$, such that $\tau_i(1) = q_i^0$. 

42
\( \tau_i(\kappa) \in Q_i \), and \((\tau_i(\kappa), \tau_i(\kappa + 1)) \in \rightarrow_i, \forall \kappa \in \mathbb{N}_+ \), where \( \tau_i(\kappa) \) denotes the \( \kappa \)-th state in \( \tau_i \). In this form, they cannot be manipulated in practice. This issue can be resolved by representing these plans in a prefix-suffix form Vardi and Wolper (1986), i.e.,

\[ \tau_i = \tau_i^{\text{pre}} \left[ \tau_i^{\text{suf}} \right]^{\omega} \]

where the prefix part \( \tau_i^{\text{pre}} \) and suffix part \( \tau_i^{\text{suf}} \) are both finite paths in \( wTS_i \), for all \( i \in \mathcal{N} \). The prefix \( \tau_i^{\text{pre}} \) is executed once and the suffix \( \tau_i^{\text{suf}} \) is repeated indefinitely. The cost associated with a plan \( \tau_i = \tau_i^{\text{pre}} \left[ \tau_i^{\text{suf}} \right]^{\omega} \) is defined as

\[ J_p(\tau_i) = \alpha J(\tau_i^{\text{pre}}) + (1 - \alpha) J(\tau_i^{\text{suf}}), \quad (3.3) \]

where \( J(\tau_i^{\text{pre}}) \) and \( J(\tau_i^{\text{suf}}) \) represent the cost of the prefix and the suffix part, respectively, and \( \alpha \in [0, 1] \) is a user-specified parameter. The cost \( J(\tau_i^{\text{suf}}) \) of the suffix part is defined as

\[ J(\tau_i^{\text{suf}}) = \sum_{\kappa=1}^{|\tau_i^{\text{suf}}|} w_i(\tau_i^{\text{suf}}(\kappa), \tau_i^{\text{suf}}(\kappa + 1)), \quad (3.4) \]

where \( |\tau_i^{\text{suf}}| \) stands for the number of states in the finite path \( \tau_i^{\text{suf}} \), \( \tau_i^{\text{suf}}(\kappa) \) denotes the \( \kappa \)-th state in \( \tau_i^{\text{suf}} \), and \( w_i \) are the weights defined in Definition 3.2.1. The cost \( J(\tau_i^{\text{pre}}) \) of the prefix part is defined accordingly.

The problem that is addressed in this paper can be summarized as follows:

**Problem 3.2.2.** Consider any initial configuration of a network of \( N \) mobile robots in their respective \( wTSs \), and any partition of the network in \( M \) subgroups \( T_m, m \in \mathcal{M} \) so that the associated graph \( G_T \), defined in Definition 2.1.1 in Chapter 2, is connected. Determine discrete motion plans \( \tau_i \), i.e., sequences of states \( q_i^v \in Q_i \), in prefix-suffix structure, for all robots such that the LTL specification \( \phi \) defined in (3.2) is satisfied, i.e., (i) the local LTL tasks \( \phi_i \) are satisfied, for all \( i \in \mathcal{N} \), (ii) intermittent communication among robots captured by \( \phi_{\text{com}} \) is ensured infinitely often, and (iii) a user-specified metric \( \sum_{i \in \mathcal{N}} J_p(\tau_i) \), such as total traveled distance, is minimized.

To solve Problem 3.2.2, we propose a decentralized algorithm that consists of three main parts; see also Figure 3.2. First, we construct motion plans indepen-
Figure 3.1: A graphical illustration of the problem formulation. A network of $N = 3$ robots (black dots) divided into $M = 3$ teams is depicted. The robot teams are selected to be: $T_1 = \{1, 2\}$, $T_2 = \{2, 3\}$, and $T_3 = \{3, 1\}$. The set $I$ consists of locations represented by red and green boxes. Red boxes comprise set $C$ and represent communication points. Black dashed lines stand for paths in the workspace $W$ that connect locations $v_e$ and $v_j$. The sets of communications points for each team are defined as $C_1 = \{v_9, v_{10}\}$, $C_2 = \{v_{10}, v_{11}\}$, and $C_3 = \{v_{12}\}$.

dently for all robots $i$ so that the local LTL tasks $\phi_i$ are satisfied; see Section 3.3. Second, we compute communication schedules, as discussed in Chapter 2, that ensure connectivity of $G_c(t)$ over time infinitely often. Then, in Section 3.4, we integrate task planning and communication control by introducing appropriate communication locations in the discrete motion plans of the robots so that the local tasks controlled in Section 3.3 are satisfied, communication at these locations respects the communication schedules, and a user specified metric is minimized.

3.3 Task Planning

In this section, we summarize an existing optimal control synthesis method that can be used to construct motion plans $\tau_i$ that satisfy the LTL statement $\phi_i$ see e.g. Smith et al. (2011); Guo and Dimarogonas (2013, 2015). First, we provide some
basic definitions related to the wTS that we will use throughout this section. More details about these definitions can be found in Baier and Katoen (2008).

**Definition 3.3.1** (Trace of infinite path). The trace of an infinite path

\[ \tau_i = \tau_i(1)\tau_i(2)\tau_i(3) \ldots, \]

of a weighted transition system wTS\(_i\), denoted by \( \text{trace}(\tau_i) \), is an infinite word that is determined by the sequence of atomic propositions that are true in the states along \( \tau_i \), i.e., \( \text{trace}(\tau_i) = L_i(\tau_i(1))L_i(\tau_i(2)) \ldots \).

**Definition 3.3.2** (Motion Plan). Given \( \phi_i \) and wTS\(_i\) both defined over the set of atomic propositions AP\(_i\), an infinite path \( \tau_i \) of wTS\(_i\) is called motion plan if and only if it satisfies \( \phi_i \) i.e., \( \text{trace}(\tau_i) \in \text{Words}(\phi) \), which is equivalently denoted by \( \tau_i \models \phi_i \).

**Definition 3.3.3** (Trace of wTS). The trace of wTS\(_i\) is defined as \( \text{trace}(wTS_i) = \bigcup_{\tau_i \in \mathcal{P}} \text{trace}(\tau_i) \), where \( \mathcal{P} \) is the set of all infinite paths \( \tau_i \) in wTS\(_i\).

To construct a motion plan \( \tau_i \) that satisfies \( \phi_i \), first, we translate \( \phi_i \) into a Nondeterministic Büchi Automaton \( B_i \) over \( 2^{AP} \), as defined in Definition (3.1.1). Second, given the weighted Transition System wTS\(_i\) and the NBA \( B_i \), a motion plan \( \tau_i \) can be constructed by checking the non-emptiness of the language of the Product Büchi Automaton (PBA) \( P_i = wTS_i \otimes B_i \) Baier and Katoen (2008), which is defined as follows:
Definition 3.3.4 (PBA). Given the wTS $wTS_i = (Q_i, q_i^0, \rightarrow_i, w_i, AP, L_i)$ and the NBA $B_i = (Q_B, Q_B^0, 2^{AP}, \rightarrow_B, F_B)$, the Product Büchi Automaton $P_i = wTS_i \otimes B_i$ is a tuple $(Q_{P_i}, Q_{P_i}^0, \rightarrow_{P_i}, w_{P_i}, F_{P_i})$ where

- $Q_{P_i} = Q_i \times Q_B$ is the set of states;
- $Q_{P_i}^0 = q_i^0 \times Q_B^0$ is a set of initial states;
- $\rightarrow_{P_i} \subseteq Q_{P_i} \times Q_{P_i}$ is the transition relation. Transition $(q_P, q'_P) \in \rightarrow_{P_i}$, where $q_P = (q_i^v, q_B) \in Q_{P_i}$ and $q'_P = (q_i^v, q_B') \in Q_{P_i}$, exists if $(q_i^v, q_i^v) \in \rightarrow_i$ and $(q_B, L_i(q_i^v), q_B') \in \rightarrow_B$;
- $w_{P_i} : Q_{P_i} \times Q_{P_i} \to \mathbb{R}_+$ is the weight function, defined as: $w_{P_i}((q_i^v, q_B), (q_i^v, q_B')) = w_i(q_i^v, q_i^v)$; and
- $F_{P_i} = Q_i \times F_B$ is a set of accepting/final states.

Next, to derive motion plans $\tau_i$ that satisfy $\phi_i$, graph search techniques are applied to $P_i$, see e.g., Smith et al. (2011); Guo and Dimarogonas (2013, 2015). In particular, to check the non-emptiness of the language of $P_i$, it suffices to check if there is a non-empty intersection between the words that can be generated by $wTS_i$ and the words that satisfy the formula $\phi_i$. Since $\text{Words}(\phi_i) = L_B$, this is equivalent to checking that $\text{trace}(wTS_i) \cap L_B \neq \emptyset$, where $\text{trace}(wTS_i) \cap L_B$ is the language of $P_i$. Then to find a motion plan $\tau_i \models \phi_i$, it suffices to find an infinite path $\tau_i$ of $wTS_i$ such that $\text{trace}(\tau_i) \in \text{trace}(wTS_i) \cap L_B$. To achieve that, an accepting run $\rho_{P_i}$ of $P_i$, i.e., a run that satisfies $\text{Inf}(\rho_{P_i}) \cap F_{P_i} \neq \emptyset$ needs to be computed. Typically, to derive such motion plans, the product automaton is viewed as a weighted graph $G_{P_i} = \{V_{P_i}, E_{P_i}, w_{P_i}\}$, where $V_{P_i} = Q_{P_i}$ and the set of edges $E_{P_i}$ is determined by the transition relation $\rightarrow_{P_i}$ with weights on the edges determined by $w_{P_i}$. Then finding a path from an initial state to an accepting state in $G_{P_i}$ (prefix) followed by
a cycle around this accepting state (suffix), which is repeated infinitely, results in an accepting run in a prefix-suffix structure as follows:

\[
\rho_{P_i} = \rho_{P_i}^{\text{pre}} \left[ \rho_{P_i}^{\text{suf}} \right]^\omega = \left[ (q_{wTS_i}^0, q_{B_i}^0), (q_{wTS_i}^1, q_{B_i}^1), \ldots, (q_{wTS_i}^f, q_{B_i}^f) \right]_{e \in \mathcal{E}_{P_i}} \left[ (q_{wTS_i}^f, q_{B_i}^f), \ldots, (q_{wTS_i}^K, q_{B_i}^K) \right]_{e \in \mathcal{E}_{P_i}}^\omega.
\]

(3.5)

Then, the projection of \( \rho_{P_i} \) onto \( w_{TS_i} \), denoted by \( \Pi_{w_{TS_i}} \rho_{P_i} \), results in a motion plan that has the following prefix-suffix structure:

\[
\tau_i = \Pi_{w_{TS_i}} \rho_{P_i} = \tau_i^{\text{pre}} \left[ \tau_i^{\text{suf}} \right]^\omega = \left[ q_{wTS_i}^0, \ldots, q_{wTS_i}^f \right] \left[ q_{wTS_i}^f, \ldots, q_{wTS_i}^K \right]^\omega,
\]

(3.6)

that satisfies \( \phi_i \) Vardi and Wolper (1986) and the prefix part \( \tau_i^{\text{pre}} \) is executed once and the suffix part \( \tau_i^{\text{suf}} \) is repeated indefinitely. Among all motion plans \( \tau_i = \tau_i^{\text{pre}} \left[ \tau_i^{\text{suf}} \right]^\omega \) that satisfy \( \phi_i \), we select the one that minimizes the cost function (3.3).

### 3.4 Integrated Task Planning and Intermittent Communication Control

In this section, we synthesize motion plans \( \tilde{\tau}_i \) in prefix-suffix form that satisfy both the local tasks \( \phi_i \), \( \forall i \in \mathcal{N} \) and the intermittent connectivity requirement \( \phi_{\text{com}} \) and minimize the total cost \( \sum_{i \in \mathcal{N}} J_p(\tilde{\tau}_i) \). To achieve this, first we compute communication schedules \( \text{sched}_i \) as discussed in Chapter 2. Then, in Section 3.4.1, we modify the motion plans \( \tau_i \models \phi_i \) by introducing communication locations so that the local tasks \( \phi_i \) are satisfied, intermittent communication is ensured infinitely often as per the designed schedules \( \text{sched}_i \), and the total cost \( \sum_{i \in \mathcal{N}} J_p(\tilde{\tau}_i) \) is minimized. Construction of the revised plans occurs in a decentralized way and offline after the construction of the communication schedules; see also Figure 3.2. Then, in Section 3.4.2, we show that the resulting motion plans can be executed asynchronously among the robots.
Algorithm 3: Decentralized construction of motion plan $\tilde{\tau}_i$

**Input:** Sequence $S$, set of teams $\mathcal{M}_i$, set of communication points $C_m$, $\forall m \in \mathcal{M}_i$ and motion plan $\tau_i \models \phi_i$

**Output:** Revised plan: $\tilde{\tau}_i = \tau_{i}^{\text{pre}} \tau_{i}^{\text{suf,1}} \tau_{i}^{\text{suf,2}} \ldots \tau_{i}^{\text{suf,n}} \ldots$

1. $n_i := 1$, $\tau_{i}^{\text{suf,n}_i} := \tau_{i}^{\text{suf}}$, $k = 1$, $D_i := \mathcal{M}_i$;

2. while no repetitive pattern is detected do

3. $\tau_{i}^{\text{suf,n}_i+1} = \tau_{i}^{\text{suf,n}_i}$;

4. while $D_i \neq \emptyset$ do

5. Robot $i$ waits until a robot $e \in \hat{S}(k-1)$ notifies it that it is the turn of team $T_m = \hat{S}(k)$, $m \in \mathcal{M}_i$ to update $v_j, j \in C_m$;

6. Delete from $\tau_{i}^{\text{suf,n}_i+1}$ the previously selected communication point for team $T_m$;

7. The robot $h \in T_m$ with the highest index solves (3.7) and transmits the solution to all other robots in team $T_m$;

8. Update $\tau_{i}^{\text{suf,n}_i+1}$ as per the solution of (3.7);

9. $D_i := D_i \setminus \{m\}$;

10. A robot $e \in \hat{S}(k) \cap \hat{S}(k+1)$ notifies robots of team $\hat{S}(k+1)$;

11. $k := k + 1$;

12. $D_i := \mathcal{M}_i$;

13. $n_i := n_i + 1$;


3.4.1 Modification of motion plans $\tau_i$

In this section, our goal is to revise the motion plans $\tau_i = \tau_{i}^{\text{pre}} [\tau_{i}^{\text{suf}]\omega} \models \phi_i$, derived in Section 3.3, so that both the local task $\phi_i$ and the intermittent communication constraints are satisfied. To achieve this, it suffices to modify the suffix parts $\tau_{i}^{\text{suf}}$, for all $i \in \mathcal{N}$, since the prefix parts $\tau_{i}^{\text{pre}}$ are executed only once and the suffix parts are executed infinitely often. Specifically, in the suffix parts $\tau_{i}^{\text{suf}}$ we introduce states of wTS$_i$ associated with communication points for each team $T_m$, $m \in \mathcal{M}_i$, without violating the local LTL$_{\omega}$ formulas $\phi_i$. In what follows, we describe how the robots in team $T_m$, $m \in \mathcal{M}_i$ select and incorporate a common communication point $v_j$, $j \in C_m$ into their respective suffix parts. This procedure is described in Algorithm 3.

Selection of communication points for each team $T_m$, $m \in \mathcal{M}_i$, occurs sequentially across the teams, in an order that is determined by the sequence of teams $\hat{S}^\omega$, where
\( \omega \) stands for the infinite repetition of the sequence \( \hat{S} \), i.e., \( \hat{S}^\omega = \hat{S}, \hat{S}, \hat{S}, \ldots \). The sequence \( \hat{S} \) is defined as \( \hat{S} = S, \bar{S} \), where \( S \) is the sequence defined in Definition 2.2.1 and \( \bar{S} \) is a sequence created by reversing the order in which teams appear in \( S \); see, e.g., Figure 3.3. This way we ensure that consecutive teams in \( \hat{S} \) are neighboring teams in \( G_T \). As discussed above, the communication points are updated sequentially across the teams in an order determined by \( \hat{S} \). Specifically, the team \( \hat{S}^\omega(k) \) will update its communication point as soon as team \( \hat{S}^\omega(k - 1) \) updates its respective communication point. Once the robots in team \( \hat{S}^\omega(k - 1) \) update their communication point a robot \( e \in \hat{S}(k) \cap \hat{S}(k - 1) \) notifies robots in team \( \hat{S}(k) \) that it is their turn to update their communication points. Consequently, only one team at a time updates its communication point.

In Algorithm 3, the nested while-loop in lines 4-11 modifies the suffix part of motion plan \( \tau_i = \tau_i^{\text{pre}} \left[ \tau_i^{\text{suf}} \right]^\omega \) for robot \( i \) by introducing states of \( Q_i \) associated with communication points. The while-loop in lines 4-11 terminates when communication points for all teams \( T_m, \ m \in \mathcal{M}_i \) have been selected or updated once; this corresponds to a single execution of the algorithm over either the sequence \( S \) or the sequence \( \bar{S} \). Once this happens, the \( n_i \)-th revised suffix part of robot \( i \), denoted by \( \tau_i^{\text{suf},n_i} \), is generated, where \( \tau_i^{\text{suf},1} = \tau_i^{\text{suf}} \), for all \( i \in \mathcal{N} \) [line 1, Algorithm 3]; see Figure 3.3, as well. This process is repeated indefinitely over the sequence \( \hat{S} \), updating in this way the communication points selected at the previous iteration \( n_i \), until the outer while-loop in lines 2-13 terminates. The latter will happen once a repetitive pattern in the revised suffix parts is detected. Specifically, in Proposition 3.5.1, we show that there exists a positive integer \( C \) common for all robots, such that the suffix parts \( \tau_i^{\text{suf},n_i} \) for all \( n_i > C \), for all robots \( i \in \mathcal{N} \), have already been constructed at a previous iteration \( \eta_i < C \) of Algorithm 3. Once this repetitive pattern is detected Algorithm 3 terminates, and motion plans in a prefix-suffix structure are constructed.

In what follows, we discuss how the suffix part \( \tau_i^{\text{suf},n_i+1} \) is constructed given the
suffix part $\tau_{i,suf,n_i}^{+,1}$ from the previous iteration [lines 2-13, Alg. 3]. The suffix part $\tau_{i,suf,n_i+1}^{+,1}$ is initialized as $\tau_{i,suf,n_i+1}^{+,1} = \tau_{i,suf,n_i}^+$ for all $n_i \geq 2$, $n_i \in \mathbb{N}$ [line 3, Alg. 3] and $\tau_{i,suf,1}^+ = \tau_{i,suf}^+$ [line 1, Algorithm 3]. Next, assume that all robots $i \in \mathcal{T}_m = \mathcal{S}(k)$ get notified by a robot $e \in \mathcal{S}(k) \cap \mathcal{S}(k-1)$ that it is the turn of team $\mathcal{T}_m$ to update its communication point; see line 5, Algorithm 3. Then, the robots in team $\mathcal{T}_m = \mathcal{S}(k)$ will collectively select a communication point $v_j, j \in \mathcal{C}_m$, for team $\mathcal{T}_m$, and incorporate the respective state $q_{i,v}^j \in \mathcal{Q}_i$ into their suffix parts $\tau_{i,suf,n_i+1}^{+,1}$ [lines 5-7, Alg. 3]. The state $q_{i,v}^j$ associated with the communication point $v_j, j \in \mathcal{C}_m$ is incorporated into $\tau_{i,suf,n_i+1}^{+,1}$, for all robots $i \in \mathcal{T}_m$, by solving the following optimization problem [lines 7-8, Alg. 3]:

$$\min_{\{q_{i,v}^j \in \mathcal{C}_m\}_{v \in \mathcal{T}_m},\{\text{index}_i^m\}_{v \in \mathcal{T}_m}} \sum_{i \in \mathcal{T}_m} J(\tau_{i,suf,n_i+1}^{+,1}) \tag{3.7}$$

subject to

$$\tau_{i,suf,n_i+1}^{+,1}(\text{index}_i^m) = q_{i,v}^j, \forall i \in \mathcal{T}_m$$

$$\tau_{i,pre}^{+,1}[\tau_{i,suf,n_i+1}^{+,1}]| = \phi_i, \forall i \in \mathcal{T}_m$$

$$\text{index}_i^m \geq \text{index}_i^o, \forall p \in \mathcal{I}^+_{s_k,m}, \forall i \in \mathcal{T}_m$$

$$\text{index}_i^m \leq \text{index}_i^o, \forall w \in \mathcal{I}^-_{s_k,m}, \forall i \in \mathcal{T}_m$$

$$q_{i,v}^j = q_{e,v}^j, \forall i, e \in \mathcal{T}_m, j \in \mathcal{C}_m,$$

where $J(\tau_{i,suf,n_i+1}^{+,1})$ stands for the cost of the suffix part $\tau_{i,suf,n_i+1}^{+,1}$, defined in (3.4) and $\text{index}_i^m$ represents the index of the entry in $\tau_{i,suf,n_i+1}^{+,1}$ where the selected communication point $j \in \mathcal{C}_m$ will be placed, i.e., $\tau_{i,suf,n_i+1}^{+,1}(\text{index}_i^m) = q_{i,v}^j, j \in \mathcal{C}_m$. According to the objective function in (3.7), the communication point $v_j$ is selected so that the sum of the costs of $\tau_{i,suf,n_i+1}^{+,1}$ for all robots $i \in \mathcal{T}_m$ is minimized. The first constraint requires that all robots $i \in \mathcal{T}_m$ place the state $q_{i,v}^j$ at the entry of $\tau_{i,suf,n_i+1}^{+,1}$ with index $\text{index}_i^m$. The second constraint in (3.7) requires that the prefix part $\tau_{i,pre}^{+,1}$ and
the infinite repetition of the resulting suffix part $\tau_{n_i+1}^{\text{suf}}$ satisfies the local task $\phi_i$. The third and the fourth constraints ensure that all states $q^{\psi_j}_i$, $j \in C_m$, for every team $T_m$, $m \in M_i$ are introduced in $\tau_{n_i+1}^{\text{suf}}$ in the order that the indices of teams $m \in M_i$ appear in $s_i$. In other words, the second and the third constraints ensure that robot $i$ will participate in communication events in the order that is determined by $\text{sched}_i = s^{\omega}_i$. As discussed in Remark 2.2.6, and as it will be shown in Proposition 3.5.4 and in Theorem 3.5.8, this ensures that the network never reaches a deadlock configuration and guarantees intermittent communication infinitely often.

Notice that in the optimization problem (3.7), the symbols $X$ that appear in the schedules $\text{sched}_i = s^{\omega}_i$ are ignored and are not introduced in the constructed suffix parts. In the second constraint, the set of teams $I^+_s, m$ collects the indices of teams $T_p$, $p \in M_i$ within which communication has to happen after communication within team $T_m$, according to the sequence $s_i$. Mathematically, the set $I^+_s, m$ is defined as

$$I^+_s, m = \{p \in M_i|n^T_p > n^T_m\},$$

where $n^T_p$ is the index of the entry in the sequence $s_i$ where the index $p \in M_i$ of team $T_p$ is located, i.e. $s_i(n^T_p) = p$. Similarly, in the third constraint the set of teams $I^-_s, m$ is defined as $I^-_s, m = \{p \in M_i|n^T_p < n^T_m\}$. The last constraint requires that all robots in team $T_m$ select the same communication point.

Finally, notice that the indices $\text{index}_i^m$ are not required to be the same for all robots in team $T_m$.

The optimization problem (3.7) is solved through a two-step procedure. First we solve the optimization problem (3.7) ignoring the first constraint $\tau_{n_i+1}^{\text{suf}} = \phi_i$. The resulting optimization problem is an integer program and can be solved using existing optimization methods. Second, we check if the infinite repetition of the suffix part $\tau_{n_i+1}^{\text{suf}}$ that we get from the first step satisfies the first constraint $\tau_{n_i+1}^{\text{suf}} = \phi_i$. If this is the case, then the solution of the optimization problem constructed in the first step is the optimal solution of the constrained optimization problem.
problem (3.7). Otherwise, the solution that we got in the first step is discarded from the feasible set of the optimization problem (3.7). Then the above procedure is repeated until a solution is found. At the worst case, the integer program constructed in the first step of this procedure is solved \(|C_m|\prod_{i\in T_m}(|\tau_i^{\text{suf},n_i}| + 1)\) times, where \(|\tau_i^{\text{suf},n_i}| + 1\) stands for the maximum number of possible values of the variable \(\text{index}_i^m\).

Notice that since the suffix part \(\tau_i^{\text{suf},n_i+1}\) is initialized as \(\tau_i^{\text{suf},n_i+1} = \tau_i^{\text{suf},n_i}\), a state associated with a communication point \(v_j, j \in C_m\) may already exist in \(\tau_i^{\text{suf},n_i+1}\) before solving the optimization problem (3.7).\(^2\) If this is the case, then this state is deleted from \(\tau_i^{\text{suf},n_i+1}\) (see line 6, Algorithm 3). Then a new communication point is selected for team \(T_m\) by solving (3.7); see line 7, Algorithm 3. The reason we select a new communication point for team \(T_m\) is that there may be a different communication point, as a solution to (3.7), that can decrease even more the cost \(\sum_{v_i \in T_m} J(\tau_i^{\text{suf},n_i+1})\); this will be discussed in Proposition 3.5.2.

Once robots \(i \in T_m\) select and incorporate a communication point for team \(T_m = \hat{S}(k), m \in \mathcal{M}_i\) into \(\tau_i^{\text{suf},n_i+1}\), then (i) the sets \(D_i\) that were initialized as \(D_i = \mathcal{M}_i\) [line 12, Alg. 3] are updated by removing team with index \(m\) from them [line 9, Alg. 3], (ii) a robot \(e \in \hat{S}(k) \cap \hat{S}(k + 1)\) notifies robots of the next team \(S(k + 1)\) that it is their turn to update their communication point [line 10, Alg. 3], and (iii) the iteration index \(k\) is updated [line 11, Alg. 3]. Once \(D_i = \emptyset\), then Algorithm 3 has been executed once over either the whole sequence \(S\) or the whole sequence \(\bar{S}\). Then, the construction of the suffix part \(\tau_i^{\text{suf},n_i+1}\) is finished, i.e., \(n_i = n_i + 1\), and the set \(D_i\) is updated as \(D_i = \mathcal{M}_i\) [lines 12-13, Alg. 3]; see also Figure 3.3. The motion plan \(\tilde{\tau}_i\) generated by Algorithm 3 has the following form \(\tilde{\tau}_i = \tau_i^{\text{pre}} \tau_i^{\text{suf},1} \tau_i^{\text{suf},2} \ldots \tau_i^{\text{suf},n_i} \ldots\). In Proposition 3.5.1, we will show that the resulting motion plan \(\tilde{\tau}_i\) has a finite

\(^2\) Note that when \(n_i > 2\) there is always a communication point in \(\tau_i^{\text{suf},n_i+1}\) from the previous iteration \(n_i\), by construction of Algorithm 3. Recall that the initial suffix parts \(\tau_i^{\text{suf},1} = \tau_i^{\text{suf}}\) do not contain any communication points.
representation and it can be written in a prefix-suffix structure.

Remark 3.4.1 (Construction of sched_i and τ_{i,suf,n_i+1}). Construction of the schedules sched_i and the suffix parts τ_{i,suf,n_i+1} are two independent processes that can occur at the same time. Specifically, once robots i ∈ T_m have constructed their respective schedules, according to Algorithm 1, construction of the suffix parts τ_{i,suf,n_i} can follow. Moreover, recall that (i) the construction of the suffix parts τ_{i,suf,n_i} occurs sequentially according to the sequence Š^ω, i.e., only robots in one team update their respective communication points in their suffix parts τ_{i,suf,n_i} at a time and (ii) the set M_i that collects the teams that the robots belong to differs across the robots. Due to these two
Figure 3.4: Graphical illustration of Algorithm 3 for a network of \( N = 3 \) robots divided into \( M = 2 \) subgroups constructed as follows: \( \mathcal{T}_1 = \{1, 2\}, \mathcal{T}_2 = \{2, 3\} \). The sets \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \) consist of communication points that are depicted by red polygons and yellow circles, respectively. Gray arrows show a path (part of prefix) that connects the initial locations of robots to their suffix parts \( \tau_{i \text{pre}} \), while the colored arrows depict the suffix parts \( \tau_{i \text{suf},n_i} \). Dashed lines stand for the transitions introduced in \( \tau_{i \text{suf},n_i} \) that do not exist in the initial suffix parts \( \tau_{i \text{suf},1} \). In Figure 3.4(a), the initial motion plans \( \tau_i \) are depicted. Figures 3.4(b) and 3.4(c) illustrate the revised suffix parts \( \tau_{i \text{suf},2} \), \( \tau_{i \text{suf},3} \), respectively, after selecting the communication points for each team.

Observations we conclude that some robots may finish the construction of their suffix parts \( \tau_{i \text{suf},n_i} \) sooner than some others. Therefore, the robots update the iteration indices \( n_i = n_i + 1 \) asynchronously during a single execution of Algorithm 3 over either the sequence \( S \) or \( \bar{S} \). Nevertheless, note that robots that finish the construction of their suffix parts \( \tau_{i \text{suf},n_i} \) sooner cannot start constructing the next suffix part immediately due to (i). This imposes an implicit synchronization across the teams. Also, observe that every time Algorithm 3 is executed once over either the whole sequence \( S \) or \( \bar{S} \), all robots \( i \in \mathcal{N} \) have updated their iteration indices \( n_i \), as \( n_i = n_i + 1 \). Therefore, it holds that \( n_i = n \), for all \( i \in \mathcal{N} \), for some \( n \in \mathbb{N}_+ \), for all \( n_i \in \mathbb{N}_+ \); see also Figure 3.3. The update of the iteration indices \( n_i \) along with the construction of the suffix parts \( \tau_{i \text{suf},n_i} \) is also described in Example 3.4.2.

Example 3.4.2 (Algorithm 3). To illustrate Algorithm 3 consider a network of
$N = 3$ robots divided into $M = 2$ teams defined as $\mathcal{T}_1 = \{1,2\}$ and $\mathcal{T}_2 = \{2,3\}$. Each robot first constructs the local motion plans $\tau_i = \tau_i^{\text{pre}}[\tau_i^{\text{suf}}]^\omega$ according to the algorithm described in Section 3.3. Next, given a sequence $S$, defined as $S = \mathcal{T}_1, \mathcal{T}_2$, they construct the following schedules of communication events as per Algorithm 1: $\text{sched}_1 = s_1^\omega = [X,1]^\omega$, $\text{sched}_2 = s_2^\omega = [2,1]^\omega$, and $\text{sched}_3 = s_3^\omega = [2,X]^\omega$. Then, given the sequence $S^\omega$ and the resulting sequences $s_i$, robots construct the suffix parts $\tau_i^{\text{suf},n_i}$. The suffix parts $\tau_i^{\text{suf},1}$ are selected as $\tau_i^{\text{suf},1} = \tau_i^{\text{suf}}$, for all robots $i \in \{1,2,3\}$, and are depicted in Figure 3.4(a). Next, the suffix parts $\tau_i^{\text{suf},2}$ are constructed, which are initialized as $\tau_i^{\text{suf},2} = \tau_i^{\text{suf},1}$, for all robots $i \in \{1,2,3\}$. Also, the indices $n_i$ are initialized as $n_i = 1$, for all $i \in \{1,2,3\}$ and the sets $\mathcal{D}_i$ are initialized as $\mathcal{D}_i = \mathcal{M}_i$, for all $i \in \{1,2,3\}$, i.e., $\mathcal{D}_1 = \{1\}$, $\mathcal{D}_2 = \{1,2\}$, and $\mathcal{D}_3 = \{2\}$. According to sequence $S^\omega$, first, robots 1 and 2 in team $\mathcal{T}_1$ will collectively select and incorporate a common communication point for team $\mathcal{T}_1$ into their suffix parts $\tau_1^{\text{suf},2}$ and $\tau_2^{\text{suf},2}$. Once this happens, it holds that $\mathcal{D}_1 = \emptyset$, $n_1 = 2$ and, therefore, construction of the suffix part $\tau_1^{\text{suf},2}$ is finished. Then, robot 1 waits for the next robots of the next team that appears in $S^\omega$ which is associated with a team that belongs to $\mathcal{M}_1$, to update the respective communication point. This team is again $\mathcal{v}_1 \in \mathcal{C}_1$, since $\mathcal{M}_1 = \{1\}$. On the other hand, robot 2 has not finished the construction of its suffix part $\tau_2^{\text{suf},2}$, since now it holds that $\mathcal{D}_2 = \{2\}$, while robot 3 $\in \mathcal{T}_3$ has not started the construction of $\tau_3^{\text{suf},2}$ yet. Next, according to the sequence $S^\omega$, robots 2 and 3 in team $\mathcal{T}_2$ select and incorporate a common communication point $\mathcal{v}_j$, $j \in \mathcal{C}_2$, in the suffix parts $\tau_2^{\text{suf},2}$ and $\tau_3^{\text{suf},2}$. Once this happens, we have that $\mathcal{D}_2 = \mathcal{D}_3 = \emptyset$, $n_2 = n_3 = 2$ and, consequently, both robot 2 and 3 have finished the construction of the suffix parts $\tau_2^{\text{suf},2}$ and $\tau_3^{\text{suf},2}$. The resulting suffix parts $\tau_i^{\text{suf},2}$, for all $i \in \{1,2,3\}$ are shown in Figure 3.4(b). Notice that now all robots $i$ have finished the construction of the suffix parts $\tau_i^{\text{suf},2}$ and $n_i = 2$ for all $i \in \{1,2,3\}$, i.e., Algorithm 3 has been executed once over the sequence $S$. Observe in Figure 3.4(b), that robot $2 \in \mathcal{T}_1 \cap \mathcal{T}_2$ has incorporated the communication points for
teams $T_1$ and $T_2$, so that during the execution of $\tau_2^{\text{sup}, 2}$, robot 2 first meets with robot 3 $\in T_2$ and then with robot 1 $\in T_1$, as required by the schedule $\text{sched}_2 = s_2^2 = [2, 1]^\omega$. Next, the sets $D_i$ are initialized again as $D_i = M_i$, for all $i \in \{1, 2, 3\}$ and the suffix parts $\tau_i^{\text{sup}, 3}$, for all $i \in \{1, 2, 3\}$, are updated in a similar way. The only difference now is that every time the robots of a team coordinate to construct their suffix parts, they first discard the previously selected communication point for their team and they choose a new one as per the solution of (3.7). The suffix parts $\tau_i^{\text{sup}, 3}$, for all $i \in \{1, 2, 3\}$, are shown in Figure 3.4(c).

3.4.2 Online Asynchronous Execution of Motion Plans

In this section, we describe how the plans $\tilde{\tau}_i$ generated by Algorithm 3 can be executed online in continuous time. Then, in the next section, we show that under the proposed continuous-time execution the global LTL specification (3.2) is satisfied.

In the majority of global LTL-based motion planning, robots are assumed to execute their assigned motion plans synchronously, i.e., all the robots pick synchronously their next states, see e.g., Kloetzer and Belta (2010); Kantaros and Zavlanos (2015). However, assuming that robot motion is performed in a synchronous way is rather conservative due to, e.g., uncertainty and exogenous disturbances in the arrival times of the robots at their next locations as per the plans $\tilde{\tau}_i$. To the contrary, here the discrete plans $\tilde{\tau}_i$ are executed asynchronously across the robots, as per Algorithm 4.

In Algorithm 4, $\tilde{\tau}_i(p_i)$ stands for the $p_i$-th state of the plan $\tilde{\tau}_i$. The different indices $p_i$ for the robots’s states in the plans $\tilde{\tau}_i$ allow us to model the situation where the robots pick asynchronously their next states in $\text{wTS}_i$. Also, in Algorithm 4, the set $P_i$ collects all the indices $p_i$ for robot $i$, for which it holds $\tilde{\tau}_i(p_i) = q^v_{i, j}$, where $q^v_{i, j}$ is associated with a communication point $v_{j, i} \in C_m$, $m \in \mathcal{M}_i$, introduced in $\tilde{\tau}_i$ by Algorithm 3. According to Algorithm 4, when the state of robot $i$ is $\tilde{\tau}_i(p_i) = q^v_{i, j}$, i.e., when robot $i$ arrives at a location $v_j$ in the workspace, it checks if $p_i \in P_i$, see lines...
Algorithm 4: Asynchronous execution of $\tilde{\tau}_i$

**Input:** Motion plan $\tilde{\tau}_i$ and sets of communication points $C_m$, for all $m \in \mathcal{M}_i$

1. $p_i = 1$
2. **while** True **do**
3.  Move towards the state $\tilde{\tau}_i(p_i)$;
4.  **if** $p_i \in \mathcal{P}$, **then**
5.    **Wait at communication point** $v_j$, $j \in C_m$ [Definition 3.4.3];
6.  **if** all robots in $T_m$ are present at node $v_j$ **then**
7.    Communication occurs within team $m$;

3-4, Alg. 4. If so, then robot $i$ performs the following control policy [line 5, Alg. 4]:

**Definition 3.4.3** (Control policy at communication locations). Every robot $i$ that arrives at a communication location $v_j$, $j \in C_m$, $m \in \mathcal{M}_i$, selected by Algorithm 3 waits there indefinitely, or until all other robots in the team arrive.

When all the other robots of team $T_m$ arrive at the communication location $v_j$, $j \in C_m$, communication for team $T_m$ occurs; see lines 6-7, Alg. 4. After that, robot $i$ moves towards the next state $\tilde{\tau}_i(p_i + 1)$. This procedure repeats indefinitely. In Proposition 3.5.4, we show that the waiting time due to the control policy defined in Definition 3.4.3 is upper bounded.

**Remark 3.4.4** (Communication Control Policy). Note that the communication control policy defined in Definition 3.4.3 is essentially the control policy defined in 2.3.1 embedded in the workspace $\mathcal{W}$. This result will be used in Proposition 3.5.4, to show that the network is deadlock-free when the motion plans $\tilde{\tau}_i$ are executed as per Algorithm 4.

3.5 Correctness and Optimality

In this section, we present results pertaining to correctness and optimality of the proposed control scheme. First, in Proposition 3.5.1, we show that Algorithm 3 terminates after a finite number of iterations $n_i$ and, consequently, the infinite paths
\( \tilde{\tau}_i \) generated by Algorithm 3 have a finite representation. Then in Proposition 3.5.2 we examine the optimality of Algorithm 3. Next, in Theorems 3.5.6 and 3.5.8, we show that the continuous-time execution of the motion plans \( \tilde{\tau}_i \) as per Algorithm 4 satisfies the local tasks \( \phi_i \) and the intermittent communication requirement, respectively.

Motion plans \( \tilde{\tau}_i \) generated by Algorithm 3 are infinite paths and, therefore, in practice they are hard to implement and manipulate. In Proposition 3.5.1, we show that the motion plans \( \tilde{\tau}_i \) have a finite representation and they can be expressed in a prefix-suffix structure, where the prefix part \( \tilde{\tau}_i^{\text{pre}} \) is executed only once and the suffix part \( \tilde{\tau}_i^{\text{suf}} \) is repeated indefinitely. Proposition 3.5.1 also shows that Algorithm 3 terminates after a finite number of iterations.

**Proposition 3.5.1 (Prefix-Suffix Form).** Algorithm 3 generates motion plans \( \tilde{\tau}_i \) that can be written in the following prefix-suffix structure:

\[
\tilde{\tau}_i = \tau_i^{\text{pre}} \tilde{\tau}_i^{\text{suf},1} \tau_i^{\text{suf},2} \ldots \tau_i^{\text{suf},P-1} \left[ \tau_i^{\text{suf},P} \ldots \tau_i^{\text{suf},C} \right] \omega
\]

where \( C, P \in \mathbb{N}_+ \) are constants, \( \tilde{\tau}_i^{\text{pre}} = \tau_i^{\text{pre}} \tilde{\tau}_i^{\text{suf},1} \tau_i^{\text{suf},2} \ldots \tau_i^{\text{suf},P-1} \) is the prefix part of \( \tilde{\tau}_i \) and it is executed only once, and \( \tilde{\tau}_i^{\text{suf}} = \tau_i^{\text{suf},P} \ldots \tau_i^{\text{suf},C} \) is the suffix part of \( \tilde{\tau}_i \) and it is repeated infinitely.

**Proof.** The general structure of motion plans \( \tilde{\tau}_i \) generated by Algorithm 3 is

\[
\tilde{\tau}_i = \tau_i^{\text{pre}} \tilde{\tau}_i^{\text{suf},1} \tau_i^{\text{suf},2} \ldots \tau_i^{\text{suf},n_i} \ldots
\]

Notice that there is an iteration index \( k \) such that all robots have the same iteration index \( n_i \), i.e., \( n_i = n \), for some \( n \in \mathbb{N}_+ \) for all robots \( i \in \mathcal{N} \) and for all \( n_i \in \mathbb{N}_+ \); see end of Remark 3.4.1. For each iteration index \( n \in \mathbb{N}_+ \) define the global suffix part \( \tau^{\text{suf},n} \) that stacks all the suffix parts \( \tau_i^{\text{suf},n_i} \) constructed by Algorithm 3 at the end of iteration \( n \), i.e., \( \tau^{\text{suf},n} = [\tau_1^{\text{suf},n_1}, \tau_2^{\text{suf},n_2}, \ldots, \tau_N^{\text{suf},n}]^T \). Also, define the global
prefix part $\tau^{\text{pre}}$ that stacks all the prefix parts $\tau^{\text{pre}}_i$ constructed in Section 3.3, i.e., $\tau^{\text{pre}} = [\tau^{\text{pre}}_1, \tau^{\text{pre}}_2, \ldots, \tau^{\text{pre}}_N]^T$. Then, we can rewrite the motion plans in (3.9) as follows:

$$\tilde{\tau} = [\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_N]^T = \tau^{\text{pre}}_r \tau^{\text{suf},1}_r \tau^{\text{suf},2} \ldots \tau^{\text{suf},n} \ldots (3.10)$$

To show that (3.8) holds for all robots $i \in \mathcal{N}$, it suffices to show that there is a repetitive pattern in the global plan $\tilde{\tau}$ given in (3.10), since we can get a repetitive pattern for $\tilde{\tau}_i$ by projection of $\tilde{\tau}$ on robot $i$, for all $i \in \mathcal{N}$.

To show this result, notice first that all suffix parts $\tau^{\text{suf},n}_i$ are finite paths by construction and, therefore, the number of possible values for $\text{index}^m_i$, returned by (3.7) is finite for all $i \in \mathcal{N}$. Also, the sets of communication points $\mathcal{C}_m$ are finite, for all $m \in \mathcal{M}$ and, therefore, the number of possible communication points the optimization problem (3.7) can select from is finite, for all $i \in \mathcal{N}$. Based on these two observations, there is a finite number of possible suffix parts $\tau^{\text{suf},n}_i$, for all robots $i \in \mathcal{N}$. As a result, there is a finite number of possible global suffix parts $\tau^{\text{suf},n}$ and, therefore, after a finite number of iterations $n$ of Algorithm 3, the global suffix parts $\tau^{\text{suf},n}$ will have appeared during a previous iteration as well.

Let $C$ and $P$ be the first iterations at which it holds that $\tau^{\text{suf},C} = \tau^{\text{suf},P-1}$ with $C > P - 1$, i.e., $\tau^{\text{suf},C}_i = \tau^{\text{suf},P-1}_i$, for all robots $i \in \mathcal{N}$. Consequently, the suffix part $\tau^{\text{suf},C+1}_i$ computed by the while-loop, in lines 4-9 of Algorithm 3, will be the suffix constructed at iteration $P$, i.e., $\tau^{\text{suf},C+1}_i = \tau^{\text{suf},P}_i$ for all robots $i \in \mathcal{N}$, since the optimization problems (3.7) that are solved to construct the suffix parts $\tau^{\text{suf},P}_i$ and $\tau^{\text{suf},C+1}_i$ are the same, for all robots $i \in \mathcal{N}$. Similarly, we have that $\tau^{\text{suf},C+2} = \tau^{\text{suf},P+1}$.

By inspection of the repetitive pattern, we conclude that for any $k \in \mathbb{N}$ it holds that $\tau^{\text{suf},C+k}_i = \tau^{\text{suf},C+k-((C+k)/(C-P+1))-1}(C-P+1)$, where $[\cdot]$ stands for the floor function.
We conclude that the global plan $\tilde{\tau}_i$ in (3.10) can be re-written as follows:

$$\tilde{\tau}_i = \tau_{\text{pre}}^1 \tau_{\text{suf},1} \tau_{\text{suf},2} \ldots \tau_{\text{suf},n} \ldots$$

$$= \tau_{\text{pre}}^{\tau_{\text{suf}}^1} \tau_{\text{suf},2} \ldots \tau_{\text{suf},P-1} \tau_{\text{suf},P} \ldots \tau_{\text{suf},C}$$

$$= \tau_{\text{suf},P} \ldots \tau_{\text{suf},C} \ldots$$

$$= \tau_{\text{pre}}^{[\tau_{\text{suf}]^{\omega}}}.$$  \hfill (3.11)

Projecting the global plan (3.11) on robot $i$, we get that (3.8) holds for all robots $i \in \mathcal{N}$, which completes the proof.

In the following proposition, we examine the optimality of Algorithm 3. Specifically, we examine how the cost $\text{cost}(k) = \sum_{i=1}^{N} J(\tau_{i}^{\text{suf},n_i(k)})$ changes with respect to iterations $n_i(k)$.

**Proposition 3.5.2 (Optimality).** Let $\tau_{i}^{\text{suf},n_i(k)}$ denote the suffix part for robot $i$ at iteration $k$ of Algorithm 3 for some $n_i(k) \geq 2$. Algorithm 3 updates the communication points in $\tau_{i}^{\text{suf},n_i(k)}$ so that the total cost $\text{cost}(k) = \sum_{i=1}^{N} J(\tau_{i}^{\text{suf},n_i(k)})$ incurred due to a single execution of the suffix parts $\tau_{i}^{\text{suf},n_i(k)}$ decreases with respect to $k - 1$, when $n_i(k) \geq 2$, for all $i \in \mathcal{N}$.

**Proof.** Assume that $n_i(k) \geq 2$ for all $i \in \mathcal{N}$ and that robots in team $\mathcal{T}_m$ are responsible for updating the respective communication point at iteration $k$. The suffix parts $\tau_{i}^{\text{suf},n_i(k)}$ and $\tau_{i}^{\text{suf},n_i(k-1)}$ denote the suffix parts of robot $i \in \mathcal{T}_m$ before and after updating the communication point for team $\mathcal{T}_m$. Also, define $d_i(k) = J(\tau_{i}^{\text{suf},n_i(k)}) - J(\tau_{i}^{\text{suf},n_i(k-1)})$ that captures the difference of the cost between the suffix parts $\tau_{i}^{\text{suf},n_i(k)}$ and $\tau_{i}^{\text{suf},n_i(k-1)}$ due to the update of the communication point. For example, if $d_i(k) < 0$ this means that $J(\tau_{i}^{\text{suf},n_i(k)}) < J(\tau_{i}^{\text{suf},n_i(k-1)})$, i.e., the cost associated with robot $i$ has decreased due to the update of a communication point at
iteration $k$. Notice that for the function $d_i(k)$ it holds that

$$\sum_{i \in N} d_i(k) = \text{cost}(k) - \text{cost}(k - 1). \quad (3.12)$$

Thus, to prove this result, it suffices to show that $\text{cost}(k) \leq \text{cost}(k - 1)$, for all $k \geq 2$ or equivalently that

$$\sum_{i \in N} d_i(k) \leq 0, \forall k \geq 2. \quad (3.13)$$

First, we express the term $\sum_{i \in N} d_i(k)$ as

$$\sum_{i \in N} d_i(k) = \sum_{i \in T_m} d_i(k) + \sum_{j \in N \setminus T_m} d_j(k). \quad (3.14)$$

Observe that $\sum_{j \in N \setminus T_m} d_j(k) = 0$, since at iteration $k$ all robots $j \in N \setminus T_m$ do not update their respective suffix parts, by construction of Algorithm 3. Thus, it suffices to show that $\sum_{i \in T_m} d_i(k) \leq 0, \forall k \geq 2$.

Recall that in the path $\tau_{i}^{\text{suf},n_i(k-1)}$ there already exists a state associated with a communication point for team $T_m$. Also, notice that the only difference between $\tau_{i}^{\text{suf},n_i(k-1)}$ and $\tau_{i}^{\text{suf},n_i(k)}$ lies in the state associated with a communication point for team $T_m$. Hence, with slight abuse of notation, before updating this communication point, it holds that $\tau_{i}^{\text{suf},n_i(k)} \equiv \tau_{i}^{\text{suf},n_i(k-1)}$ and, consequently, $J(\tau_{i}^{\text{suf},n_i(k)}) = J(\tau_{i}^{\text{suf},n_i(k-1)})$. In order to construct $\tau_{i}^{\text{suf},n_i(k)}$, the aforementioned state is updated so that the total cost for team $T_m$, captured by $\sum_{i \in T_m} J(\tau_{i}^{\text{suf},n_i(k)})$, is minimized, as per (3.7). Notice that the communication point for team $T_m$ and its respective entry in $\tau_{i}^{\text{suf},n_i(k-1)}$ is a feasible point for (3.7) for the construction of $\tau_{i}^{\text{suf},n_i(k)}$, since before updating them, it holds that $\tau_{i}^{\text{suf},n_i(k)} \equiv \tau_{i}^{\text{suf},n_i(k-1)}$, as previously discussed. In case this feasible point is the solution of (3.7) for the construction of the suffix parts $\tau_{i}^{\text{suf},n_i(k)}, \forall i \in T_m$, then after updating the respective communication point, it holds
that: $J(\tau_{i}^{\text{suf},n_{i}(k)}) = J(\tau_{i}^{\text{suf},n_{i}(k-1)})$, otherwise, $J(\tau_{i}^{\text{suf},n_{i}(k)}) < J(\tau_{i}^{\text{suf},n_{i}(k-1)})$. Consequently, in any case it holds that $J(\tau_{i}^{\text{suf},n_{i}(k)}) \leq J(\tau_{i}^{\text{suf},n_{i}(k-1)})$, i.e., $\sum_{i \in \mathcal{T}_m} d_i(k) \leq 0$, $\forall k \geq 2$, which completes the proof.

Notice that the total cost $\text{cost}(k)$ increases for all iterations $k$ associated with the construction of the suffix part $\tau_{i}^{\text{suf},2}$. The reason is that the suffix parts $\tau_{i}^{\text{suf},2}$ are initialized as $\tau_{i}^{\text{suf},2} = \tau_{i}^{\text{suf},1}$, where $\tau_{i}^{\text{suf},1}$ do not contain any states associated with communication points. Thus, when such states are introduced in the suffix parts $\tau_{i}^{\text{suf},2}$ the total cost $\text{cost}(k)$ is expected to increase.

**Remark 3.5.3** (Optimality of Algorithm 3). Notice that Proposition 3.5.2 does not guarantee that Algorithm 3 will find the optimal plan that minimizes the cost $J_p(\tau_{i}) = \alpha \sum_{i \in \mathcal{N}} J(\tau_{i}^{\text{pre}}) + (1 - \alpha) \sum_{i \in \mathcal{N}} J(\tau_{i}^{\text{suf}})$. Instead it only ensures that the total cost $\text{cost}(k) = \sum_{i \in \mathcal{N}} J(\tau_{i}^{\text{suf},n_{i}(k)})$ of the revised suffix parts decreases with every iteration $k$ until the algorithm converges to a plan in prefix-suffix structure as shown in Proposition 3.5.1. Sub-optimality is due to the decomposition of the global LTL formula (3.2) into task planning (Section 3.3) and intermittent communication control (Chapter 2) that are solved independently. The optimal plan can be found by translating the global LTL formula (3.2) into a NBA, constructing a product automaton across all robots of the network as, e.g., in Ulusoy et al. (2013, 2014), and using graph search methods to find the optimal plan. However, such centralized methods are computationally expensive and resource demanding. Moreover, recall that in this chapter we assume that the teams $\mathcal{T}_m$ are fixed and never change. Note that the total cost of the revised plans can be further minimized if the robots in every team $\mathcal{T}_m$ update not only the communication point $v_j$, $j \in \mathcal{C}_m$, but also the teams they belong to. Optimal design of the teams is discussed in Chapter 6.

Next, to show that the local tasks $\phi_i$ and the intermittent connectivity requirement $\phi_{\text{com}}$ are satisfied, we first need to show that the network is deadlock-free when
the motion plans $\tilde{\tau}_i$ are executed according to Algorithm 4. Specifically, we assume that there is a deadlock, if there are robots of any team $\mathcal{T}_m$ that are waiting forever at a communication point, selected by Algorithm 3, for the arrival of all other robots of team $\mathcal{T}_m$.

**Proposition 3.5.4** (Deadlock-free). The mobile robot network is deadlock-free when the motion plans $\tilde{\tau}_i$ generated by Algorithm 3 are executed according to Algorithm 4.

**Proof.** Let $\mathcal{W}_{v_e} \subset \mathcal{T}_m$ denote the set of robots that are waiting at communication point $v_e$, $e \in \mathcal{C}_m$, selected by Algorithm 3, for the arrival of the other robots that belong to team $\mathcal{T}_m$. Assume that the robots in $\mathcal{T}_m \setminus \mathcal{W}_{v_e}$ never arrive at that node so that communication at node $v_e$ for team $\mathcal{T}_m$ never occurs. This means that the robots in $\mathcal{T}_m \setminus \mathcal{W}_{v_e}$ are waiting indefinitely at communication locations $v_j \in \mathcal{C}_n$, $j \neq e$, $n \neq m$, $n \in \mathcal{N}_{\mathcal{T}_m}$, selected by Algorithm 3, to communicate with robots in team $\mathcal{T}_n$. The fact that there are robots that remain indefinitely at node $v_j \in \mathcal{C}_n$ means that a communication within team $\mathcal{T}_n$ never occurs by construction of Algorithm 4. Following an argument similar to the above, we conclude that the robots in $\mathcal{T}_n \setminus \mathcal{W}_{v_j}$ are waiting indefinitely at nodes $v_k \neq j \in \mathcal{C}_f$ to communicate with robots that belong to a team $\mathcal{T}_f$, $f \in \mathcal{N}_{\mathcal{T}_n}$. Therefore, if a communication event never occurs for team $\mathcal{T}_m$, then all robots $i \in \mathcal{N}$ need to be waiting at the communication locations selected by Algorithm 3 and, consequently, there is no communication location where all robots are present, i.e., there is no team within which communication will ever occur. Throughout the rest of the proof we will refer to this network configuration as a stationary configuration.

In what follows, we show that the network can never reach a stationary configuration when the motion plans $\tilde{\tau}_i$ are executed asynchronously as per Algorithm 4. First, recall that due to the last two constraints in the optimization problem (3.7) states $q^{v_j}_i$ associated with communication points $j \in \mathcal{C}_m$, for all $m \in \mathcal{M}_i$ are
introduced in $\tilde{\tau}_i$ in the order that teams $T_m, m \in \mathcal{M}_i$ appear in the schedule of communication events $\text{sched}_i$. Second, note that according to Algorithm 4, during the execution of the plans $\tilde{\tau}_i$, robots $i$ wait only at communication points $v_j, j \in C_m, m \in \mathcal{M}_i$, selected by Algorithm 3, until all the other robots within team $T_m$ arrive at $v_j$, exactly as in the asynchronous execution of the schedules $\text{sched}_i$ discussed in Algorithm 2; see also Remark 3.4.4. These two observations along with the fact that the network is deadlock-free under the asynchronous execution of the schedules $\text{sched}_i$, as per Algorithm 2 shown in Proposition 2.4.2, entail that the network is deadlock-free, as well, when all motion plans $\tilde{\tau}_i$ are executed asynchronously, as per Algorithm 4, which completes the proof.

**Remark 3.5.5** (Bounded waiting times). Proposition 3.5.4 shows also that the waiting times of robots at the communication points that were introduced by Algorithm 4 are bounded.

**Theorem 3.5.6** (LTL Tasks). Assume that there is always a transition in $wTS_i$ from any state $q_{i,v}^j$ to any other state $q_{i,v}^e$ associated with a communication point $v_e, e \in C_m, m \in \mathcal{M}_i$ and for all robots $i \in \mathcal{N}$. Also, consider the motion plans $\tilde{\tau}_i$ generated by Algorithm 3. The asynchronous execution of the motion plans $\tilde{\tau}_i$ as per Algorithm 4 satisfies the LTL statements $\phi_i$, i.e., $\tilde{\tau}_i \models \phi_i$, for all robots $i$.

**Proof.** First recall that the waiting times at the communication points introduced in $\tilde{\tau}_i$ by Algorithm 3 are bounded due to Proposition 3.5.4. Consequently, all robots $i$ will eventually execute the suffix parts $\tau_i^{\text{suf}, n_i}, \forall n_i \in \{1, \ldots, P\}$ that appear in $\tilde{\tau}_i$ without getting trapped indefinitely in a deadlock configuration.

Next, to show this result, first we need to show that all transitions in $wTS_i$ that are due to $\tilde{\tau}_i$ respect the transition rule $\rightarrow_i$; see Definition 3.2.1. Next, we need to show that the infinite run $\rho_{B_i}$ of the NBA $B_i$ that corresponds to $\phi_i$ over word $\sigma$
generated during the execution of $\widetilde{\tau}_i$ is accepting, i.e.,

$$\text{Inf}(\rho_{B_i}) \cap F_{B_i} \neq \emptyset. \quad (3.15)$$

Notice that all transitions incurred by the finite path $\tau_i^{\text{pre}, \tau_i^{\text{suf},1}}$ respect the transition rule $\rightarrow_i$, by construction of the motion plan $\tau_i = \tau_i^\text{pre}[\tau_i^{\text{suf},1}]^\omega \models \phi_i$. Also, by assumption, there is always a transition from any state $q_i^{v_e}$ to any other state $q_i^{v_e}$ associated with a communication point $v_e, e \in C_m$, for all $m \in \mathcal{M}_i$. This implies that all transitions incurred by any suffix part $\tau_i^{\text{suf},n_i}, n_i \geq 2$ respect the transition rule $\rightarrow_i$. Also, transitions from the last state of the suffix part $\tau_i^{\text{suf},n_i}$ to the first state of $\tau_i^{\text{suf},n_i+1}$ always respect the transition rule $\rightarrow_i$. The reason is that $\tau_i^{\text{suf},n_i+1}(1)$ either is associated with a communication point and in this case the transition rule $\rightarrow_i$ is satisfied by assumption, or is the same as $\tau_i^{\text{suf},n_i}(1)$, by construction of $\tau_i^{\text{suf},n_i}$, and in this case the transition rule $\rightarrow_i$ is trivially satisfied.

Next, we show that (3.15) holds. To show this result, first recall that $\tau_i^{\text{pre}}$ is the projection of a path $p_i^{\text{pre}}$ onto $w_{TS_i}$ that lives in the state-space $Q_{P_i}$ of the PBA defined in Definition 3.3.4 that connects an initial state $q_{P_i}^{0} \in Q_{P_i}^0$ to a final state $q_{P_i}^{F} \in F_{P_i}$. Also, notice that $\tau_i^{\text{pre}}[\tau_i^{\text{suf},n_i}]^\omega \models \phi_i$, for all $n_i \in \{1, \ldots, P, \ldots, C\}$, by construction of $\tau_i^{\text{suf},n_i}$, i.e., $\text{Inf}(\rho_{B_i}^{n_i}) \cap F_{B_i} \neq \emptyset$, where $\rho_{B_i}^{n_i}$ is the infinite run of $B_i$ over the word $\sigma$. This means all the suffix parts $\tau_i^{\text{suf},n_i}$ are projections of paths $p_i^{\text{suf},n_i}$ that live in the state-space $Q_{P_i}$ of the PBA and constitute cycles around the final state $q_{P_i}^{F} \in F_{P_i}$. Thus, the concatenation $p_i^{\text{pre}} = p_i^{\text{pre}}p_i^{\text{suf},1}\ldots p_i^{\text{suf},P-1}$ is still a path that connects an initial state $q_{P_i}^{0} \in Q_{P_i}^0$ to the final state $q_{P_i}^{F} \in F_{P_i}$, as $p_i^{\text{pre}}$ does. Notice that $p_i^{\text{pre}}$ visits only once the final state $q_{P_i}^{F}$ while $\overline{p_i^{\text{pre}}}$ visits the same final state $P$ times. Similarly, $\overline{p_i^{\text{suf}}} = p_i^{\text{suf},P} \overline{p_i^{\text{suf},1}} \ldots \overline{p_i^{\text{suf},P+C}}$ is a cyclic path around the final state $q_{P_i}^{F}$, as each suffix part $p_i^{\text{suf},n_i}$ is. Also, observe that $p_i^{\text{suf},n_i}$ is a single cycle

---

4 The generated word $\sigma$, called also trace of $\widetilde{\tau}_i$ Clarke et al. (1999) and denoted by $\text{trace}(\widetilde{\tau}_i)$, is defined as $\sigma = \text{trace}(\widetilde{\tau}_i) := L_i(\widetilde{\tau}_i(1))L_i(\widetilde{\tau}_i(2))\ldots$, where $L_i$ is the labeling function defined in Definition 3.2.1.
around the final state $q_F^i$, while $\tilde{p}_{i}^{\text{inf}}$ can be viewed as $C + 1$ cycles around this final state. Thus, we get that the path $\tilde{p}_i = \tilde{p}_i^\text{pre}[\tilde{p}_i^\text{inf}]^\omega$ is a path in $Q_i^P$ that visits the final state $q_F^i \in \mathcal{F}_P$ infinitely often. Recall, that $\tilde{\tau}_i$ is the projection of $\tilde{p}_i$ onto the state-space $Q_i$ of $\text{wTS}_i$, i.e., $\tilde{\tau}_i = \Pi|_{\text{wTS},\tilde{p}_i}$. Thus, we get that the infinite run of $B_i$ over the word $\sigma$ satisfies (3.15) completing the proof.

**Remark 3.5.7** (Theorem 3.5.6). In Theorem 3.5.6 the assumption that there is always a transition in $\text{wTS}_i$ from any state $q_i^{v_j}$ to any other state $q_i^{v_g}$ associated with a communication point $v_e, e \in C_m$, for all $m \in M_i$ and for all robots $i \in N$ guarantees that the motion plans $\tilde{\tau}_i$ satisfy the transition rule $\rightarrow_i$ of $\text{wTS}_i$, $\forall i \in N$, as shown in its proof. In other words, this assumption implies that any communication point $v_e, e \in C_m$ can be reached by any other location $v_j, j \in I$ in the workspace without going through any other location $v_g, g \neq j, g \in I$. This assumption can be relaxed if the optimization problem (3.7) is not solved with respect to communication points $v_j, \forall j \in C_m$ but with respect to finite paths that (i) connect any two consecutive states in $\tilde{\tau}_i$, (ii) abide by the transition rule $\rightarrow_i$, and (iii) go through a communication point $v_j, \forall j \in C_m$.

**Theorem 3.5.8** (Intermittent Communication). Consider the motion plans $\tilde{\tau}_i$ generated by Algorithm 3. The asynchronous execution of the motion plans $\tilde{\tau}_i$ as per Algorithm 4 satisfies the intermittent communication requirement captured by the global LTL statement $\phi_{\text{com}}$.

**Proof.** The continuous time execution of the plans $\tilde{\tau}_i$ as per Algorithm 4 ensures that all robots within team $T_m$ wait for each other at a common communication point selected by Algorithm 3 until communication within team $T_m$ happens, for all $m \in M$. Due to Proposition 3.5.4, it holds that the waiting times of robots at the communication points are bounded and, therefore, the network is deadlock-free under the execution of the plans $\tilde{\tau}_i = \tilde{\tau}_i^\text{pre}[\tilde{\tau}_i^\text{inf}]^\omega$. Also, since by construction there
is at least one communication point for team $T_m$, $m \in M$, for all robots $i \in \mathcal{N}$ in the suffix part $\tilde{\tau}^{\text{suf}}_i$, which is executed indefinitely, communication within any team $T_m$ occurs infinitely often, which completes the proof. \qed

**Theorem 3.5.9** (Task and Intermittent Communication). Consider the motion plans $\tilde{\tau}_i$ generated by Algorithm 3. The asynchronous execution of the motion plans $\tilde{\tau}_i$ as per Algorithm 4 satisfies the global LTL statement $\phi = (\wedge_{i \in \mathcal{N}} \phi_i) \land \phi_{\text{com}}$.

**Proof.** First define the global plan: $\tilde{\tau} = [\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_N]^T$. The continuous time execution of the global plan $\tilde{\tau}$ as per Algorithm 4 satisfies $\tilde{\tau} \models (\wedge_{i \in \mathcal{N}} \phi_i)$ due to Theorem 3.5.6 and $\tilde{\tau} \models \phi_{\text{com}}$ due to Theorem 3.5.8. By definition of the conjunction operator Baier and Katoen (2008), it holds that $\tilde{\tau} \models (\wedge_{i \in \mathcal{N}} \phi_i) \land \phi_{\text{com}}$ which completes the proof. \qed

**Remark 3.5.10** (More compact form of motion plans $\tilde{\tau}_i$). Recall from Remark 3.4.1 that there is an iteration index $k$ such that all robots have the same iteration index $n_i$, i.e., $n_i = n$ for some $n \in \mathbb{N}_+$ for all robots $i \in \mathcal{N}$ and for all $n_i \in \mathbb{N}_+$. Then, Proposition 3.5.2 implies that the total cost incurred by a single execution of all revised suffix parts $\tau_i^{\text{suf}, n}$, captured by $\sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, n})$, decreases with respect to iterations $n$. Combining this result with Proposition 3.5.1, we conclude that $\sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, 1}) \leq \sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, 2}) \leq \cdots \leq \sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, P-1}) \leq \sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, P})$ and $\sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, P}) = \sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, P+1}) = \cdots = \sum_{i \in \mathcal{N}} J(\tau_i^{\text{suf}, C})$. Thus, we can eliminate from $\tilde{\tau}_i$ all the suffix parts $\tau_i^{\text{suf}, n}$, for $n = 1, \ldots, P-1$, and keep the suffix parts $\tau_i^{\text{suf}, n}$, for $n = P, \ldots, C$ that incur the minimum total cost. Thus, we can rewrite the plans $\tilde{\tau}_i$ as $\tilde{\tau}_i = \tau_i^{\text{pre}}[\tau_i^{\text{suf}, P} \ldots \tau_i^{\text{suf}, C}]^\omega$. The results shown in Theorems 3.5.6 and 3.5.8 hold for the compact form of the plans as well.
3.6 Simulation Studies

In this section, a simulation study is provided that illustrates our approach for a network of $N = 10$ robots. Robots are categorized into $M = 10$ teams as follows: $T_1 = \{1, 2\}$, $T_2 = \{2, 3\}$, $T_3 = \{3, 4\}$, $T_4 = \{2, 4, 5\}$, $T_5 = \{1, 5\}$, $T_6 = \{5, 6\}$, $T_7 = \{6, 7, 9\}$, $T_8 = \{7, 8\}$, $T_9 = \{8, 9\}$, and $T_{10} = \{9, 10\}$. Notice that the construction of teams $T_m$ results in a connected graph $G_T$, shown in Figure 3.5, with $\ell = \max\{d_{T_m}\}_{m=1}^M + 1 = 5$, as discussed in Section 5.1. Mobility of each robot in the workspace is captured by a wTS with $|Q_i| = 140$ states that represent $W = 140$ locations of interest, illustrated in Figure 4.3.1, and weights $w_i$ that capture the distance between its states. Among the $W = 140$ locations of interest, $R = 100$ locations correspond to possible communication points. Also, among the available $R = 100$ communication points, we assume that for each team $T_m$ there are ten available communication points, i.e., $|C_m| = 10$, for all $m \in \mathcal{M}$ and that $C_m \cap C_n = \emptyset$, for all $m, n \in \mathcal{M}$. Specifically, we consider the following sets of indices for the available communication points for every team $T_m$: $C_m = \{41 + 10(m-1), 41 + 10(m-1) + 1, \ldots, 41 + 10(m-1) + 9\}$, $\forall m \in \mathcal{M}$. Also, the parameter $\alpha$ in (3.3) is selected as $\alpha = 0.5$.

To model uncertainty in robot mobility, caused by exogenous disturbances that may affect the arrival times of the robots at the communication locations, we assume that the time required for robot $i$ to travel from location $v_{e_i}$ to $v_{j_i}$, with $(q_{i_e}^{v_{e_i}}, q_{i_j}^{v_{j_i}}) \in \rightarrow_i$, is denoted by $t_{e_j}^{i}$ and is generated by a uniform distribution on $[1, 2]$, at the moment when robot $i$ arrives at location $v_{e_i}$.

The LTL tasks for robots 1, 6, 7, and 8 are

$$
\phi_1 = (\square \diamond \pi_{1_e}^{v_{e_1}}) \land (\square \diamond \pi_{1_e}^{v_{e_8}}) \land (\square \diamond \pi_{1_e}^{v_{36}}) \land (\square \diamond \pi_{1_e}^{v_{21}}) \\
\land (\diamond \pi_{1_e}^{v_{2}}) \land (\neg \pi_{1_e}^{v_{2}} \cup \pi_{1_e}^{v_{8}})
$$

(3.16)
After constructing the initial motion plans \( \tau_i = \tau_i^{\text{pre}}[\tau_i^{\text{suf}}]^{\omega} \), for all robots \( i \), schedules of communication events are constructed as per Algorithm 1. The resulting schedules have the following form:

\[
\begin{align*}
sched_1 &= [X, 1, 5, X, X]^{\omega}, \quad sched_6 = [X, 6, X, 7, X]^{\omega}, \\
sched_2 &= [X, 1, 2, 4, X]^{\omega}, \quad sched_7 = [X, 8, X, 7, X]^{\omega}, \\
sched_3 &= [X, 3, 2, X, X]^{\omega}, \quad sched_8 = [X, 9, 8, X, X]^{\omega}, \\
sched_4 &= [X, 3, 4, X, X]^{\omega}, \quad sched_9 = [X, 10, 9, 7, X]^{\omega}, \\
sched_5 &= [X, 6, 5, 4, X]^{\omega}, \quad sched_{10} = [X, 10, X, X, X]^{\omega}.
\end{align*}
\]
Then, given the above meeting schedules, Algorithm 3 constructs the revised motion plans $\tilde{\tau}_i$, which have the following structure

\[
\tilde{\tau}_i = \tilde{\tau}_{i}^{\text{pre}} \left[ \tilde{\tau}_{i}^{\text{suf}} \right] = \tau_{i}^{\text{pre\_suf\_1}} \tilde{\tau}_{i}^{\text{suf\_2}} \tilde{\tau}_{i}^{\text{suf\_3}} \tilde{\tau}_{i}^{\text{suf\_4}} \omega,
\]

for all robots $i \in \mathcal{N}$. The suffix parts $\tau_{i}^{\text{suf\_ns}}$ for robot $7 \in \mathcal{T}_7 \cap \mathcal{T}_8$ have the following form:

\[
\tau_{7}^{\text{suf\_1}} = q_{7}^{V_8} q_{7}^{V_9} q_{7}^{V_10} q_{7}^{V_1} q_{7}^{V_9} q_{7}^{V_10} q_{7}^{V_24} q_{7}^{V_2} q_{7}^{V_34},
\]

\[
\tau_{7}^{\text{suf\_2}} = q_{7}^{V_8} q_{7}^{V_114} q_{7}^{V_9} q_{7}^{V_10} q_{7}^{V_1} q_{7}^{V_10} q_{7}^{V_12} q_{7}^{V_{19}} q_{7}^{V_{21}},
\]

\[
\tau_{7}^{\text{suf\_3}} = q_{7}^{V_8} q_{7}^{V_115} q_{7}^{V_9} q_{7}^{V_10} q_{7}^{V_1} q_{7}^{V_10} q_{7}^{V_12} q_{7}^{V_{19}} q_{7}^{V_{21}},
\]

\[
\tau_{7}^{\text{suf\_4}} = q_{7}^{V_8} q_{7}^{V_114} q_{7}^{V_9} q_{7}^{V_10} q_{7}^{V_1} q_{7}^{V_10} q_{7}^{V_12} q_{7}^{V_{19}} q_{7}^{V_{21}},
\]

Similarly, the suffix parts $\tau_{8}^{\text{suf\_ns}}$ for robot $8 \in \mathcal{T}_8 \cap \mathcal{T}_9$ have the following form:

\[
\tau_{8}^{\text{suf\_1}} = \tau_{8}^{\text{suf}} = q_{8}^{V_2} q_{8}^{V_3} q_{8}^{V_4} q_{8}^{V_5} q_{8}^{V_6} q_{8}^{V_7} q_{8}^{V_8} q_{8}^{V_9},
\]

\[
\tau_{8}^{\text{suf\_2}} = q_{8}^{V_2} q_{8}^{V_114} q_{8}^{V_3} q_{8}^{V_4} q_{8}^{V_5} q_{8}^{V_130} q_{8}^{V_6} q_{8}^{V_9} q_{8}^{V_{12}} q_{8}^{V_8},
\]

\[
\tau_{8}^{\text{suf\_3}} = \tau_{8}^{\text{suf}} = q_{8}^{V_2} q_{8}^{V_3} q_{8}^{V_4} q_{8}^{V_5} q_{8}^{V_{115}} q_{8}^{V_{124}} q_{8}^{V_6} q_{8}^{V_9} q_{8}^{V_{12}} q_{8}^{V_8},
\]

\[
\tau_{8}^{\text{suf\_4}} = \tau_{8}^{\text{suf}} = q_{8}^{V_2} q_{8}^{V_3} q_{8}^{V_4} q_{8}^{V_5} q_{8}^{V_{130}} q_{8}^{V_6} q_{8}^{V_9} q_{8}^{V_{12}} q_{8}^{V_8}.
\]
Figure 3.6: Intermittent communication of $N = 10$ robots. Black squares and colored circles comprise the set $\mathcal{I}$. Black Boxes stand for locations $v_j$ with $j \in \mathcal{I} \setminus \mathcal{C}$. The colored circles comprise the set $\mathcal{C}$, i.e., stand for communication points, rectangles represent the robots, and the blue rhombus stands for a user. We assume that there is direct transition from any location $v_k$ to any other communication point $v_j$, $j \in \mathcal{C}$. Each color is associated with a unique team, i.e., communication points with the same color belong to the same set $\mathcal{C}_m$ and colors inside a rectangle correspond to the teams that this robot belongs. Enlarged colored circles stand for the selected communication points. The blue and the yellow lines correspond to the suffix parts of robots 7 and 8, respectively, that belong to team $\mathcal{T}_8$. Also, Figures 3.6(a), 3.6(b), and 3.6(c) show the suffix parts at iterations $n_i = 2$, $n_i = 3$, and $n_i = 4$, respectively, for robots 7 and 8. The red star on top of the robots stands for information initially collected by robot 1, which needs to be transmitted to the user.

Notice that in $\tau^\text{snf,n}_7$, for all $n_7 \geq 2$, the communication point for team $\mathcal{T}_8$ appears before the communication point for team $\mathcal{T}_7$, as dictated by the schedule $\text{sched}_7 = [8, X, 7]^{\omega}$ and the constraints of the optimization (3.7). Similar observations can be made for the suffix part of robot 8. Figure 4.3.1 shows the revised suffix parts for robots 7 and 8 at consecutive iterations $n_i$ of Algorithm 3. Also, observe in Figure 4.3.1, that as time goes by, the communication points are updated and the final communication point selected for each team is depicted in Figure 3.6(c).

To illustrate that the proposed motion plans ensure intermittent communication among the robots infinitely often, we consider the following two test cases. First, we implement a consensus algorithm over the dynamic network $\mathcal{G}_c$. Specifically, we assume that initially all robots generate a random number $v_i(t_0)$ and when all robots $i \in \mathcal{T}_m$ meet at a rendezvous point $j \in \mathcal{C}_m$ they perform the following consensus
update $v_i(t) = \frac{1}{|\mathcal{T}_m|} \sum_{e \in \mathcal{T}_m} v_e(t)$. Figure 5.4(b) shows that eventually all robots reach a consensus on the numbers $v_i(t)$, which means that communication among robots takes place infinitely often, as proven in Theorem 3.5.8. Second we assume that robot 1 carries a message, illustrated by a red star in Figure 4.3.1, which needs to be transmitted to a user located at $v_{19}$ which is visited infinitely often by robot 7 according to $\phi_7$. This message is transmitted to all robots of any team $\mathcal{T}_m$, $m \in \mathcal{M}$ during a communication event for this team, only if there is at least one robot $i \in \mathcal{T}_m$ that carries it. Figures 3.6(a)-3.6(c) depict the successful transmission of the information collected by robot 1 to the user.

The communication events over time for teams $\mathcal{T}_1$ and $\mathcal{T}_5$ are depicted in Figure 3.8. Observe in Figure 3.8 that the communication time instances do not depend linearly on time, which means that communication within these teams is aperiodic. Figure 3.9 shows that the total traveled distance $\sum_{i=1}^{N} J(\tau_i^{\text{suf},n_i})$ due to a single execution of the suffix parts $\tau_i^{\text{suf},n_i}$ as communication points are updated decreases, if $n_i \geq 2$, as shown in Proposition 3.5.2.

Note also that due to excessive memory requirements it would be impossible to generate optimal motion plans $\tilde{\tau}_i$ by using either existing optimal control synthesis methods Kantaros and Zavlanos (2015); Ulusoy et al. (2013, 2014) that rely on the
construction of a synchronous product automaton or off-the-shelf model checkers Cimatti et al. (2002); Holzmann (2004) that can construct feasible but not optimal paths. Specifically, Kantaros and Zavlanos (2015); Ulusoy et al. (2013, 2014) rely on the construction of a product transition system (PTS), whose state space has dimension $|Q_{PTS}| = \times_{\forall i} |Q_i| = W^{|N|} = 140^{10} = 2.8925 \times 10^{21}$. This PTS is combined with the Büchi Automaton $B$ that corresponds to the LTL statement $\phi = (\land_{\forall i \in N} \phi_i) \land \phi_{com}$ to construct a Product Büchi Automaton whose state space has dimension $|Q_{PBA}| = |Q_{PTS}| \times |Q_B| = 2.8925 \times 10^{21} \times |Q_B|$ which is too large to manipulate in practice let alone searching for an optimal accepting infinite run.

3.7 Experimental Results

In this section, we present experimental results to validate the proposed task planning and intermittent communication control framework. Four differential-driven “iRobot Create” robots are deployed within a $2.5m \times 2.0m$ workspace as shown in Figure 3.10, whose positions and orientations can be tracked in real-time via an Optitrack motion capture system. The communication among the coordination module, the robot actuation module, and the Optitrack is handled via the Robot operating system (ROS).
Figure 3.9: Evolution of the total cost $\sum_{i=1}^{10} J(\tau_{i}^{suf,n_i})$ with respect to iterations $k$ and $n_i$. Every time a communication point is updated the total traveled distance $\sum_{i=1}^{10} J(\tau_{i}^{suf,n_i})$ decreases. Red dots correspond to the total cost after a single execution over the sequence $S$, i.e., when the iteration index $n_i$ is common for all robots $i \in \mathcal{N}$. Notice that the total cost $\sum_{i=1}^{10} J(\tau_{i}^{suf,n_i})$ do not change after $n_i \geq 4$, since Algorithm 3 converged at iteration $n_i = 4$.

3.7.1 System Description

The robots start from positions $(0.25, 0.25), (0.25, 0.50), (0.25, 0.75), (0.25, 1.0)$, respectively at time $t = 0$. The communication range is set to $0.6m$ uniformly for all robots. Robot $R_1$ has the task to visit locations $v_2 = (0.75, 0.25), v_4 = (1.75, 0.25)$, and $v_{11} = (0.25, 1.25)$, infinitely often, which is described by the LTL formula $\phi_1 = (\square \Diamond \pi_1^{v_2}) \land (\square \Diamond \pi_1^{v_4}) \land (\square \Diamond \pi_1^{v_{11}})$. Robot $R_2$ has the task to visit locations $v_5 = (2.25, 0.25), v_{12} = (0.75, 1.25)$, and $v_1 = (0.25, 0.25)$ infinitely often, which is described by the LTL formula $\phi_2 = (\square \Diamond \pi_2^{v_5}) \land (\square \Diamond \pi_2^{v_{12}}) \land (\square \Diamond \pi_2^{v_{12}})$. Robot $R_3$ has the task to visit locations $v_6 = (0.25, 0.75), v_8 = (1.25, 0.75)$, and $v_{19} = (1.75, 1.75)$ infinitely often, which is described by the LTL formula $\phi_3 = (\square \Diamond \pi_3^{v_6}) \land (\square \Diamond \pi_3^{v_8}) \land (\square \Diamond \pi_3^{v_{19}})$. Robot $R_4$ has the task to $v_9 = (1.75, 0.75), v_{16} = (0.25, 1.75)$, and $v_{18} = (1.25, 1.75)$, which is described by the LTL formula
Figure 3.10: Workspace layout for the experiment study in Section 3.7. On the left is the actual workspace. The waypoints and communication points are shown on the floor. The tripod represents the virtual user. On the right is the visualization panel to monitor Robots $R_1, R_2, R_3, R_4$ are in cyan, green, blue and magenta, respectively. The messages stored at each robot are indicated by filled stars. The message sending and receiving actions are shown by filled green text boxes. The connected lines represent each robot’s revised suffix from Algorithm 3. The full experiment video can be found in ExperimentVideo (2017).

\[ \phi_4 = (\Box \Diamond \pi_4^{\forall_3}) \land (\Box \Diamond \pi_4^{\forall_{16}}) \land (\Box \Diamond \pi_4^{\forall_{18}}). \]

Furthermore, there are three teams within the system: team one $T_1 = \{R_1, R_2\}$, team two $T_2 = \{R_1, R_3\}$ and team three $T_3 = \{R_3, R_4\}$. A virtual user is placed at $v_6 = (0.25m, 0.5m)$, as shown in Figure 3.10, and robot $R_3$ is responsible for visiting this user infinitely often as dictated by $\phi_3$. The messages generated by every robot are indicated by filled stars, as shown in the visualization panel of Figure 3.10. Each time two robots meet at a communication point, they can exchange their stored messages. For instance, robot $R_1$ has messages $m_1$ and $m_2$ while $R_2$ has messages $m_2$ and $m_3$. When they meet and exchange messages, afterwards they both have the messages $m_1$, $m_2$, and $m_3$.

As shown in Figure 3.10, the size of an iRobot is around $0.4m$ in diameter. Given the confined workspace, a local reactive collision avoidance scheme is needed for the actual plan execution. We rely on the framework of reciprocal velocity obstacles (RVO) introduced in Van den Berg et al. (2008) to resolve inter-robot collision locally among the robots. Particularly, the relative position and relative velocity between each robot and its neighboring robots are used to compute the desired velocity that
would avoid future collisions. This is done for each robot at each time step.

3.7.2 Experimental Results

Following the procedure described in Section 3.3, each robot constructs a local plan \( \tau_i \) that satisfies its task. For instance, the wTS\(_1\) of robot \( R_1 \) has 25 states and 25\(^2\) transitions, i.e., there is a direct transition among any two states. Moreover, the NBA \( B_1 \) has 4 states and 13 transitions and its product \( P_1 \) has 25 \( \times \) 4 = 100 states and 1680 transitions. It took around 0.75s for the model-checking algorithm, described in Section 3.3, to find \( \tau_1 \) with the minimal total cost \( J(\tau_1) \). Hereafter, for the sake of simplicity, we write the motion plans in terms of the coordinates of locations that robots need to visit instead of using the state of the wTS. For example, the discrete plan \( \tau_1 \) is given by \( \tau_1 = \tau_1^{\text{pre}}[\tau_1^{\text{suf}}]^{\omega} \), where \( \tau_1^{\text{pre}} = [(0.25, 0.25)(0.75, 0.25)(0.25, 1.25)(1.75, 0.25)] \) and \( \tau_1^{\text{suf}} = [(1.75, 0.25)(0.25, 1.25)(0.75, 0.25)] \).

Next, based on Section 3.4, the initial motion plans \( \tau_i \) of all robots are revised to meet the intermittent communication constraints. Specifically, motion plans of the form \( \tilde{\tau}_i = \tau_i^{\text{pre}}[\tau_i^{\text{suf},1}[\tau_i^{\text{suf},2}]^{\omega} \) are constructed. For example, robot \( R_1 \) changes its initial plan \( \tau_1 \) to \( \tilde{\tau}_1 = \tau_1^{\text{pre}}[\tau_1^{\text{suf},1}[\tau_1^{\text{suf},2}]^{\omega} \), where \( \tau_1^{\text{suf},1} = \tau_1^{\text{suf}} \) and

\[
\tau_1^{\text{suf},2} = [(1.75, 0.25)(2.25, 0.25)(0.25, 1.25)(0.75, 0.25)].
\]

Similarly, the suffix parts for robots \( R_2, R_3, \) and \( R_4 \) are

\[
\tau_2^{\text{suf},2} = [(0.75, 1.25)(2.25, 0.25)(2.25, 0.25)(0.25, 0.25)(1.75, 1.25)],
\]
\[
\tau_3^{\text{suf},2} = [(1.25, 0.75)(1.75, 1.75)(1.75, 1.75)(0.25, 0.75)(1.75, 1.25)],
\]
\[
\tau_4^{\text{suf},2} = [(1.75, 1.75)(1.25, 1.75)(1.75, 0.75)(0.25, 1.75)].
\]

Moreover, according to the suffix parts \( \tau_i^{\text{suf},2} \), the communication points selected by Algorithm 3 for teams \( T_1 = \{R_1, R_2\}, T_2 = \{R_2, R_3\}, \) and \( T_3 = \{R_3, R_4\} \) are (2.25, 0.25), (1.75, 1.25), and (1.75, 1.75), respectively. As mentioned in Section 3.7.1, the robots have a limited communication range and when two robots meet, they exchange messages with each other. We run the experiment for around 400s until all
robots and the user have all messages originally gathered by each robot. The final trajectories of robots $R_1$ and $R_2$ are shown in Figure 3.11.

The communication events among the robots and the user are shown in Figure 3.12. It can be seen that the robots are not connected for most of the time and communication only happens at intermittent time instants (13 communication events in total). Finally, the evolution of the messages stored at each robot is illustrated in Figure 3.13, which verifies that eventually all robots and the user have all the messages gathered by the robots. The full experiment video can be found in ExperimentVideo (2017).
Figure 3.12: Evolution of communication events among the robots, and among the robots and the user over time. The black line represents a communication event between two robots or a robot and the user.

Figure 3.13: The evolution of messages stored at each robot and the user. The messages initially gathered by robots $R_1, R_2, R_3, R_4$ are represented by $m_1, m_2, m_3, m_4$ in cyan, blue, green, magenta, respectively. The four memory slots for each robot and the user are used to store these messages. The messages are exchanged whenever communication happens. Eventually, all the initially generated messages have been transmitted to all robots and the user.
In this chapter, we consider robots that are tasked with collectively estimating a collection of hidden states. As in Chapter 3, we assume that the robots have limited communication capabilities and can only communicate when they are sufficiently close to each other. Therefore, intermittent communication protocols are preferred that allow the robots to temporarily disconnect from the network in order to collect information free of communication constraints. Our goal in this chapter is to design controllers that minimize the uncertainty of the hidden states and ensure that the communication network is connected over time infinitely often. Note that unlike Chapter 3, in this chapter, the teams of robots do not meet at *a priori* known locations in space to communicate, but instead they form connected sub-networks that are unknown and they are decided by the robots considering their objective to explore their environment and gather information.
4.1 Problem Formulation

Let \( x(t) \in \mathbb{R}^n \) denote a state variable that evolves according to the following nonlinear dynamics:

\[
x(t + 1) = f(x(t), u(t), w(t)),
\]

where \( u(t) \in \mathbb{R}^{d_u} \) and \( w(t) \in \mathbb{R}^{d_w} \) denote the control input and the process noise at discrete time \( t \). Without loss of generality, we assume that the process noise \( w(t) \) is normally distributed as \( w(t) \sim \mathcal{N}(0, Q(t)) \), where \( Q(t) \in \mathbb{S}_+^{d_w} \) and \( \mathbb{S}_+^{d_w} \subset \mathbb{R}^{d_w \times d_w} \) is the set of symmetric positive-definite matrices.

Consider also \( N \) mobile robots tasked with collaboratively estimating the state \( x(t) \) of the dynamic process in (4.1). Let \( \mathcal{W} \subset \mathbb{R}^d \) be the domain where the robots live, where \( 1 \leq d \leq 3 \), and let \( O \subset \mathcal{W} \) denote the set of obstacles. We assume that the robots collect measurements inside the obstacle-free subset \( \mathcal{W}_{\text{free}} = \mathcal{W} \setminus O \) of this domain according to the following sensing model:

\[
y(t, q) = h(x(t), q, v(t)),
\]

where \( y \in \mathbb{R}^m \) is the measurement vector at discrete time \( t \) taken at location \( q \in \mathcal{W}_{\text{free}} \) and \( v(t) \sim \mathcal{N}(0, R(t)) \) is the white measurement noise with covariance \( R(t) \in \mathbb{S}_+^{d_v} \). The dynamics of the state, control vectors, sensing model, and process and measurement noise covariances are known.

To define a communication network among the robots, we first partition the robot team into \( M \geq 1 \) robot subgroups, called also teams, denoted by \( T_i \), where \( i \in \{1, 2, \ldots, M\} \), and require that every robot belongs to two subgroups, so that the resulting team membership graph \( \mathcal{G} \), defined in Definition 2.1.1, in Chapter 2, is connected. We denote by \( r_{ij} \) a robot that belongs to teams \( T_i \) and \( T_j \) that is governed by the following nonlinear dynamics:

\[
p_{ij}(t + 1) = g(p_{ij}(t), u_{ij}(t)),
\]
where $p_{ij}(t) \in W_{\text{free}}$ stands for the position of robot $r_{ij}$ and $u_{ij}(t) \in \mathbb{R}^{d_r}$ stands for a control input. Without loss of generality, we assume that all robots have the same dynamics.

We assume that the robots have limited communication capabilities and, therefore, they can communicate their measurements only when they are sufficiently close to each other. Specifically, every robot is able to communicate with another robot if it lies within a communication range $R \ll \text{diam}(W)$, where $\text{diam}(W)$ is the diameter of the domain $W$. Without loss of generality we assume that the communication range is the same for all robots. Since the robots have limited communication capabilities, we assume that robots in every subgroup $\mathcal{T}_i$ communicate when all of them construct a connected sub-network. Hereafter, we denote by $\mathcal{G}_{\mathcal{T}_i} = \{\mathcal{V}_{\mathcal{T}_i}, \mathcal{E}_{\mathcal{T}_i}(t)\}$ the communication graph constructed by robots in team $\mathcal{T}_i$, where the set of nodes $\mathcal{V}_{\mathcal{T}_i}$ is indexed by the members of team $\mathcal{T}_i$ and the set of edges $\mathcal{E}_{\mathcal{T}_i}(t)$ collects edges that emerge between the robots in $\mathcal{T}_i$, when their distance is less than or equal to $R$. This way, a dynamic robot communication network is constructed as defined in Definition 2.1.2 in Chapter 2.

The problem that we address in this paper can be summarized as follows:

**Problem 4.1.1.** Given the dynamic process (4.1) and measurement model (4.2), and a network of $N \geq 1$ robots divided into $M \geq 1$ teams $\mathcal{T}_i$, $i \in \{1, \ldots, M\}$, so that $\mathcal{G}_\mathcal{T}$ is connected, determine paths $p_{ij}(t) \in W$ for all robots $r_{ij}$, so that (i) the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often, i.e., the robots in team $\mathcal{T}_i$ form a connected graph infinitely often, for all $i \in \{1, \ldots, M\}$, and (ii) all robots collectively minimize the estimation error and uncertainty of the state $x(t)$.

**Remark 4.1.2** (Teams). Note that in this chapter, for simplicity, we require that every robot belongs to two teams. However, more complex team membership graphs $\mathcal{G}_\mathcal{T}$ can be considered, as in Chapter 3, where robots can belong to any number of
To address Problem 4.1.1, we propose a distributed control framework that consists of two components that is described in section 4.2.1. The first one pertains to the distributed, discrete controller constructed in Chapter 2 that generates an infinite sequence of communication events for every robot team and guarantees that the communication network $G_c(t)$ is connected over time infinitely often. The generated sequence of communication events acts as an input to the second component of the proposed controller that determines motion plans for the robots that allow them to gather information, while ensuring that robots within every team form connected sub-networks to exchange the gathered information.

4.2 Informative Path Planning and Intermittent Communication Control

4.2.1 Informative Planning

Given the graph $G_T$, we construct communication schedules $\text{sched}_{ij}$ for all robots $r_{ij}$ as discussed in Chapter 2. Recall that the schedules $\text{sched}_{ij}$ determine abstract sequence of communication events that are not associated with any physical location in space. Similarly, the index $z_{ij}$, defined in Chapter 2, indicates the place of a communication event within the sequence $\text{sched}_{ij}$ but is not associated with physical time. Hereafter, we refer to the indices $z_{ij}$, defined in Chapter 2, as epochs. In this section, we embed the sequence of communication events $\text{sched}_{ij}$ over the epochs $z_{ij}$ into time $t$ and space $W$. For this we design robot trajectories that not only allow the robots to obtain measurements of the state that minimize estimation uncertainty, but also ensure that all robots in every team communicate with each other at epochs specified by the schedules $\text{sched}_{ij}$. Recall from Chapter 2, that the epochs $z_{ij}$ are updated asynchronously across the robots $r_{ij} \in T_i$. Also, recall that by construction of the schedules $\text{sched}_{ij}$, the communication event for team $T_i$ occurs at an epoch,
denoted by \( z_i \), that is common for all robots \( r_{ij} \in T_i \). Hereafter, for simplicity of notations, we drop all subscripts from these epochs, when it is clear from the context, and we simply refer to them as epochs \( z \).

Let \( z \) be an epoch associated with a communication event for team \( T_i \), i.e., \( \text{sched}_{ij}(z) = i \), for all \( r_{ij} \in T_i \). Moreover, let \( \mathbf{p}_{ij}^z : [t_{0,ij}, t_{f,ij}] \rightarrow \mathcal{W}_{\text{free}} \) denote the \( z \)-th segment of path \( \mathbf{p}_{ij}(t) \) of robot \( r_{ij} \in T_i \) starting at the discrete time \( t_{0,ij} \) and ending at \( t_{f,ij} \). Communication of team \( T_i \) at the discrete time step \( z \) takes place at the time instant \( t_{f,i} = \max_{r_{ij} \in T_i}\{t_{f,ij}\} \) when all agents arrive at their locations \( \mathbf{p}_{ij}(t_{f,ij}) \).

The starting location \( \mathbf{p}_{ij}(t_{0,ij}) \) of this path coincides with the location where the last communication event for robot \( r_{ij} \) took place within team \( T_j \); see also Figure 4.1. The discrete time and corresponding time instant when communication within team \( T_j \) occurs are denoted by \( \tilde{z} \) and \( t_{f,j} \), respectively, where \( z - \ell + 1 \leq \tilde{z} < z \) by periodicity of the communication schedules with period \( \ell \). Thus, robot \( r_{ij} \) starts executing the path segment \( \mathbf{p}_{ij}^z \) at the time instant \( t_{0,ij} = t_{f,j} \) and finishes its execution at \( t_{f,ij} \). At time \( t_{f,i} \), when the robots in team \( T_i \) form a connected communication graph, they fuse all collected and communicated measurements since their last communication event occurred at the discrete time \( t_{f,i}^{z-\ell} \) to update their beliefs about the evolution of the state \( \mathbf{x}(t) \). Next, these robots collectively design the path segments \( \mathbf{p}_{ij}^{z+\ell} \) that will result in a connected configuration for team \( T_i \) at discrete time \( z + \ell \), since by construction the communication schedule \( \text{sched}_{ij} \) is periodic, i.e., \( \text{sched}_{ij}(z) = \text{sched}_{ij}(z + \ell) = i \); see Figure 4.1 as well. In what follows, our goal is to design path segments \( \mathbf{p}_{ij}^{z+\ell} \) that minimize estimation uncertainty and satisfy at least three constraints:

- First, the paths \( \mathbf{p}_{ij}^{z+\ell} \) do not intersect with the obstacles and respect the dynamics (4.3).

- Second, the end locations \( \mathbf{p}_{ij}(t_{f,ij}^{z+\ell}) \) for all robots \( r_{ij} \in T_i \) correspond to a con-
Figure 4.1: Illustration of the proposed planning algorithm for robot $r_{14}$. The communication schedule $\text{sched}_{14}$ has the following form: $\text{sched}_{14} = [1, 4, X]^ω$, with period $\ell = 3$. The figure shows the path segments $p^k_{14}$ that robot $r_{12}$ follows, where the path $p^5_{14}$ was designed at discrete time $z = 5 - T = 2$. The starting location of the path $p^7_{14}$ that leads to a connected configuration within team $\mathcal{T}_i$ starts at the location $p^t_{14} = p_{14}$ at which the last communication event for robot $r_{14}$ occurred. Also, observe that since $\text{sched}_{14}(6) = X$, there is no path $p^6_{14}$.

\begin{itemize}
  \item Third, the end times $t^z_{f,ij}$ of the paths $p^z_{ij}$ are equal to $t^z_{f,i}$ for all robots $r_{ij} \in \mathcal{T}_i$, so that there are no robots waiting for the arrival of other robots.
\end{itemize}

Next, we formulate the optimal control problem that the robots $r_{ij} \in \mathcal{T}_i$ solve when they communicate at discrete time $z$ to design paths $p^z_{ij}$. Let $\mathcal{C}(W_{\text{free}})$ denote the set of continuous functions in the obstacle-free domain $W_{\text{free}}$. Furthermore, let $\mathcal{G}^{z+\ell}_{\mathcal{T}_i} = (\mathcal{V}_{\mathcal{T}_i}, \mathcal{E}^{z+\ell}_{\mathcal{T}_i})$ denote the communication graph formed by the robots in team $\mathcal{T}_i$ at discrete time $z + \ell$ where $\mathcal{V}_{\mathcal{T}_i} = \{r_{ij} \in \mathcal{T}_i\}$ and

$$\mathcal{E}^{z+\ell}_{\mathcal{T}_i} = \{(r_{ij}, r_{ie}) \mid \|p_{ij}(t^z_{f,ij}) - p_{ie}(t^z_{f,ie})\| \leq R\}.$$  

We define by $\mathcal{C} = \{\mathcal{G}(\mathcal{V}, \mathcal{E}) \mid \lambda_2(\mathcal{L}(\mathcal{G})) > 0\}$ the set of all connected graphs $\mathcal{G}$ with
vertices $\mathcal{V}$ and edges $\mathcal{E}$, i.e., the set of graphs $\mathcal{G}$ whose Laplacian matrix $L(\mathcal{G})$ has positive second smallest eigenvalue Godsil and Royle (2001). Then, the optimal control problem that robots $r_{ij} \in \mathcal{T}_i$ solve is defined as follows:

$$\min_{P_{T_i}} \sum_{t=t_{0,i}}^{t_{f,i}} \text{unc}(P_{T_i}(t))$$

s.t. $P_{ij}(t_{0,ij}^z) = P_{ij}(t_{f,ij}^z), \forall r_{ij} \in \mathcal{T}_i$

$$P_{ij}(t) \in C(W_{\text{free}}), \forall r_{ij} \in \mathcal{T}_i$$

$$P_{ij}(t + 1) = g(P_{ij}(t), u_{ij}(t)), \forall r_{ij} \in \mathcal{T}_i$$

$$\mathcal{G}_{T_i}^{z+\ell} \in \mathcal{C},$$

$$t_{f,i} \geq t_{0,ij}^z, \forall r_{ij} \in \mathcal{T}_i$$

$$t_{f,ij} = t_{f,i}, \forall r_{ij} \in \mathcal{T}_i, \quad (4.4)$$

In (4.4), $P_{T_i}$ stands for the augmented path of the robots that lives in the joint space $W|\mathcal{T}_i|$. Projection of this path on the workspace of robot $r_{ij}$ yields the path segment $p_{ij}^{z+\ell}$. $\text{unc}(P_{T_i}(t))$ denotes an uncertainty metric, such as the maximum eigenvalue or the trace of the posterior covariance matrix. Then, the objective function measures the cumulative uncertainty in the estimation of the state $x$ after fusing information from all robots $r_{ij} \in \mathcal{T}_i$ collected along the individual paths $p_{ij}$ from $t_{0,i} = \min\{t_{0,ij}^z\}$ up to time $t_{f,i}$. The first constraint enforces that the paths start from the location at which the previous communication event for teams $\mathcal{T}_j$ occurred. The second constraint in (4.4) ensures that the designed paths are continuous and lie in the free space. The third constraint ensures that all robots $r_{ij} \in \mathcal{T}_i$ can travel along their path segments $p_{ij}^{z+\ell}$ according to dynamics (4.3). The fourth constraint ensures that the communication graph $\mathcal{G}_{T_i}^{z+\ell}$ constructed at discrete time $z + \ell$, once all robots in $\mathcal{T}_i$ reach the end point of their respective paths $p_{ij}^{z+\ell}$, is connected. The fifth constraint requires the final time $t_{f,i}$ to be greater than the initial time $t_{0,ij}^z$, for all
robots $r_{ij} \in \mathcal{T}_i$. Finally, the last constraint requires that all robots $r_{ij} \in \mathcal{T}_i$ terminate the execution of their path segments $p_{ij}^{t+\ell}$ at time instants $t_{f,ij}$ that are the same and equal to $t_{f,i}$, i.e., there is no waiting time.

Regarding the optimal control problem (4.4), two points are relevant. First, without an additional constraint to promote exploration of the environment, it has a degenerate solution which is to meet as soon as possible. Appropriate selection of this constraint is problem dependent. Second, we can equivalently formulate the problem in terms of the robot control vectors $u_{ij}(t)$ but since our solution is sampling-based, we design the paths directly and deduce the control inputs accordingly.

### 4.2.2 Sampling-based Planning

Solving the path planning problem (4.4) in practice can be very challenging particularly because the function $\text{unc}(P_{\mathcal{T}_i}(t))$ is often non-smooth, the set $\mathcal{W}_\text{free}$ can be non-convex, and the problem explicitly involves the time variable $t$. Sampling-based algorithms are versatile tools that address the first two difficulties when the objective function is monotonic and bounded Karaman and Frazzoli (2011). Appearance of the meeting time and locations create additional challenges that further justify application of a sampling-based algorithm to solve (4.4). Specifically, here we employ the RRT* algorithm Karaman and Frazzoli (2011) and modify it to build a tree that explores the joint free domain $\mathcal{W}_{\text{free}}^{[\mathcal{T}_i]}$ and computes the path segments $p_{ij}^{t+\ell}$. The proposed sampling algorithm is summarized in Algorithm 5.

Application of Algorithm 5 requires definition of a cost function and a goal set $\mathcal{X}_g^i$. Referring to (4.4), the cost of the augmented path $P_{\mathcal{T}_i}$ that lives in $\mathcal{W}_{\text{free}}^{[\mathcal{T}_i]}$ is defined as $\text{Cost}(P_{\mathcal{T}_i}) = \sum_{t=t_{0,i}}^{t_{f,i}} \text{unc}(P_{\mathcal{T}_i}(t))$. This cost function is additive and thus monotone since estimation uncertainty is always positive, i.e., $\text{Cost}(P_{\mathcal{T}_i}^1) \leq \text{Cost}(P_{\mathcal{T}_i}^1|P_{\mathcal{T}_i}^2) = \text{Cost}(P_{\mathcal{T}_i}^1) + \text{Cost}(P_{\mathcal{T}_i}^2)$, where $|$ stands for the concatenation of the paths $P_{\mathcal{T}_i}^1$ and $P_{\mathcal{T}_i}^2$. 

86
Algorithm 5: Sampling-based Informative Path Planning

1. Set $V_s = \{v_0\}$, $E_s = \emptyset$, and $X_g' = \emptyset$;
2. for $s = 1, \ldots, n_{\text{sample}}$ do
   3. Sample $W_{\text{free}}$ to acquire $v_{\text{rand}}$;
   4. Find the nearest node $v_{\text{nearest}} \in V_s$ to $v_{\text{rand}}$;
   5. Steer from $v_{\text{nearest}}$ toward $v_{\text{rand}}$ to select $v_{\text{new}}$;
   6. if $\text{CollisionFree}(v_{\text{nearest}}, v_{\text{new}})$ then
      7. Update the set of vertices $V_s = V_s \cup \{v_{\text{new}}\}$;
      8. Build the set $V_{\text{near}} = \{v \in V_s \mid \|v - v_{\text{new}}\| < r\}$;
      9. Extend the tree towards $v_{\text{new}}$ (Algorithm 6);
     10. Rewire the tree (Algorithm 7);
   11. Find $v_{\text{end}} \in X_g^i$ with smallest uncertainty;
   12. Return the path $P_{T_i}^{z+\ell} = (v_0, \ldots, v_{\text{end}})$;
   13. Project $P_{T_i}^{z+\ell}$ onto the workspace of $r_{ij}$ to get $p_{ij}^{z+\ell}$;

The goal set in our sampling-based algorithm captures the constraints of the optimal control problem (4.4). Specifically, we define the goal set for team $T_i$ as follows:

$$X_g^i = \{v \in W_{\text{free}}^{T_i} \mid (i) \lambda_2(L(G_{T_i}^{z+\ell}(v))) > 0, \quad \text{(ii) } \min_{r_{ij} \in T_i} t_{ij}(v) = \max_{r_{ij} \in T_i} t_{ij}(v)\}.$$  \hfill (4.5)

In words, the goal set $X_g^i$ collects all points $v \in W_{\text{free}}^{T_i}$ that satisfy two conditions. First, the configuration $v$ should correspond to a connected communication graph at discrete time $z + \ell$ capturing the fourth constraint in (4.4). Second, the time instants at which robots $r_{ij}$ arrive at the projection of $v \in W_{\text{free}}^{T_i}$ on their own workspace, denoted by $t_{ij}(v)$, are the same for all robots $r_{ij} \in T_i$, i.e., $\min_{r_{ij} \in T_i} t_{ij}(v) = \max_{r_{ij} \in T_i} t_{ij}(v)$ capturing the last constraint in (4.4). Depending on the problem, additional conditions can be added to this set. Particularly, a clause must be added to promote exploration of the environment and prevent degenerate solutions, cf. Section 4.2.5.

To enforce the first constraint of (4.4), we select the root $v_0 \in W_{\text{free}}^{T_i}$ of the tree $G_s = \{V_s, E_s\}$ to match the positions $p_{ij}^{z+\ell}(t_{0,ij}^{z+\ell})$ of the robots $r_{ij} \in T_i$ in the joint workspace.
Figure 4.2: The schematic representation of a sampling-based solution for the planning problem (4.4). Robots are illustrated by red and green disks residing in 1-D workspace resulting in a 2-D joint space $W_{|T_i|}$ shown above. The gray square and circles in this space denote the root of the tree and the samples taken by algorithm. The goal set $X_g^i \subset W_{|T_i|}$ is a subset of the area between the red dashed lines where the two robots are within communication range $R$. The time constraint (ii) in (4.5) depends on the initial times of the robots and determines the size of this subset. Red circle stands for a sample that lies within the goal set and is connected to the root through the red solid lines.

space $W_{\text{free}}$. In other words, the tree generated by Algorithm 5 is rooted at the end point of the previous paths $p_{ij}^\hat{z}$, where $z < \hat{z} \leq z + \ell - 1$. Also, the tree is initialized as $V_s = \{v_0\}$, $E_s = \emptyset$ [line 1, Alg. 5]. The constraint enforcing the dynamics of the robots is explained in the steering function below and the rest of the constraints are enforced by construction of Algorithm 5. Figure 4.2 shows a schematic of the algorithm.

Sampling function

At every iteration of Algorithm 5, a new sample $v_{\text{rand}}$ from the joint space $W_{\text{free}}^{|T_i|}$ is taken [line 3, Alg. 5]. Also, we assume that the first $B$ samples are drawn from a distribution $f_1 : W_{\text{free}}^{|T_i|} \rightarrow [0, 1]$, e.g., a uniform distribution that forces the
Algorithm 6: Extend

1. Set \( \text{unc}_{\text{min}} = \text{Cost}(v_{\text{nearest}}); \)
2. for \( v_{\text{near}} \in V_{\text{near}} \) Compute \( \text{Cost}(v_{\text{new}}) \) if parent of \( v_{\text{new}} \) is \( v_{\text{near}} \) (Algorithm 8);
3. if \( \text{CollisionFree}(v_{\text{near}}, v_{\text{new}}) \land \text{Cost}(v_{\text{new}}) < \text{unc}_{\text{min}} \) then
4. \( \text{Set } v_{\text{min}} = v_{\text{near}}, \text{unc}_{\text{min}} = \text{Cost}(v_{\text{new}}); \)
5. Update the set of edges \( E_s = E_s \cup \{(v_{\text{min}}, v_{\text{new}}); \}
6. if \( v_{\text{new}} \in X^* \) then
7. \( \text{Update } X^*_g = X^*_g \cup \{(\text{unc}_{\text{min}}, v_{\text{new}}); \}

robots to explore \( W_{\text{free}} \) and after that samples are drawn from a second distribution \( f_2 : W_{\text{free}} \rightarrow [0, 1] \) that promotes the construction of a connected communication graph \( G_{T_i}; \) cf. Section 4.2.5 for details.

Steer

The next step in Algorithm 5 is to examine if the tree can be extended towards the new sample \( v_{\text{rand}}. \) First, among all the nodes in the set \( V_s \) we find the nearest node to \( v_{\text{rand}}, \) which is denoted by \( v_{\text{nearest}} \) \([\text{line } 4, \text{ Alg. } 5]\). Then, we employ a steering function that given the robot dynamics (4.3) returns a point \( v_{\text{new}} \) \([\text{line } 3, \text{ Alg. } 5]\), that (i) minimizes the distance \( \|v_{\text{new}} - v_{\text{rand}}\|, \) (ii) satisfies \( \|v_{\text{nearest}} - v_{\text{new}}\| \leq \epsilon, \) for some \( \epsilon > 0, \) and (iii) ensures that all robots starting from \( v_{\text{nearest}} \) arrive at \( v_{\text{new}} \) at the same time \( t, \) if possible. \([\text{line } 5, \text{ Alg. } 5]\). Note that the steering function captures the dynamics of the robots Karaman and Frazzoli (2010), i.e., the third constraint in (4.4). See Section 4.4 for details about the employed steering function.

Extend

The new point \( v_{\text{new}} \) will be added to the set \( V_s \) if given a control input \( u_{ij} \) and robot dynamics (4.1), the trajectory starting from \( v_{\text{nearest}} \) and ending at \( v_{\text{new}} \) lies in \( W_{\text{free}}. \) This is checked using the function \( \text{CollisionFree}(v_{\text{nearest}}, v_{\text{new}}) \) \([\text{line } 6, \text{ Alg. } 5]\). If this line does not intersect any obstacles, we add \( v_{\text{new}} \) to the set \( V_s \) \([\text{line } 7, \text{ Alg. } 5]\) and we construct the set \( V_{\text{near}} = \{v \in V_s | \|v - v_{\text{new}}\| < r\}, \) for some \( r > 0 \)
Algorithm 7: Rewire

1. for $v_{\text{near}} \in V_{\text{near}}$ do
2.     if $\text{CollisionFree}(v_{\text{new}}, v_{\text{near}}) \land \min_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}}) = \max_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}})$ then
3.         Compute cost $\hat{\text{Cost}}(v_{\text{near}})$ (Algorithm 8);
4.         if $\hat{\text{Cost}}(v_{\text{near}}) \leq \text{Cost}(v_{\text{near}}) \land \min_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}}) = \max_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}})$ then
5.             Update the set of edges $\mathcal{E}_s = \mathcal{E}_s \setminus \{(v_{\text{parent}}, v_{\text{near}})\} \cup \{(v_{\text{new}}, v_{\text{near}})\}$;
6.         Update the goal set $\mathcal{X}_g$;

selected as in Karaman and Frazzoli (2011) that collects all nodes that are within distance $r$ from the point $v_{\text{new}}$ [line 8, Alg. 5]. Among the nodes that belong to $V_{\text{near}}$, we pick the parent node of $v_{\text{new}}$, denoted by $v_{\text{min}}$, that incurs the minimum possible cost for the node $v_{\text{new}}$ denoted by $\text{Cost}(v_{\text{new}})$ [line 9, Alg. 5]. Computation of $\text{Cost}(v_{\text{new}})$ is described in Section 4.2.3. The node $v_{\text{min}}$ is computed by Algorithm 6. Once, the parent $v_{\text{min}}$ of $v_{\text{new}}$ has been selected, we update the set of edges as $\mathcal{E}_s = \mathcal{E}_s \cup \{(v_{\text{min}}, v_{\text{new}})\}$ [line 5, Alg. 6]. If $v_{\text{new}}$ is in goal set, we update the set $\mathcal{X}_g$ as $\mathcal{X}_g = \mathcal{X}_g \cup \{v_{\text{new}}\}$ [line 7, Alg. 6].

Rewire

After extending the tree $G_s$ towards $v_{\text{new}}$ the rewiring process follows that checks if it is possible to further reduce the cost of the nodes of the tree by rewiring them through $v_{\text{new}}$ [line Alg. 5]. The rewiring operation is described in Algorithm 7. In particular, for every node $v_{\text{near}} \in V_{\text{near}}$ that satisfies (i) $\min_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}}) = \max_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}})$, and (ii) can be connected through an obstacle-free path to $v_{\text{new}}$, we compute $\hat{\text{Cost}}(v_{\text{near}})$ assuming that it was connected through $v_{\text{new}}$ [lines 1-3, Alg. 7]. Then we rewire $v_{\text{near}}$ if the cost $\hat{\text{Cost}}(v_{\text{near}})$ using $v_{\text{new}}$ as parent is less than the current cost $\text{Cost}(v_{\text{near}})$ of $v_{\text{near}}$ and if there are control inputs $u_{ij}$ that can still drive all robots to $v_{\text{near}}$ at the same time, which may be different than before [line 4, Alg. 7]. If both conditions are satisfied then we update the set of edges $\mathcal{E}_s$ by deleting
the previous edge [\(v_{\text{parent}}, v_{\text{near}}\)], where \(v_{\text{parent}}\) stands for the previous parent of \(v_{\text{near}}\), and adding the new edge [\(v_{\text{new}}, v_{\text{near}}\)] [line 5, Alg. 7]. Notice that the requirement that a node \(v_{\text{near}} \in \mathcal{V}_{\text{near}}\) should satisfy \(\min_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}}) = \max_{r_{ij} \in \mathcal{T}_i} t_{ij}(v_{\text{near}})\) to get rewired does not exist in the RRT* algorithm 5.

If a node \(v_{\text{near}}\) is rewired then we update the cost of all successor nodes of \(v_{\text{near}}\) in \(\mathcal{V}_s\). Also, notice that after rewiring a node \(v_{\text{near}}\), the time instant \(t_{ij}(v_{\text{near}})\) associated with the arrival of robots \(r_{ij}\) at \(v_{\text{near}}\) may change. As a result, \(\text{cost}(v_{\text{near}})\) may change. The same holds for all successor nodes of \(v_{\text{near}}\). Thus, after rewiring, the goal set needs to be updated, since \(v_{\text{near}}\) and its successor nodes may enter or leave the goal set [line 6, Algorithm 7].

After taking \(n_{\text{sample}}\) samples, Algorithm 5 terminates and returns the node \(v_{\text{end}} \in \mathcal{X}_{\text{g}}\) with the smallest uncertainty and finds the path \(P_{\mathcal{T}_i}^{z+\ell} = (v_0, \ldots, v_{\text{end}})\) that connects \(v_{\text{end}}\) to the root \(v_0\) of the tree [lines 11-12, Alg. 5]. The individual path segments \(p_{ij}^{z+\ell}\) of the agents are obtained by projecting the augmented path \(P_{\mathcal{T}_i}^{z+\ell}\) into the workspace of each agent. Note that the paths start at different initial times \(t_{0,ij}^{z+\ell}\) but end at the same final time \(t_{f,i}^{z+\ell}\) since \(v_{\text{end}}\) belongs to the goal set.

### 4.2.3 Computation of Cost\((v_{\text{new}})\)

In this section, we discuss the computation of the cost of node \(v_{\text{new}}\) given a candidate parent node \(v_{\text{near}}\); see Algorithm 8. Notice that \(\text{cost}(v_{\text{new}})\) is equivalent to the cost of the path that connects the node \(v_{\text{new}}\) to the root of the tree \(v_0\). In Algorithm 8, to compute \(\text{cost}(v_{\text{new}})\), we first construct a set \(\mathcal{Y} = \{(t_1, q_1), \ldots, (t_2, q_2), \ldots, (t_{\ell}, q_{\ell})\}\) that collects the time instants \(t_{\xi}\) and locations \(q_{\xi} \in \mathcal{W}_{\text{free}}\) at which the robots within team \(\mathcal{T}_i\) would take measurements, following their projected path from \(v_{\text{near}}\) to \(v_{\text{new}}\). We assume that the robots take measurements every \(\Delta t > 0\) time steps and order
the set \( Y \) in time such that \( t_1 \leq t_\xi \leq t_\Xi \), where

\[
t_1 = \min_{r_{ij} \in T_i} t_{ij}(v_{\text{near}}) \quad \text{and} \quad t_\Xi = \min_{r_{ij} \in T_i} t_{ij}(v_{\text{new}}).
\]

The cost of node \( v_{\text{new}} \) is initialized as \( \text{Cost}(v_{\text{new}}) = \text{Cost}(v_{\text{near}}) \) [line 1, Alg 8]. Then, using the Extended Kalman Filter (EKF) from \( t_{\text{near}} \) until \( t_{\text{new}} \) we update the cost of \( v_{\text{new}} \) by fusing sequentially all predicted measurements that would be taken at locations and time instants collected in \( Y \). Specifically, during the time intervals \( (t_{\xi-1}, t_\xi) \), we execute the prediction equation of the EKF to compute the predicted covariance matrix \( C(t) \). Given the predicted covariance matrix \( C(t) \), we compute \( \text{unc}(P_{T_i}(t)) \), where \( P_{T_i}(t) \in \mathcal{W}^{||T_i||}_{\text{free}} \) stands for the positions of robots in team \( T_i \) at time \( t \). Then, we update the cost of \( v_{\text{new}} \) as \( \text{Cost}(v_{\text{new}}) = \text{Cost}(v_{\text{new}}) + \text{unc}(P_{T_i}(t)) \), for all \( t \in (t_{\xi-1}, t_\xi) \) [lines 4-5, Alg. 8]. At time step \( t_\xi \), using the predicted measurement that would be taken at this time instant, we compute the estimated covariance matrix \( C(t_\xi) \) [lines 6-7, Alg. 8]. Given \( C(t_\xi) \), we compute \( \text{unc}(P_{T_i}(t_\xi)) \) and update the cost of node \( v_{\text{new}} \) as \( \text{Cost}(v_{\text{new}}) = \text{Cost}(v_{\text{new}}) + \text{unc}(P_{T_i}(t_\xi)) \) [line 7, Alg. 8]. This procedure is repeated for the next time interval \( (t_\xi, t_{\xi+1}) \).

It is possible that multiple concurrent measurements would be taken by robots. In such case these measurements need to be fused together in Algorithm 8. Moreover, during planning the actual measurements \( y(t_\xi) \) in (4.7a) are unavailable and the predicted state will directly be used in (4.8b).

**Remark 4.2.1 (Estimation Filter).** In Algorithm 8 we have used EKF to fuse the measurements collected by the agents but any other estimation filter could be used. Particularly, the Information form of the Kalman filter is attractive for distributed fusion since the information from different measurements is additive Durrant-Whyte et al. (2001). For the case of out-of-order data, the Information form requires storage of an \( n \times 1 \) information vector corresponding to each measurement Nettleton and
Algorithm 8: Extended Kalman Filter

**Input:** Dynamics (4.1), observation model (4.2), and set \( \mathcal{Y} \); Process noise covariance \( Q(t) \) and measurement noise covariance \( R(t) \); Initial estimates \( \hat{x}_{\text{near}} \) and \( C(t_{\text{near}}) \)

1. Initialize \( \text{Cost}(v_{\text{new}}) = \text{Cost}(v_{\text{near}}) \);
2. Set \( \xi = 2 \);
3. while \( \xi \leq \Xi \) do
4.   for \( t = t_{\xi-1} : t_{\xi} \) do
5.     Compute the EKF prediction
6.     \[
\begin{align*}
\dot{x}(t) &= f(\dot{x}(t-1), u(t-1)) \\
C(t) &= \nabla f|_{\dot{x}(t-1)} C(t-1) \nabla f|_{\dot{x}(t-1)}^T + Q(t) \\
\text{Cost}(v_{\text{new}}) &= \text{Cost}(v_{\text{new}}) + \text{unc}(P_{\tau(t)})
\end{align*}
\]
(4.6a)
(4.6b)
(4.6c)
7.     Compute Innovations
8.     \[
\begin{align*}
\dot{y}(t_{\xi}) &= y(t_{\xi}) - h(q_{\xi}, \dot{x}(t_{\xi})) \\
S(t_{\xi}) &= \nabla h|_{\dot{x}(t_{\xi})} C(t_{\xi}) \nabla h|_{\dot{x}(t_{\xi})} + R(t_{\xi})
\end{align*}
\]
(4.7a)
(4.7b)
Kalman gain: \( K(t_{\xi}) = C(t_{\xi}) \nabla h|_{\dot{x}(t_{\xi})} S(t_{\xi})^{-1} \);
9.     Compute the EKF estimates
10. \[
\begin{align*}
\dot{x}(t_{\xi}) &= \dot{x}(t_{\xi}) + K(t_{\xi})\dot{y}(t_{\xi}) \\
C(t_{\xi}) &= (I_n - K(t_{\xi})) \nabla h|_{\dot{x}(t_{\xi})} C(t_{\xi})
\end{align*}
\]
(4.8a)
(4.8b)
Update: \( \text{Cost}(v_{\text{new}}) = \text{Cost}(v_{\text{new}}) + \text{unc}(P_{\tau(t)}) \);

**Durrant-Whyte (2001) and the Kalman form requires storage of the time, location, and value of each measurement, i.e., a \((1 + d + m) \times 1\) vector. Thus, selection of the appropriate filter is problem-dependent.**

4.2.4 Completeness and Optimality

Our proposed sampling-based algorithm is probabilistically complete, i.e., if there exist paths \( p_{ij}^{z+\ell} \) that end up in the goal set (4.5), then Algorithm 5 will find such paths with probability 1, as \( n_{\text{sample}} \to \infty \). To show this result, recall that RRT* is probabilistically complete as a result of the functions ‘Steer’ and ‘Extend’ Karaman
and Frazzoli (2011). The only requirement in the ‘Steer’ function is that the node $v_{\text{new}}$ should be closer to $v_{\text{rand}}$ than $v_{\text{nearest}}$ is, which is trivially satisfied by Algorithm 5.

Nevertheless, Algorithm 5 is not asymptotically optimal since we rewire a node that belongs to a subset of $V_{\text{near}}$ only if after rewiring its cost decreases and all robots can still arrive at this node simultaneously. On the other hand, RRT* examines all nodes that belong to $V_{\text{near}}$ ignoring the time constraint discussed before. To recover the asymptotic optimality of the RRT* we can relax the timing constraint (ii) in the goal set (4.5) and perform the rewiring step as in the RRT* algorithm. As a result, the robots will not arrive at their meeting location simultaneously. Requiring all robots to form a connected graph at the same time has two benefits. First, it prevents the robots from waiting in the vicinity of a meeting location which would be suboptimal for estimation of a dynamic process. Second aligning the time stamp of measurements of the agents during sampling, speeds up the fusion process and alleviates the need for storing measurement information every time a new node is added to the tree.

4.2.5 Integrated Path Planning and Intermittent Communication Control

The integrated algorithm is described in Algorithm 9. Every robot $r_{ij}$ follows the path $p_{z_{ij}}$ while taking measurements [line 2, Alg. 9].\footnote{Notice that before the first communication event, robots in team $\mathcal{T}_i$ have not designed yet any paths $p_{ij}^\epsilon$. The paths that the robots of team $\mathcal{T}_i$ should follow to participate in their first communication event are \textit{a priori} planned and are denoted by $p_{ij}^1$.} When all robots in team $\mathcal{T}_i$ reach the final waypoint of their respective path segments $p_{ij}^\epsilon$, they form a connected communication sub-network. Note that it is possible that due to uncertainty or exogenous disturbances, the robots in team $\mathcal{T}_i$ may not reach the final waypoint of their path segment $p_{ij}^\epsilon$, simultaneously. Therefore, when the robots in team $\mathcal{T}_i$ arrive at the final waypoint of their respective paths $p_{ij}^\epsilon$, they perform the following control
Algorithm 9: Proposed Control Framework for robot $r_{ij}$

Input: Initial paths $p_{ij}^1$, $p_{ij}^2$, and the plan $\text{sched}_{ij}$

1. for $z = 2 : \infty$ do
2. Move along the path $p_{ij}^z$ and take measurements at each waypoint;
3. When the final waypoint of $p_{ij}^z$ is reached, wait until all robots in the respective team have arrived;
4. Exchange information with all other robots $r_{im} \in T_i$;
5. Update the state estimate and the covariance matrix up to the current time $t$;
6. Compute the paths $p_{ij}^{z+\ell}$ to be followed to reach the next communication event (Algorithm 5);

policy, as in Chapter 3 [line 3, Alg. 9]

Definition 4.2.2 (Communication Control Policy). Every robot $r_{ij} \in T_i$ that arrives at the final waypoint of the path $p_{ij}^z$, waits there indefinitely, or until all other robots in the team arrive.

When all robots in $T_i$ arrive at the final waypoint of their respective paths $p_{ij}^z$, they form a connected communication sub-network and they exchange the information they have collected since their last communication event [line 4, Alg. 9]. This information can be due to new measurements or communication with other teams $T_j$. Given the new set of measurements, the state estimate and the respective covariance are updated using the Extended Kalman Filter [line 5, Alg. 9] given in Algorithm 8. Finally, the robots in team $T_i$ compute the paths $p_{ij}^{z+\ell}$ that will allow them to reconnect at epoch $z + \ell$ [line 6, Alg. 9].

4.2.6 Correctness

Since the robots communicate intermittently, information is propagated across the network with a delay. In the following proposition, we show that this propagation delay depends on the structure of the graph $G_T$ and on the period $\ell$ of the schedules $\text{sched}_{ij}$.
Proposition 4.2.3 (Delays). The worst case delay, in terms of the epochs $z_{ij}$, with which information generated at robot $r_{ij}$ will propagate to every other robot in the network is $D_{Gr_T} = (\ell - 1)L_{Gr_T}$, where $L_{Gr_T}$ is the longest shortest path in $G_T$ and $\ell$ is the period of the schedules $\text{sched}_{ij}$ defined in Definition 2.2.2, in Chapter 2. Specifically, $L_{Gr_T}$ is defined as $L_{Gr_T} = \max_{i,e \in \{1, \ldots, M\}} |\ell_{ie}|$, where $\ell_{ie}$ denotes the shortest path in $G_T$ that connects the teams $T_i$ and $T_e$, and $|\ell_{ie}|$ is the number of nodes in $\ell_{ie}$.

Proof. Assume that at epoch $z_{ij}$ robot $r_{ij}$ collects a packet of information $I$. Without loss of generality, assume that the next communication event for robot $r_{ij}$ is within team $T_i$. Due to the periodic structure of $\text{sched}_{ij}$ with period $\ell$, the next communication event for team $T_i$ will occur within the next $\ell - 1$ epochs. Thus, the packet of information collected by robot $r_{ij}$ at epoch $z_{ij}$ will be transmitted to all robots in team $T_i$ in $K_1 \leq \ell - 1$ epochs. Next consider the number of epochs required for this packet of information to be transmitted from team $T_i$ to any other team $T_e$. Since the graph $G_T$ is connected by assumption, there is at least one path that connects team $T_i$ to team $T_e$ and, therefore, information can be propagated from $T_i$ to $T_e$. Let $\ell_{ie}$ denote the shortest path between teams $T_i$ and $T_e$. Then, $I$ will be transmitted from $T_i$ to $T_e$ through the path $\ell_{ie}$, within $(|\ell_{ie}| - 1)(\ell - 1)$ epochs, where $|\ell_{ie}|$ is the length of the shortest path. Therefore, we get that $K_{ie} \leq (|\ell_{ie}| - 1)(\ell - 1)$.

Finally, given an arbitrary team $T_i$, we get $K_{ie} \leq (L_{gr_T} - 1)(\ell - 1)$, where $L_{gr_T}$ stands for the longest shortest path in $G_T$, i.e., $L_{gr_T} = \max_{i,e \in \{1, \ldots, M\}} |\ell_{ie}|$ by definition. Therefore, the packet of information collected by robot $r_{ij}$ at epoch $z_{ij}$ will be transmitted to any other team $T_e$ and, consequently, to all robots by the epoch $z^*_{ij}(z_{ij}) = z_{ij} + K_1 + K_{ie} \leq z_{ij} + \ell - 1 + (L_{gr_T} - 1)(\ell - 1) = z_{ij} + L_{gr_T}(\ell - 1)$ completing the proof.

Proposition 4.2.3 implies that any measurement collected at epoch $k_{ij}$ by a robot $r_{ij}$ will be propagated to all teams by epoch $k^*_{ij}(k_{ij}) = k_{ij} + D_{gr_T}$. This result quantifies
the amount of information that the agents need to store and implies that given an
epoch $k_{ij}$, all robots have fused the same set of measurements up to epoch $k_{ij} - D_{G_T}$. Consequently, information collected before that epoch can be discarded by robot $r_{ij}$. This observation is verified in our simulation studies.

Next, we show that the communication network $G_c(t)$ is connected over time infinitely often when the robots navigate the workspace as per Algorithm 9. To show this result, we first need to show that the network is deadlock-free, i.e., when the robots execute the path segments $p_{ij}^z$ according to Algorithm 9. The proof of this result is identical to the proof of Proposition 3.5.4 in Chapter 3. Using this result, we can show that the communication network $G_c(t)$ is connected over time infinitely often. This result is stated in Proposition 4.2.5.

**Proposition 4.2.4** (Deadlock-free). The mobile robot network is deadlock-free when the paths $p_{ij}^z$ are executed according to Algorithm 9.

**Proposition 4.2.5** (Intermittent Connectivity). The communication graph $G_c(t)$ is connected over time infinitely often as long as all robots $r_{ij}$ design and execute paths $p_{ij}^z$ as per Algorithm 9.

**Proof.** To show that $G_c(t)$ is connected over time infinitely often, it suffices to show that all teams $T_i$ communicate infinitely often. This is because every robot $r_{ij}$ belongs to two teams $T_i(t)$ and $T_j$, and the team membership graph $G_T$ is connected. To show this result, it suffices to show that the time interval between two communication events for all teams $T_i$ is finite. To show this, recall first that once the robots $r_{ij} \in T_i$ communicate, they coordinate to design their next connected configuration. Such a connected configuration will be found with probability 1, since the proposed algorithm is probabilistically complete, i.e., Second, recall that due to Proposition 4.2.4, it holds that the waiting times of robots at their respective positions in the connected configurations are bounded and, therefore, the network is deadlock-free.
Therefore, we conclude that robots in team $\mathcal{T}_i$ will eventually communicate again, for all $m \in \mathcal{M}$ completing the proof.

4.3 Simulation Studies

In this section, we demonstrate the performance of the proposed algorithm for a target tracking problem in a non-convex environment. The targets move in $\mathcal{W}_{\text{free}}$ and are modeled with a linear time-invariant dynamics

$$x_a(t + 1) = A_a x_a(t) + B_a u_a(t) + w_a(t),$$

where $a$ denotes the target index, $x_a(t) \in \mathbb{R}^3$, and $w_a(t) \sim (0, Q_a(t))$. For simplicity, we assume that the targets follow linear or circular paths corrupted by process noise. We also consider first-order dynamics

$$p_{ij}(t + 1) = p_{ij}(t) + u_{ij}(t),$$

for the robots, where $p_{ij}(t) \in \mathbb{R}^2$, $u_{ij} \in \mathbb{R}^2$, and $\|u_{ij}\| \leq 0.1$m. The environment where the targets and robots live is a square $10 \times 10$m$^2$ indoor environment $\mathcal{W}$, shown in Figure 4.7, and their communication range is $R = 0.2$m. Observe that the communication radius is quite small compared to the size of the workspace. The robots need to move in $\mathcal{W}_{\text{free}}$ to estimate the positions $x_a(t)$ of all targets.

We equip the robots with omnidirectional, range-only, line-of-sight sensors with limited range of 5m. Every robot can take noisy measurements $y_{ij}$ of its distance from all targets that lie within its sight and range. Specifically, the measurement associated with target $a$ is given by

$$y_{ij,a} = \ell_{ij,a}(t) + v(t) \quad \text{if} \quad \ell_{ij,a}(t) \leq 5,$$

where $\ell_{ij,a}(t) = \|p_{ij}(t) - x_a(t)\|$, $v(t) \sim \mathcal{N}(0, \sigma^2(t))$, and

$$\sigma(t) = \begin{cases} 0.01, & \text{if } \ell_{ij,a}(t) \leq 1, \\ 0.045 \ell_{ij,a}(t) - 0.035, & \text{if } 1 < \ell_{ij,a}(t) \leq 3, \\ 0.1, & \text{if } 3 < \ell_{ij,a}(t) \leq 5. \end{cases}$$
The standard deviation $\sigma(t)$ is initially very small, i.e., 0.01m, when the robot and target are less than 1m apart and then linearly increases until it becomes flat at 0.1m between 3m to 5m. This model captures the fact that the range readings become less accurate as the distance increases and is designed to motivate the robots to approach the targets.

In the following simulations, we consider $N = 8$ robots and 8 targets. The initial positions of the targets are stacked in the vector

$$x_0 = [1, 2, 1.5, 1, 1, 1.3, 8.5, 1.5, 0.8, 9.2, 9, 1, 7, 8, 1.1, 5.9, 9, 1.2, 3, 9, 1.3, 1.8, 8.2, 0.9]^T.$$ 

The Extended Kalman Filter (EKF) is initialized as

$$\hat{x}_0 = [1, 1.1, 1, 1, 1.9, 1, 1.9, 0.5, 1, 7, 9.5, 1, 4, 9, 1, 2.5, 7, 1, 1, 9, 1]^T,$$

and the respective covariance matrix $C(0)$ is diagonal with entries equal to 0.25.

We select the cost function $\text{unc}(P_{\mathcal{T}_i}(t))$ as $\text{unc}(P_{\mathcal{T}_i}(t)) = \lambda_n(C_{\mathcal{T}_i}(t))$, where $\lambda_n(\cdot)$ denotes maximum eigenvalue and $C_{\mathcal{T}_i}$ is the covariance matrix computed using only local information from team $\mathcal{T}_i$. In this way, we minimize the worst-case uncertainty of localizing the targets. The goal set is constructed as in (4.5) where we add a third constraint to prevent degenerate solutions as discussed in 4.2.2. Specifically, we require each team $\mathcal{T}_i$ to design paths so that the uncertainty of the most uncertain target, determined according to local information of team $\mathcal{T}_i$, drops below a threshold $\delta$. In other words, we require $\lambda_n(C_{\mathcal{T}_i}^{a_i}(t_{f,i})) \leq \delta$, where $a_i$ is the most uncertain target from the view point of team $\mathcal{T}_i$ and $C_{\mathcal{T}_i}^{a_i}(t)$ is the $a_i$-th diagonal block in the covariance matrix $C_{\mathcal{T}_i}$. In what follows, we select $\delta = 0.12^2 m^2$, i.e., we want the team $\mathcal{T}_i$ to localize the most uncertain target with uncertainty not worse than 0.12m. If the team $\mathcal{T}_i$ does not find a feasible path that satisfies this constraint, it checks for the next most uncertain targets, respectively.

Finally, the sampling function defined in Section 4.2.2, is constructed as follows.
First, we draw a pre-defined number of samples from a uniform distribution $f_1$ defined over $W_{\text{free}}^{T_i}$ that forces the team $T_i$ to explore the domain. Then, we switch to a distribution $f_2$ that promotes the robots to get closer and form a connected configuration. This distribution is constructed in a hierarchical way. Specifically, we first draw a potential meeting location $q \in W_{\text{free}}$ from a uniform distribution. Then, we replicate this point to construct the vector $\mu = [q^T, \ldots, q^T]^T \in W_{\text{free}}^{T_i}$ and draw the desired samples from the following normal distribution $f_2(v) = \mathcal{N}(\mu, (2R)^2 I)$.

In what follows, in Section 4.3.1, we examine the performance of our algorithm for two different configurations of the graph $G_T$. Then, in Section 4.3.2, we compare our proposed algorithm to an algorithm that preserves end-to-end network connectivity for all time and a heuristic approach that allows disconnection of the network similar to the approach proposed here, but selects meeting locations randomly. The first comparison establishes the significant advantage of the framework presented here against the extensive literature that enforces all-time connectivity constraints. The second comparison is meant to justify the use of sampling-based Algorithm 5 to solve the planning problem (4.4) instead of a simple heuristic.

4.3.1 Comparative Results for Different Graphs $G_T$

First, we assume that the network of $N = 8$ robots is divided into $M = 8$ teams defined as $T_1 = \{r_{12}, r_{18}\}$, $T_2 = \{r_{12}, r_{23}\}$, $T_3 = \{r_{23}, r_{34}\}$, $T_4 = \{r_{14}, r_{45}\}$, $T_5 = \{r_{56}, r_{45}\}$, $T_6 = \{r_{56}, r_{67}\}$, $T_7 = \{r_{67}, r_{78}\}$, $T_8 = \{r_{18}, r_{78}\}$. The resulting graph, denoted by $G^1_T$, is given in Figure 4.3(a) and is connected. Given the decomposition of the network into $M = 8$ teams, we can construct the following communication schedules $\text{sched}_{ij}$ for all robots $r_{ij}$.
Figure 4.3: Graph of teams for a network of $N = 8$ robots. Figure 4.3(a) depicts the graph $G_T$ when the robots divided into $M = 8$ teams as: $\mathcal{T}_1 = \{r_{12}, r_{18}\}$, $\mathcal{T}_2 = \{r_{12}, r_{23}\}$, $\mathcal{T}_3 = \{r_{23}, r_{34}\}$, $\mathcal{T}_4 = \{r_{14}, r_{45}\}$, $\mathcal{T}_5 = \{r_{56}, r_{45}\}$, $\mathcal{T}_6 = \{r_{56}, r_{67}\}$, $\mathcal{T}_7 = \{r_{67}, r_{78}\}$, $\mathcal{T}_8 = \{r_{18}, r_{78}\}$. Figure 4.3(b) depicts the graph $G_T$ when the robots divided into $M = 5$ teams as: $\mathcal{T}_1 = \{r_{12}, r_{14}, r_{15}\}$, $\mathcal{T}_2 = \{r_{12}, r_{23}, r_{25}\}$, $\mathcal{T}_3 = \{r_{23}, r_{34}, r_{35}\}$, $\mathcal{T}_5 = \{r_{15}, r_{25}, r_{35}, r_{45}\}$.

\[
\begin{bmatrix}
\text{sched}_{12} \\
\text{sched}_{23} \\
\text{sched}_{34} \\
\text{sched}_{45} \\
\text{sched}_{56} \\
\text{sched}_{67} \\
\text{sched}_{78} \\
\text{sched}_{81}
\end{bmatrix}
= \begin{bmatrix}
1 & 2 \\
3 & 2 \\
3 & 4 \\
3 & 4 \\
5 & 6 \\
7 & 6 \\
7 & 8 \\
1 & 8
\end{bmatrix}\omega.
\]

Notice that the period of these schedules is $T = 2$. At epoch $z = 1$ the robots in teams $\mathcal{T}_1$, $\mathcal{T}_3$, $\mathcal{T}_5$, $\mathcal{T}_7$ communicate and decide on their next paths so that they re-connect after $T = 2$ discrete time steps, i.e., at $z = 3$.

For comparison, we also consider a second denser graph $G_T^2$ with $M = 5$ teams which is shown in Figure 4.3(b). In this case, the robots are organized into the following teams, $\mathcal{T}_1 = \{r_{12}, r_{14}, r_{15}\}$, $\mathcal{T}_2 = \{r_{12}, r_{23}, r_{25}\}$, $\mathcal{T}_3 = \{r_{23}, r_{34}, r_{35}\}$, $\mathcal{T}_4 = \{r_{15}, r_{25}, \ldots\}$.
Figure 4.4: Graphical depiction of the error $\| \hat{x}_g(t) - x(t) \|$, and the uncertainty metric $\lambda_n(C(t))$ with respect to time $t$ considering the graphs $G^1_T$ and $G^2_T$.

$\{r_{14}, r_{34}, r_{45}\}$, and $\mathcal{T}_5 = \{r_{15}, r_{25}, r_{35}, r_{45}\}$, and the communication schedules $\text{sched}_{ij}$ have the following form:

$$
\begin{bmatrix}
sched_{12} \\
sched_{23} \\
sched_{34} \\
sched_{14} \\
sched_{15} \\
sched_{25} \\
sched_{35} \\
sched_{45}
\end{bmatrix} =
\begin{bmatrix}
1 & 2 & X \\
3 & 2 & X \\
3 & 4 & X \\
1 & 4 & X \\
1 & X & 5 \\
X & 2 & 5 \\
3 & X & 5 \\
X & 4 & 5
\end{bmatrix}^{\omega}.
$$

The comparative results for these graphs are depicted in Figure 4.4. Specifically, Figure 4.4 shows the evolution of the localization error $e_l(t) = \| \hat{x}_g(t) - x(t) \|$, and
Figure 4.5: Comparison of the evolution of the estimates $\hat{x}_g(t)$, $\hat{x}_{T_2}(t)$, and the uncertainty metric $\lambda_n(C(t))$ when the teams are organized according to $G^1_T$. Each purple dashed line is associated with an epoch. Specifically, the $z$-th purple line corresponds to the time instant $t$ at which all robots have reached the $z$-th epoch. The thick purple dashed line denotes the epoch up to which the evolution of the team estimate $\hat{x}_{T_2}(t)$ matches exactly the evolution of $\hat{x}_g(t)$.

the uncertainty metric $\lambda_n(C(t))$ as a function of discrete time $t$. In the error metric $e(t)$, $x(t)$ is a vector that stacks the true positions of all targets and $\hat{x}_g(t)$ stands for the estimate of $x(t)$ with corresponding covariance matrix $C(t)$ assuming global fusion of all measurements taken by all robots at time $t$. Observe in Figure 4.4, that the proposed algorithm can better estimate the positions of the targets, when the teams are constructed as in $G^1_T$. The reason is that the communication schedules $s\text{ched}_{ij}$ in $G^1_T$ allow more teams to visit multiple targets at the same time compared to when the teams are determined according to $G^2_T$. 

103
Next, we compare these two team topologies in terms of the delays they introduce in propagating data across teams. To this end, we define the following metric that encapsulates the average delay with which team $T_i$ receives information from all other teams:

$$
\bar{e}_d^i(t) = \frac{1}{t - t^*} \sum_{t^*}^t \|\hat{x}_g(t) - \hat{x}_{T_i}(t)\|.
$$

In (4.9), $t^*$ stands for the time instant at which the global state estimate $\hat{x}_g(t)$, and the local state estimate $\hat{x}_{T_i}(t)$, after fusing measurements of team $T_i$, differ for the first time.\(^2\) For instance, this time instance $t^*$ is illustrated in Figure 4.5 for team $T_2$ of $G_T^1$. Observe that the evolution of the estimate $\hat{x}_{T_i}(t)$ matches exactly the evolution of $\hat{x}_g(t)$ up to 4 epochs earlier, as expected from Proposition 4.2.3. Specifically, according to Proposition 4.2.3, we have that $D_{G_T^2} = (2 - 1) \times 5 = 5$ which means that at any epoch $z$ all teams will fuse all measurements taken by the robots until the epoch $z - 5$. As a result, $\hat{x}_{T_i}(t)$ should coincide with $\hat{x}_g(t)$ at least until this epoch.

Recall that the discrepancy between $\hat{x}_g(t)$ and $\hat{x}_{T_i}(t)$ is due to the fact that the network gets disconnected and, therefore, measurements taken by team $T_i$ are not communicated to team $T_j$ instantaneously. The evolution of the error $\hat{e}_d^i(t)$ is depicted in Figure 4.6 for team $T_2$ for both team structures. Notice that $\hat{e}_d^2(t)$ is always larger for $G_T^1$, as expected due to Proposition 4.2.3. Specifically, the delay for the graph $G_T^2$ is $D_{G_T^2} = (3 - 1) \times 2 = 4$ and is smaller than $D_{G_T^1} = 5$. To further illustrate this observation, in Figure 4.6, we also plot for team $T_2$ the value of $t^*$ for time instances at which communication within that team happens. Note that the time interval $t^*(t) - t$ is always larger when the teams are organized as in $G_T^1$. We conclude that the selection of the the graph $G_T$ depends on the specific problem and should achieve a balance between less uncertainty and smaller delays.

\(^2\) Note that the time instants $t^*$ depends on the considered team and epoch.

104
In Figure 4.7, we plot the paths that teams $\mathcal{T}_4$ and $\mathcal{T}_3$ from graph $\mathcal{G}^1_T$ designed at the epochs $z = 4$ and $z = 5$, respectively. Recall, that $r_{34} \in \mathcal{T}_3 \cap \mathcal{T}_4$. At $z = 4$, robots of team $\mathcal{T}_4$ design paths to decrease the uncertainty of target 3. However, observe in Figure 4.7(a) that the robots, at the end of their paths, cannot sense target 3 due to an obstacle. Therefore, the uncertainty of that target will not decrease when the robots travel along these paths. The reason for this behavior is that the paths are designed based on the predicted state given the filter initialization and available local information. These paths will lead to connected graph for team $\mathcal{T}_4$ at epoch.
Figure 4.7: Figures 4.7(a) and 4.7(b) depict the paths of teams $T_4$ and $T_3$, as per $G^1_T$, designed at epochs $z = 4$ and $z = 5$, and the corresponding paths of the targets during the time that the robots traverse these path segments. In these figures, the black straight lines show the obstacles and the squares indicate the starting point of the paths. Moreover, the circles and stars correspond to the end of the path segments for the robots and targets, respectively.

Observe also in Figure 4.7(b) that the path for robot $r_{34} \in T_3 \cap T_4$ starts at the end of its path designed at the previous epoch $z = 4$, by construction of Algorithm 5. The trajectories of the robots and targets for the graph $G^1_T$, and the sequence of meeting times and locations are shown in the following video file: http://people.duke.edu/~rk157/Movies/DSEstimation.avi.

4.3.2 Comparison with Alternative Approaches

In this section, we compare the proposed algorithm to an algorithm that preserves network connectivity for all time and a heuristic approach that allows the network to disconnect but selects random meeting locations for each team. In what follows, we employ a team decomposition as in $G^1_T$. Specifically, for the connectivity preserving approach, we assume that the network of $N = 8$ robots maintains a fixed connected configuration throughout the whole experiment. Then, to design informative paths for this network, we apply the RRT* algorithm to design paths for the geometric center of this configuration, so that the uncertainty of the most uncertain target drops...
below a threshold $\delta$. Once the connected network travels along the resulting path, the RRT* algorithm is executed again to find the next informative path. Notice that the paths constructed for the all-time connected network are asymptotically optimal. On the other hand, in the proposed heuristic the path segments $p_{ij}^k$ are selected to be the geodesic paths that connect the initial locations to the randomly selected meeting location. When traveling along the paths designed by this heuristic, the time constraint is not satisfied and the robots need to wait in their meeting locations for their team members.

Figure 4.8 shows the evolution of the localization error $e_\ell(t) = \|\hat{x}_g(t) - x(t)\|$, and the uncertainty metric $\lambda_n(C(t))$ as a function of time $t$ for both approaches compared to our method. Notice that our method outperforms the algorithm that requires all robots to remain connected for all time. The reason is that our proposed algorithm allows the robots to disconnect in order to visit multiple targets simultaneously which cannot happen using an all-time connectivity algorithm. Observe also that our algorithm performs better than the heuristic approach.

The heuristic approach performs better in Figure 4.8 than the algorithm that enforces end-to-end network connectivity for all time. Nevertheless, this is not always the case. To elaborate more, we compare the performance of the algorithms for 4 and 8 targets in Table 4.1 in terms of the average error $\bar{e}_\ell = \sum_{t=0}^{t_{end}} \|\hat{x}_g(t) - x(t)\|/t_{end}$ and the average uncertainty $\bar{\lambda} = \sum_{t=0}^{t_{end}} \lambda_n(C(t))/t_{end}$, where $t_{end}$ is the total number of time steps. Observe that as we decrease the number of targets, the all-time connectivity algorithm performs better than the heuristic approach. The reason for this is that the all-time connectivity approach forces the network of robots to visit the targets sequentially. Thus, for larger number of targets it takes longer to revisit a specific target and this results in uncertainty spikes. On the other hand, the heuristic approach selects the meeting locations randomly. Thus, as we populate the domain with targets, the paths designed by the heuristic cross nearby a target with larger
probability which improves its performance. Our proposed algorithm outperforms both approaches regardless of the number of targets. In Table 4.2, for the case of 4 targets, we considered a network with $N = 4$ robots divided in the following teams $\mathcal{T}_1 = \{r_{12}, r_{14}\}$, $\mathcal{T}_2 = \{r_{12}, r_{23}\}$, $\mathcal{T}_3 = \{r_{23}, r_{34}\}$, and $\mathcal{T}_4 = \{r_{34}, r_{14}\}$.

Finally observe that the local estimation results for team $\mathcal{T}_2$ in Figure 4.5 are still considerably better than the results of competitive methods shown in Figure 4.8. Particularly, for 8 targets the numerical values for $\mathcal{T}_2$ are $\bar{e}_\ell = 1.044 \pm 0.628m$ and $\bar{\lambda} = 0.177 \pm 0.071m^2$, respectively. This is important since in reality, a user would access the information of an individual robot and not the whole network. In a similar scenario for all-time connected case, the whole network needs to be recalled to an access point which would disrupt the estimation task. Furthermore, it is more
probable that the robots would be assigned with another task that involves using their estimates. In such a scenario, the most relevant targets are the ones that are close to a team for which the local estimate of that team would be the most accurate in the network.

4.4 Steering function

In this section we present the steering function discussed in Section 4.2.5 for robots with single-integrator dynamics. This function is described in Algorithm 10 and tries to synchronize the time of the robots by adjusting their step sizes or stalling them altogether if the time difference cannot be compensated in the current step. In line 1, we project the vector $v_{\text{rand}} - v_{\text{nearest}}$ onto the space of every robot $r_{ij} \in T_i$. The resulting vector is denoted by $d_{ij}$. Let $\bar{\epsilon} > 0$ be a user-specified parameter that determines the maximum step size of the robots. Then, in line 2 we compute the smallest step size $s$ among robots as $s = \min_{r_{ij} \in T_i} (\bar{\epsilon}, ||d_{ij}||)$. Next, in line 3 we compute the time $\Delta t$ required by this robot to travel distance $s$ if it moves with maximum velocity $u_{\text{max}}$, i.e., $\Delta t = s/u_{\text{max}}$. In line 5 we compute the quantity

Table 4.1: Comparison with Alternative Approaches

<table>
<thead>
<tr>
<th></th>
<th>all-time</th>
<th>heuristic</th>
<th>intermittent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\epsilon}_{\ell}$ (m)</td>
<td>2.517 ± 0.810</td>
<td>1.201 ± 0.570</td>
<td>0.929 ± 0.561</td>
</tr>
<tr>
<td>$\bar{\lambda}$ (m$^2$)</td>
<td>0.563 ± 0.129</td>
<td>0.334 ± 0.161</td>
<td>0.177 ± 0.068</td>
</tr>
</tbody>
</table>

Table 4.2: Comparison with Alternative Approaches

<table>
<thead>
<tr>
<th></th>
<th>all-time</th>
<th>heuristic</th>
<th>intermittent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\epsilon}_{\ell}$ (m)</td>
<td>1.030 ± 0.466</td>
<td>1.993 ± 0.566</td>
<td>0.879 ± 0.461</td>
</tr>
<tr>
<td>$\bar{\lambda}$ (m$^2$)</td>
<td>0.226 ± 0.061</td>
<td>0.405 ± 0.181</td>
<td>0.167 ± 0.084</td>
</tr>
</tbody>
</table>
Algorithm 10: Steer

**Input:** Nodes $v_{\text{nearest}}$ and $v_{\text{rand}}$

1. Compute vector $d_{ij}$ by projecting $v_{\text{rand}} - v_{\text{nearest}}$ onto the space of robot $r_{ij} \in T_i$;
2. Compute $s = \min_{r_{ij} \in T_i} (\epsilon, ||d_{ij}||)$;
3. Compute travel time $\Delta t = s/u_{\text{max}}$;
4. Compute the compensated time for each robot:
5. $\Delta t_{c,ij} = \Delta t - (t_{ij}(v_{\text{nearest}}) - \min_{r_{ij} \in T_i} t_{ij}(v_{\text{nearest}}))$
6. if $\Delta t_{c,ij} > 0$ then
   7. $v_{\text{new},ij} = v_{\text{nearest},ij} + u_{\text{max}} \Delta t_{c,ij} d_{ij}/||d_{ij}||$;
   8. else
   9. $v_{\text{new},ij} = v_{\text{nearest},ij}$;

$\Delta t_{c,ij} = \Delta t - (t_{ij}(v_{\text{nearest}}) - \min_{r_{ij} \in T_i} t_{ij}(v_{\text{nearest}}))$ that captures the time that each robot can compensate by traveling along its path segment. If $\Delta t_{c,ij} > 0$, then the next sample for robot $r_{ij}$, denoted by $v_{\text{new},ij}$, is given by a point on the line connecting $v_{\text{nearest},ij}$ and $v_{\text{rand},ij}$, i.e., $v_{\text{new},ij} = v_{\text{nearest},ij} + u_{\text{max}} \Delta t_{c,ij} d_{ij}/||d_{ij}||$. If $\Delta t_{c,ij} \leq 0$, the robot $r_{ij}$ does not move, i.e., $v_{\text{new},ij} = v_{\text{nearest},ij}$ [lines 6-9, Alg. 10]. This ensures that the robots will have similar times at $v_{\text{new}}$ if they move or the discrepancy between their times is decreased otherwise. Eventually, all the robots will start to move and their times will match.
Distributed Intermittent Connectivity Control for Mobile Robot Networks under Time-Critical Dynamic Tasks

In this chapter, we consider robots that are tasked with accomplishing high-level, dynamic time-critical tasks. The assigned tasks are (i) dynamic in the sense that the task specifications can change with time, and (ii) time-critical meaning that the information the robots collect as they navigate the workspace is time-critical and, therefore the robots should hold onto the gathered data for a long time; instead, robots have to communicate with each other before they visit a specified number of task waypoints. As in Chapter 3, we assume that the robots have limited communication capabilities and can only communicate when they are physically close to each other. Our goal in this chapter is to design controllers that ensure that the assigned dynamic time-critical tasks are accomplished, the communication network is connected over time infinitely often, and a user-specified metric is minimized.
5.1 Problem Formulation

Consider \( N \geq 1 \) mobile robots operating in a workspace \( W \subset \mathbb{R}^d, \, d = 2, 3 \), and let
\[
\dot{x}_i(t) = f_i(x_i(t), u_i(t))
\]
denote the equations of motion of robot \( i \), where \( x_i(t) \in \mathbb{R}^d \) and \( u_i(t) \in \mathbb{R}^d \) are the position and control input of robot \( i \), respectively, at time \( t \geq 0 \). Let \( \mathcal{N} = \{1, \ldots, N\} \) denote the set of all robots.

We assume that the robots have to accomplish a time-critical dynamic task, defined as \( \mathcal{H}_i = \{p^1_i, \ldots, p_h^i, \ldots, p^{H_i}_i\} \), where \( p_h^i \in W \), are waypoints associated with locations in space where the tasks take place, \( h_i \in \{1, \ldots, H_i\} \), and \( H_i \in \mathbb{N}_+ \). Note that we impose no restriction on the structure of the sequence \( \mathcal{H}_i \), i.e., it can be periodic or aperiodic, and \( H_i \) can be finite or infinite. Moreover, we assume that the tasks \( \mathcal{H}_i \) are not \textit{a priori} known to the robots and instead, they are revealed over time. Specifically, at any time \( t \), every robot \( i \) has access to a part of the task \( \mathcal{H}_i \), defined as \( \mathcal{H}_i^{\text{cur}}(t) = \{p^0_i(t), \ldots, p^\xi_i(t)\} \subseteq \mathcal{H}_i \), where \( \xi^0_i(t) \) through \( \xi^\xi_i(t) \) are consecutive indices to \( \{1, \ldots, H_i\} \) and point to the entries in \( \mathcal{H}_i \) that are the first and the last entries in \( \mathcal{H}_i^{\text{cur}}(t) \); see also Figure 6.1(a). The current tasks \( \mathcal{H}_i^{\text{cur}}(t) \) can be updated as the robots navigate the workspace, by adding to them additional waypoints from \( \mathcal{H}_i \). Specifically, the current task \( \mathcal{H}_i^{\text{cur}}(t^+) \) of robot \( i \) at time \( t^+ \), right after an update at time \( t \), is constructed as \( \mathcal{H}_i^{\text{cur}}(t^+) = \mathcal{H}_i^{\text{cur}}(t) \cup \{p^\xi_i(t^+), \ldots, p_{i+1}^\xi(t^+)\} \subseteq \mathcal{H}_i \).

The time instants \( t \) when the current tasks \( \mathcal{H}_i^{\text{cur}}(t) \) are updated, as well as, and the corresponding the new task specifications/waypoints \( \{p^\xi_i(t^+), \ldots, p_{i+1}^\xi(t^+)\} \) are determined on-line and are not known \textit{a priori}. Also, to ensure that \( \mathcal{H}_i^{\text{cur}}(t) \) are always finite, every robot \( i \) deletes from \( \mathcal{H}_i^{\text{cur}}(t) \) all waypoints that they have already visited.
Moreover, the assigned tasks are *time-critical* in the sense that the information collected by the robots as they visit waypoints included in $\mathcal{H}_i^{\text{cur}}(t)$ is time-critical and, as result, they should not hold onto the gathered data for a long time. Instead, they have to communicate to other robots frequently enough, according to desired specifications. Specifically, we require that every robot $i$ should communicate with other robots before visiting a specified number of waypoints included in $\mathcal{H}_i^{\text{cur}}(t)$ since the last communication event they participated. This allowed number of task waypoints that they can visit without communicating can change with time based, e.g., on the importance of the collected data.

To define a communication network among the robots, we partition the robot team into $M \geq 1$ robot teams $\mathcal{T}_m$, $m \in \{1, \ldots, M\}$, so that the resulting team membership graph $\mathcal{G}_T$, defined in Definition 2.1.1, in Chapter 2, is connected. In this chapter, we assume that the teams are *a priori* known and can be selected arbitrarily as long as the graph of teams $\mathcal{G}_T$ is connected. Since the robots have limited communication capabilities, we assume that the robots in every team $\mathcal{T}_m$ can only communicate if all of them are simultaneously present at a common location $v_j \in \mathcal{W}$, hereafter called a communication point. We assume that there are $R \geq 1$ available communication points in the workspace at locations $v_j \in \mathcal{W}$, where $j \in \mathcal{C} \subset \mathcal{I}$. The indices $j$ of the communication points $v_j$ where communication can take place for the robotic team $\mathcal{T}_m(t)$ are collected in a finite and fixed set $\mathcal{C}_m \subseteq \mathcal{C}$, where the sets $\mathcal{C}_m$ are not necessarily disjoint.

We assume that the robots have limited communication capabilities and, therefore, they can communicate only if they are physically close to each other at a common location in space, hereafter called a communication point. Specifically, we assume that there are $R \geq 1$ available communication points at locations $v_j \in \mathcal{W}$, for $j = 1, \ldots, R$, and we denote by $\mathcal{C} = \{1, \ldots, R\}$ the index set of all communication points. The locations $v_j$ where communication can take place for the robotic team
\( \mathcal{T}_m \) are collected in a finite and fixed set \( \mathcal{C}_m \subseteq \mathcal{C} \), where the sets \( \mathcal{C}_m \) are not necessarily disjoint. When all robots in a team \( \mathcal{T}_m \) have arrived at a common communication location, we assume that communication happens and the robots leave to accomplish their tasks or communicate with other teams. This way, a dynamic robot communication network is constructed as defined in Definition 2.1.2, in Chapter 2. Moreover, we assume that the communication points \( v_j, j \in \mathcal{C}_m \) for the first communication event of all teams \( \mathcal{T}_m \) are also user-specified. To ensure that information is continuously transmitted across the network of robots, we require that the communication graph \( \mathcal{G}_c(t) \) is connected over time infinitely often, i.e., that all robots in every team \( \mathcal{T}_m \) meet infinitely often at a common communication point \( v_j, j \in \mathcal{C}_m \), that does not need to be fixed over time.

The goal in this chapter is to design paths \( \mathcal{P}_i(t) \) for all robots \( i \) so that the assigned tasks \( \mathcal{H}_i^{\text{cur}}(t) \) are accomplished, the intermittent connectivity requirement is satisfied, and a user defined cost \( \sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t)) \) is minimized, where

\[
J(\mathcal{P}_i(t)) = \sum_{k_i=1}^{K_i(t)-1} w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t)).
\] (5.2)

Specifically, the paths \( \mathcal{P}_i(t) \) consist of the waypoints in \( \mathcal{H}_i^{\text{cur}}(t) \) in the given order as well as communication points from the sets \( \mathcal{C}_m \) associated with the teams \( \mathcal{T}_m \) to which robot \( i \) belongs. Note that any communication points from these sets \( \mathcal{C}_m \) can enter \( \mathcal{P}_i(t) \) in any order. Optimization of the cost in (5.2) ensures that the communication points are selected and placed in \( \mathcal{P}_i(t) \) optimally. Moreover, in (5.2), \( K_i(t) \) denotes the number of waypoints in \( \mathcal{P}_i(t) \), \( \mathcal{P}_i^{k_i}(t) \) stands for the \( k_i \)-th waypoint in \( \mathcal{P}_i(t) \), and \( w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t)) \) represents the cost to transition from \( \mathcal{P}_i^{k_i}(t) \) to \( \mathcal{P}_i^{k_i+1}(t) \). Hereafter, we define the transition cost \( w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t)) \) as the distance between \( \mathcal{P}_i^{k_i}(t) \) and \( \mathcal{P}_i^{k_i+1}(t) \), i.e.,

\[
w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t)) = \| \mathcal{P}_i^{k_i}(t) - \mathcal{P}_i^{k_i+1}(t) \|.
\] (5.3)
Figure 5.1: Graphical illustration of the problem formulation 5.1.1. A network of $N = 3$ robots (colored dots) divided into $M = 3$ teams is depicted. The robot teams are selected to be: $\mathcal{T}_1 = \{1, 2\}$, $\mathcal{T}_2 = \{1, 3\}$, and $\mathcal{T}_3 = \{3, 2\}$. The green polygons, the blue square, and the red star stand for the communication points in the sets $C_1$, $C_2$, and $C_3$, respectively. Figure 6.1(a) illustrates the sequences $\mathcal{H}_i^{\text{cur}}(t)$ and Figure 6.1(b) depicts the paths $\mathcal{P}_i(t)$ that include the task waypoints of $\mathcal{H}_i^{\text{cur}}(t)$ and the communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. The black rectangle stands for an obstacle. The communication schedules are $\text{sched}_1 = [1, 2, X]^\omega$, $\text{sched}_2 = [1, X, 3]^\omega$, and $\text{sched}_3 = [X, 2, 3]^\omega$.

Note that alternative transition costs $w$ can be defined that can capture, e.g., consumed energy or travel time. The problem that is addressed in this paper can be summarized as follows and illustrated in Figure 5.1.

**Problem 5.1.1.** Given dynamic task specifications $\mathcal{H}_i^{\text{cur}}(t)$ and fixed teams $\mathcal{T}_m$, $m \in \{1, \ldots, M\}$, select respective communication points $v_j$, $j \in \mathcal{C}_m$ so that the robot paths $\mathcal{P}_i(t)$ for all $i \in \mathcal{N}$ satisfy: (i) the assigned tasks are accomplished, i.e., all robots $i$ go through all waypoints of $\mathcal{H}_i^{\text{cur}}(t)$ in the order they appear in $\mathcal{H}_i^{\text{cur}}(t)$; (ii) the communication graph $G_c(t)$ is connected over time infinitely often; (iii) all robots $i \in \mathcal{N}$ share the collected time-critical information frequently enough with all robots in teams $\mathcal{T}_m$, for all $m \in \mathcal{M}_i$, according to desired specifications; and (iv) the total cost function $\sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t))$ is minimized.

To solve Problem 5.1.1, we propose a distributed algorithm that decouples the
task planning and the intermittent communication problem. Specifically, first we
design schedules of communication events for all robots, independently of the as-
signed task, that ensure intermittent communication among robots in every team
\( T_m(t) \) infinitely often, as in Chapter 2. Given these schedules we construct initial
paths \( P_i(t_0) \), for all \( i \in N \), using the task specifications \( H^{cur}_i(t_0) \), if they are available,
and the user-specified initial communication points \( v_j, j \in C_m \), where \( t_0 \) stands for
the initial time. Then, given \( P_i(t_0) \) we also compute the time instants when the first
communication event within every team will occur. The paths \( P_i(t) \) are updated
online for all \( t > t_0 \) so that that the assigned tasks \( H^{cur}_i(t) \) are accomplished, the
intermittent connectivity requirement is satisfied, and the total cost \( \sum_{i \in N} J(P_i(t)) \)
is minimized. To achieve this, we select communication points that are introduced
in the dynamic tasks \( H^{cur}_i(t) \) so that the total cost \( \sum_{i \in N} J(P_i(t)) \) is minimized.

5.2 Integrated Task Planning and Intermittent Communication Con-
trol

In this section, we synthesize paths \( P_i(t) \) that satisfy the assigned tasks \( H^{cur}_i(t) \), the
intermittent connectivity requirement, and minimize the total cost \( \sum_{i \in N} J(P_i(t)) \).
To achieve this, we select communication points that are introduced in the dynamic
tasks \( H^{cur}_i(t) \) so that the total cost \( \sum_{i \in N} J(P_i(t)) \) is minimized while respecting the
communication schedules \( \text{sched}_i \) constructed in Chapter 2. In Section 5.2.1, we
present the distributed initialization of the paths \( P_i(t_0) \) and when these paths can be
updated by adding new task waypoints or communication points. Next, in Sections
5.2.2-5.2.3, we discuss how the robots select the communication points for their teams
and the time instants when they should communicate and how these decisions are
integrated with the paths \( P_i(t) \). Finally, in Section 5.2.4, we discuss how the paths
\( P_i(t) \) are executed online.
5.2.1 Construction of paths $\mathcal{P}_i(t)$

Initialization

Using the schedules $\text{sched}_i$ constructed in Chapter 2 for a given decomposition of the network into fixed teams $\mathcal{T}_m$, we design the initial paths $\mathcal{P}_i(t_0)$ that include (i) all waypoints in $\mathcal{H}^\text{cur}_{i}(t_0)$ in the order they appear in $\mathcal{H}^\text{cur}_{i}(t_0)$, and (ii) the user-specified communication points $v_j$, $j \in C_m$, for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. Specifically, first the paths $\mathcal{P}_i(t_0)$ are initialized as $\mathcal{P}_i(t_0) = \mathcal{H}^\text{cur}_{i}(t_0)$. If the task specifications $\mathcal{H}^\text{cur}_{i}(t_0)$ are not available, then the paths $\mathcal{P}_i(t_0)$ are initialized as $\mathcal{P}_i(t_0) = \emptyset$. Then, the paths $\mathcal{P}_i(t_0)$ are updated by incorporating into them the user-specified communication points $v_j$, $j \in C_m$, for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. The index $k^m_i \in \{1, \ldots, K_i(t_0)\}$ of the entry in $\mathcal{P}_i(t_0)$ where the communication point $v_j$, $j \in C_m$ for team $\mathcal{T}_m$ will be placed can be selected either arbitrarily or optimally so that the cost function $J(\mathcal{P}_i(t_0))$ is minimized. The only requirement is that the communication points are introduced in $\mathcal{P}_i(t_0)$ in the order the respective communication events appear in $\text{sched}_i$, for all $i \in \mathcal{N}$. In this way, we ensure that the communication events during the execution of the paths $\mathcal{P}_i(t_0)$ will occur in the order determined by the $\text{sched}_i$. Note that during the initialization phase, the $X$’s that appear in $\text{sched}_i$ are ignored and are not introduced in the paths $\mathcal{P}_i(t_0)$.

Online Construction

At any time $t$ every robot $i$ can update the current task $\mathcal{H}^\text{cur}_{i}(t)$ by appending additional waypoints from $\mathcal{H}_i$, as discussed in Section 5.1. The additional waypoints are appended to the paths $\mathcal{P}_i(t)$, as well. Moreover, when the robots $i \in \mathcal{T}_m$ meet at the respective communication point that appears in their paths $\mathcal{P}_i(t)$, they communicate and coordinate to select the next communication point for team $\mathcal{T}_m$ and the time instant when they will communicate again, and design their corresponding paths $\mathcal{P}_i(t^+)$ that they will have at the time instant $t^+$, i.e., right after leaving this
communication point. This coordination process is described in Sections 5.2.2-5.2.3.

5.2.2 Selection of Next Communication Point

To select the next communication point $v_j, j \in C_m$ for team $T_m$ and incorporate it into $P_i(t)$ giving rise to the paths $P_i(t^+)$, the robots $i \in T_m(t)$ solve the following integer program.

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in C_m, \{k_i^m\} \in T_m} J(P_i(t^+)) \\
\text{subject to} & \quad P_i^{k_i^m}(t^+) = v_j, \\
& \quad k_i^m > k_i^{LC}(t), \\
& \quad k_i^m \geq k_i^o(t), \text{ where,} \\
& \quad K_i(t^+) \geq k_i^o(t) \geq \min(k_i^{LC}(t) + 2, K_i(t^+)), \\
& \quad k_i^m \leq k_i^b(t), \text{ where } K_i(t^+) \geq k_i^b(t) \geq k_i^o(t),
\end{align*}
\]

In the optimization problem 5.4 the paths $P_i(t^+)$ are initialized as $P_i(t^+) = P_i(t)$. In the objective function (5.4a), $J(P_i(t^+))$ stands for the cost of the path $P_i(t^+)$ defined in (5.2). Also, $K_i(t^+)$ stands for the number of waypoints in $P_i(t^+)$. Note that $K_i(t^+) = K_i(t) + 1$ since $P_i(t^+)$ includes all waypoints of $P_i(t)$ and the next communication point for team $T_m$ that does not exist in $P_i(t)$. Moreover, $k_i^m$ represents the index of the entry in $P_i(t^+)$ where the selected communication point $v_j, j \in C_m$ will be placed, i.e., $P_i^{k_i^m}(t^+) = v_j, j \in C_m$.

The first constraint (5.4b) requires that all robots $i \in T_m$ will select the same communication point $v_j, j \in C_m$ for the next communication event associated with team $T_m$ and incorporate it into the entry of $P_i(t^+)$ with index $k_i^m$. The second constraint (5.4c) ensures that all communication points $v_j, j \in C_m$, for every team
Figure 5.2: Graphical illustration of optimization problem (5.4). Robots 1 and 2 (black dots) in team $T_1$ meet at the selected communication point $P_1(t) = P_2(t)$ (green polygon) and coordinate to select the next communication point for team $T_1$. Red and blue squares stand for the waypoints that robots 1 and 2 have to visit to accomplish their tasks, respectively. The communication points for teams $T_2$ and $T_3$ are represented by colored stars. The resulting paths $P_1(t^+)$ and $P_2(t^+)$ comprise the red and blue, both solid and dashed, lines. The gray line stands for an edge in the path $P_1(t)$ that does not exist in the path $P_1(t^+)$ due to the introduction of the communication point for team $T_1$.

$T_m$, $m \in M_i$ are introduced in $P_i(t^+)$ in the order that the respective communication events appear in $s_i$. In particular, in the second constraint, the index $k_i^{LC}(t)$ is defined as the index of the entry in $P_i(t)$ where the last communication point has been introduced, i.e., none of the waypoints $P_i^{k_i}(t)$, for all $k_i \in \{k_i^{LC}(t) + 1, \ldots, K_i(t)\}$ is a communication point. This constraint requires that robot $i$ will participate at the next communication event for team $T_m$ only after it has visited all other communication points that already exist in $P_i(t^+)$, for all robots $i \in T_m$. This combined with the fact that the communication points appear in the path $P_i(t_0)$ in the order determined by $\text{sched}_i$, for all $i \in N$, entails that the communication points are introduced into all subsequent paths $P_i(t)$, for all $t > t_0$ in the order that is
determined by \( \text{sched}_i = s_i^\omega \), as well, for all \( i \in \mathcal{N} \); see also Example 5.2.1. As it will be shown in Proposition 5.3.1, this constraint ensures that the network never reaches a deadlock configuration and guarantees intermittent communication infinitely often. Notice that the symbols \( X \) that appear in the schedules \( \text{sched}_i = s_i^\omega \) are ignored and are not introduced in the paths \( \mathcal{P}_i(t^+) \).

The last two constraints (5.4d)-(5.4e) are additional constraints for \( k_i^m \) and determine how frequently communication events should occur. Specifically, the second constraint requires that the index \( k_i^m \) is greater than \( k_i^a(t) \) which is also an index of entries of the path \( \mathcal{P}_i(t) \) and can change with time. The index \( k_i^a(t) \) is selected under the following two requirements. First, to ensure feasibility of the optimization problem (5.4a), \( k_i^a(t) \) should satisfy \( K_i(t^+) \geq k_i^a(t) \), for all \( n_i \geq 1 \), since there are only \( K_i(t^+) \) possible entries for the communication point \( v_{j,n} \), \( j \in \mathcal{C}_m \) in the path \( \mathcal{P}_i(t^+) \).

Second, we require that \( k_i^a(t) \) is selected so that \( k_i^a(t) \geq \min(k_i^{LC}(t) + 2, K_i(t^+)) \). Essentially, this requirement motivates robot \( i \) to visit \( k_i^a(t) - k_i^{LC}(t) - 1 \) waypoints in the path \( \mathcal{P}_i(t^+) \) that are not communication points, if there are such waypoints, before communicating with another team; see also Figure 5.2. As it will be discussed in Theorem 5.3.2, this ensures that robot \( i \) will accomplish its assigned task, i.e., it will eventually visit all waypoints associated with the assigned task in the path \( \mathcal{P}_i(t^+) \). The last constraint is similar to the second one and it requires that \( k_i^m \) is smaller than \( k_i^b(t) \), which is also an index of entries in the path \( \mathcal{P}_i(t) \). The index \( k_i^b(t) \) is selected so that the inequality \( K_i(t^+) \geq k_i^b(t) \geq k_i^a(t) \) is satisfied to ensure feasibility of (5.4). In other words, the last two constraints require that the index of the next communication point for team \( T_m \) in the path \( \mathcal{P}_i(t^+) \) should belong to \( [k_i^a(t), k_i^b(t)] \subseteq \mathbb{N} \). Finally, notice that the indices \( k_i^m \) are not required to be the same for all robots in team \( T_m \).

**Example 5.2.1** (Optimization Problem (5.4)). Assume that communication within
team $T_1 = \{1, 2\}$ happens as shown in Figure 5.2 and its members coordinate to select their next communication point. Observe in Figure 5.2 that $K_1(t) = 9$, $K_2(t) = 5$, $K_1(t^+) = 10$, and $K_2(t^+) = 6$. Also, observe that $k_1^{LC}(t) = 6$ and $k_2^{LC}(t) = 5$. The parameter $k_1^a(t)$ should satisfy $10 \geq k_1^a(t) \geq \min\{8, 10\} = 8$. In this example, we select $k_1^a(t) = 8$, which means that robot 1 has to visit at least $k_1^a(t) - k_1^{LC}(t) - 1 = 1$ waypoint before communicating again with team $T_1$, after the last communication event at $P_i^{k_1^{LC}(t)}$ for team $T_4$. The parameter $k_1^b(t)$ should satisfy $k_1^b(t) \geq k_1^a(t) = 8$ and we select $k_1^b(t) = 9$. As for robot 2, we have that $k_2^a(t)$ should satisfy $6 \geq k_2^a(t) \geq \min\{6, 7\} = 6$ and, therefore, we select $k_2^a(t) = 6$. Also, $k_2^b(t)$ should satisfy $6 \geq k_2^b(t) \geq 6$ and, thus, we select $k_2^b(t) = 6$. Observe also that $k_2^a(t) - k_2^{LC}(t) - 1 = 0$, i.e., robot 2 will not visit any waypoints associated with the assigned task after the communication event for team $T_3$, since there are no such waypoints. The schedules of robots 1 and 2 are $sched_1 = [1, X, 4]$ and $sched_2 = [1, X, 3]$. Observe that the communication points appear in $P_1(t^+)$ and $P_2(t^+)$ respect the corresponding schedules for both robots.

**Remark 5.2.2** ($k_i^a(t)$ and $k_i^b(t)$). In practice, $k_i^a(t)$ can be selected so that robot $i$ collects a sufficiently large amount of information before communicating with another team. On the other hand, $k_i^b(t)$ controls the amount of information that robot $i$ is allowed to hold onto before sharing it with other robots. For example, if robot $i$ is expected to collect highly critical information at the next waypoints, then $k_i^b(t)$ should be selected small, so that the collected data can be propagated to the network, as soon as possible. Thus, $k_i^a(t)$ and $k_i^b(t)$ can control the frequency at which communication events occur. Moreover, $k_i^b(t)$ can also capture buffer constraints as, e.g., in Guo and Zavlanos (2017).
5.2.3 Selection of Next Meeting Time Instant

In Section 5.1 we assumed that communication between robots in team $\mathcal{T}_m(t)$ happens only when all robots in that team are simultaneously present a common communication location $v_j, j \in \mathcal{C}_m$. Nevertheless, this control policy can lead to large waiting times which can be undesirable in case of time-critical missions, or it can even be infeasible if the robots cannot stay stationary, e.g., due to their dynamics. To avoid waiting delays, once the robots $i \in \mathcal{T}_m(t)$ select their next communication point, they also compute the time instant $t^m$ at which they will communicate again, so that the waiting time at that next communication point is zero.

The computation of the time instant $t^m$, called also meeting time instant, is described in Algorithm 11. First, every robot $i \in \mathcal{T}_m$ computes the last time instant $L_i \in \mathbb{R}_+$ that it needs to participate in a communication event during the execution of the path $\mathcal{P}_i(t)$, i.e., $L_i = \max_{e \in \mathcal{M}_i}(t^e)$ [line 2, Alg. 11]. This communication event takes place at the communication point $\mathcal{P}_{i}^{k_i(t)}(t) = \mathcal{P}_{i}^{k_i(t)}(t^+)$, where $k_i^{LC}(t)$ was defined in Section 5.2.2. Second, given the next communication point $v_j, j \in \mathcal{C}_m$, for team $\mathcal{T}_m$, determined by the solution of (5.4), every robot $i \in \mathcal{T}_m(t)$ computes the minimum time required to travel from the location $\mathcal{P}_{i}^{k_i(t)}(t^+) = \mathcal{P}_{i}^{k_i(t)}(t^+)$ to $v_j = \mathcal{P}_{i}^{\text{index}_i}(t^+)$, denoted by $t^r_i$ [line 3, Alg. 11]. Then, any time instant $t^m \geq \max_{i \in \mathcal{T}_m(t)}(L_i + t^r_i)$, is a feasible time instant. Here, feasibility of $t^m$ means that there exists a controller which given the robot dynamics can drive robot $i$ from the communication point $\mathcal{P}_{i}^{k_i(t)}(t^+)$ to $v_j$ within $t^m - L_i$ time units, for all robots $i \in \mathcal{T}_m$. Design of such a control input for arbitrary robot dynamics is out of the scope of this paper. In this work, we select $t^m = \max_{i \in \mathcal{T}_m(t)}(L_i + t^r_i)$ [line 4, Alg. 11].

Remark 5.2.3 (Initial Meeting Time Instants). Given the initial paths $\mathcal{P}_i(t_0)$, fea-

---

Note that here we assume that if robot $i$ can arrive at a location $v_j$ at time instant $L_i + t^r_i$, then it can also arrive at $v_j$ at $t^m \geq L_i + t^r_i$ which is a reasonable assumption as long as the the robot velocities are not bounded below or have sufficiently small lower bounds.
Algorithm 11: Computation of meeting time instant \( t_m \)

**Input:** Paths \( \mathcal{P}_i(t^+) \), robot dynamics \( \mathbf{x}_i(t) = f_i(\mathbf{x}_i(t), \mathbf{u}_i(t)) \), for all robots \( i \in \mathcal{T}_m \), \( v_{ji} \in \mathcal{C}_m \)

**Output:** Meeting time instant \( t_m \)

1. **for** \( i \in \mathcal{T}_m \) **do**
   2. Compute time at which the latest communication event for robot \( i \) will occur: \( L_i = \max_{e \in \mathcal{M}_i} (t_e) \);
   3. Given dynamics \( \mathbf{x}_i(t) = f_i(\mathbf{x}_i(t), \mathbf{u}_i(t)) \) and the next communication point \( \mathbf{v}_j, j \in \mathcal{T}_m \), compute the minimum required travel time, denoted by \( t_i^{tr} \), from \( \mathcal{P}_i^{k_i}(t^+) \) to \( \mathbf{v}_j \);
   4. Next meeting time instant for team \( \mathcal{T}_m \): \( t_m = \max_{i \in \mathcal{T}_m} (L_i + t_i^{tr}) \);

Possible initial meeting time instants \( t_m \) can be designed if every robot \( i \in \mathcal{N} \) runs Algorithm 11 for each team \( \mathcal{T}_m \), \( m \in \mathcal{M}_i \) in the order determined by \( \mathbf{sched}_i \). Notice that all robots in team \( \mathcal{T}_m \) will run Algorithm 11 for team \( \mathcal{T}_m \) at the same time, by construction of \( \mathbf{sched}_i \), and will collectively compute \( t_m \). In this initialization process, (i) in line 2, we set \( t_e = t_0 \) for all \( t_e, e \in \mathcal{M}_i(t_0) \), that have not been computed at previous iterations of Algorithm 11, and (ii) in line 3, we compute the travel time from the communication point that appears right before the communication point of team \( \mathcal{T}_m \) in \( \mathcal{P}_i(t_0) \) (or \( \mathbf{x}_i(t_0) \) if this communication point is not defined) to the communication point of \( \mathcal{T}_m \). We show via simulations that even if the initial meeting time instants are selected arbitrarily and, as a result, they are not necessarily feasible, resulting in non-zero waiting times, the waiting time will eventually become zero for all teams \( \mathcal{T}_m \), \( m \in \mathcal{M} \). By construction of Algorithm 11, if the initial meeting time instants are feasible, then the waiting time will always be zero for all teams \( m \in \mathcal{M} \).

5.2.4 Online Execution of Paths \( \mathcal{P}_i(t) \)

In this section we discuss how the robots execute their paths \( \mathcal{P}_i(t) \), for all \( t \geq t_0 \). Given the paths \( \mathcal{P}_i(t) \), for any \( t \geq t_0 \), robots start moving towards the next unvisited waypoint in the path \( \mathcal{P}_i(t) \), i.e., the first waypoint in \( \mathcal{P}_i(t) \), denoted by \( \mathcal{P}^1_i(t) \), since
Algorithm 12: Online Execution of $\mathcal{P}_i(t)$, $t \geq t_0$

Input: Sets $C_m, \forall m \in \mathcal{M}, \mathcal{P}_i(t_0)$

while Terminate=0 do

1. Move towards $\mathcal{P}_i^1(t)$;
2. if ($x_i(t) = \mathcal{P}_i^1(t) \land \mathcal{P}_i^1(t) = v_j, j \in C_m, m \in \mathcal{M}_i(t)$) then
   3. Wait until all robots of team $\mathcal{T}_m(t)$ arrive;
   4. Select next communication point, next meeting time, and design new paths;
5. if ($x_i(t) = \mathcal{P}_i^1(t)$) then
   6. Delete the visited waypoint $\mathcal{P}_i^1(t)$ from $\mathcal{P}_i(t)$;

visited waypoints are deleted from $\mathcal{P}_i(t)$ [line 2, Alg. 12]. When robot $i$ reaches the waypoint $\mathcal{P}_i^1(t)$, it checks if this location corresponds to a communication point associated with a team $\mathcal{T}_m, m \in \mathcal{M}_i$ [line 3, Alg. 12]. If this is not the case, then robot $i$ deletes the waypoint $\mathcal{P}_i^1$ from the path $\mathcal{P}_i(t)$, as there is no need to store it anymore, and moves towards the next waypoint $\mathcal{P}_i^1(t)$ [line 7, Alg. 12]. Otherwise, if $\mathcal{P}_i^1(t) = v_j, j \in C_m, m \in \mathcal{M}_i$ robot $i$ communicates and coordinates with all other robots in team $\mathcal{T}_m$ to select next communication point, next meeting time instant, and design new paths $\mathcal{P}_i(t^+)$ [line 5, Alg. 12]. Note that uncertainty and exogenous disturbances may affect the arrival times of the robots at the communication points. Thus, if the robots $i \in \mathcal{T}_m$ are not able to arrive at $v_j, j \in C_m$ at the meeting time instants $t^m$ computed in Section 5.2.3, then they wait for each other until all of them arrive at this communication point [line 4, Alg. 12].

5.3 Correctness

In this section, we present results pertaining to correctness of the proposed control scheme. Specifically, in Theorems 5.3.2 and 5.3.3, we show that when the robots follow the paths $\mathcal{P}_i(t)$ both the assigned task and the intermittent communication requirement are satisfied. To show these results, we first need to show that the system is deadlock-free when the paths $\mathcal{P}_i(t)$ are executed as per Algorithm 12. Specifically,
we assume that there is a deadlock, if there are robots of any team $\mathcal{T}_m$ that are waiting forever at a communication point for the arrival of all other robots of team $\mathcal{T}_m$. The proof of the following result is the same as the proof of Proposition 3.5.4 in Chapter 3 and, therefore, is omitted.

**Proposition 5.3.1** (Deadlock-free). The mobile robot network is deadlock-free when the paths $\mathcal{P}_i(t)$ are executed according to Algorithm 12.

**Theorem 5.3.2** (Tasks). The construction and execution of $\mathcal{P}_i(t)$ as per the proposed algorithm ensures that all robots will accomplish the assigned task.

*Proof.* First, note that by construction of $\mathcal{P}_i(t)$, the paths $\mathcal{P}_i(t)$ preserve the order in which task waypoints appear in $\mathcal{H}_i^\text{cut}(t)$. Therefore, to show this result, it suffices to show that there exists a time instant $t'_i \geq t$ when robot $i$ will visit all task waypoints that appear in the path $\mathcal{P}_i(t)$, for all $t \geq t_0$ and for all $i \in \mathcal{N}$. This is shown by contradiction. Specifically, assume that robot $i$ will never visit any of the task waypoints that appear in $\mathcal{P}_i(t)$. This can happen in two cases. First, this may occur if the network reaches a deadlock configuration which cannot happen due to Proposition 2.4.2. Second, this may happen if robot $i$ always introduces the communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, in consecutive entries of its path $\mathcal{P}_i(t)$ and, therefore, never visits any task waypoints between any two consecutive communication events. Nevertheless, this cannot happen, if there are task waypoints in $\mathcal{P}_i(t)$, due to the second constraint in (5.4) completing the proof. \qed

**Theorem 5.3.3** (Intermittent Connectivity). The construction and execution of $\mathcal{P}_i(t)$ as per the proposed algorithm ensures that the dynamic communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often.

*Proof.* To show that $\mathcal{G}_c(t)$ is connected over time infinitely often, it suffices to show that all teams $\mathcal{T}_m$ communicate infinitely often. This is because every robot belongs
to at least one team $T_m$, by construction of the sets $M_i$, and the team membership graph $G_T$ is connected. To show this, it suffices to show that time interval between two communication events for all teams $T_m$ is finite. First recall, at any time instant $t \geq t_0$, there exists a communication point for all teams $T_m, m \in M_i$, in the paths $P_i(t)$, for all $i \in N$. Second, recall that due to Proposition 2.4.2, it holds that the waiting times of robots at the communication points are bounded and, therefore, the network is deadlock-free. Therefore, we conclude that robots in team $T_m$ will eventually communicate again, for all $m \in M$ completing the proof.

Remark 5.3.4 (Comparison with Chapter 3). Algorithm 3 in Chapter 3 incorporates optimally communication points in periodic paths, which are known a priori, by exploiting their periodicity. Nevertheless, here, we consider arbitrary tasks that are not necessarily satisfied by periodic paths. As a result, (i) the optimality guarantees provided in Proposition 3.5.2 in Chapter 3 do not hold here, and (ii) if we apply both algorithms to periodic tasks the resulting paths will be different.

5.4 Simulation Studies

In this section, a simulation study is provided that illustrates our approach for a network of $N = 15$ robots that reside in a $10 \times 10$ square workspace free of obstacles. Robots are categorized into $M = 12$ teams as follows: $T_1 = \{1, 2, 9\}$, $T_2 = \{3, 4, 5\}$, $T_3 = \{3, 6, 13\}$, $T_4 = \{1, 3, 14\}$, $T_5 = \{2, 5, 6, 11\}$, $T_6 = \{4, 12, 14\}$, $T_7 = \{5, 9, 15\}$, $T_8 = \{4, 9, 12\}$, $T_9 = \{6, 7, 10, 15\}$, $T_{10} = \{7, 8, 11\}$, $T_{11} = \{8, 10, 11, 12\}$, and $T_{12} = \{7, 10, 13\}$ resulting in a connected graph $G_T$. In the workspace, there are $R = 60$ communication points that are randomly located in $W$, where we select $|C_m| = 5$, for all $m \in M$ and $C_m \cap C_n = \emptyset$, for all $m, n \in M$. Also, we assume that the robot dynamics are given by $x_i(t) = u_i(t), \|u_i(t)\| \leq u_{i \text{max}}$.

Robot 1 has to follow a finite path with $H_1 = 4$, which is randomly generated
at the beginning capturing point-to-point navigation tasks Karaman and Frazzoli (2011) or co-safe LTL tasks Bhatia et al. (2010). For all the other robots we select $H_i = \infty$. Specifically, we assume that robots 2 and $N$ have to follow periodic paths forever to accomplish their assigned tasks. These paths are randomly generated at the beginning resembling in this way surveillance Nardi et al. (2016), estimation Lan and Schwager (2016), or LTL tasks Guo and Zavlanos (2017).\footnote{For periodic paths, $K_i(t)$ can be selected arbitrarily large, since it can be viewed as an infinite and known path. As a result, if $K_i(t)$ is greater than the period of a periodic path, then during a single execution of this periodic path, robot $i$ may not necessarily communicate with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, which is not the case in Chapter 3.} Also, the periodic path $\mathcal{P}_N$ goes through a user that receives the collected information. All the other robots have to follow infinite and aperiodic paths. These robots initially construct finite paths which are randomly generated and then they extend those paths at random time instants by a number of waypoints that is randomly selected from $[1, 10]$ resembling tasks in unknown or dynamic environments as, e.g., in Guo and Dimarogonas (2015); Alonso-Mora et al. (2017), or receding horizon planning approaches Hollinger and Singh (2010); Ulusoy and Belta (2014).

The schedules of communication events have the following form:

$$\text{sched}_1 = [1, 4, X, X]^\omega, \quad \text{sched}_9 = [1, 8, 7, X]^\omega,$$

$$\text{sched}_2 = [1, 5, X, X]^\omega, \quad \text{sched}_{10} = [9, 12, X, 11]^\omega,$$

$$\text{sched}_3 = [2, 4, 3, X]^\omega, \quad \text{sched}_{11} = [X, 5, 10, 11]^\omega,$$

$$\text{sched}_4 = [2, 8, 6, X]^\omega, \quad \text{sched}_{12} = [X, 8, 6, 11]^\omega,$$

$$\text{sched}_5 = [2, 5, 7, X]^\omega, \quad \text{sched}_{13} = [X, 12, 13, X]^\omega,$$

$$\text{sched}_6 = [9, 5, 3, X]^\omega, \quad \text{sched}_{14} = [X, 4, 6, X]^\omega,$$

$$\text{sched}_7 = [9, 12, 10, X]^\omega, \quad \text{sched}_{15} = [9, X, 7, X]^\omega,$$

$$\text{sched}_8 = [X, X, 10, 11]^\omega.$$
Moreover, we select $k_i^a(t) = \min(K_i(t^+), k_i^{LC}(t) + 2)$, for all $i \in \mathcal{N}$, which means that robot $i$ has to visit at least one waypoint associated with the assigned task, if there exists such a waypoint in $\mathcal{P}_i(t)$, between consecutive communication events. Also, we assume that every time the robots visit a waypoint related to the assigned task, they collect one packet of information while they should never keep more than three packets that have never been transmitted to other robots. To capture such limitations, we select $k_i^b(t) = \min(K_i(t^+), k_i^{LC}(t) + 4)$, for all $i \in \mathcal{N}$. Notice that the selected values for $k_i^a(t)$ and $k_i^b(t)$ meet all the requirements described in Section 5.2.2 to guarantee feasibility of the optimization problem (5.4), for all $t > t_0$. Observe in Figure 5.3 that all robots visit at least one and at most three waypoints related to the assigned task between consecutive communication events, as required. Note also that in this simulation study, it always holds that $k_i^a(t) = k_i^{LC}(t) + 2$ and $k_i^b(t) = k_i^{LC}(t) + 4$, for all $t \geq t_0$ and for all robots $i \neq 1$. Since robot 1 has to follow a finite path to

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_3.png}
\caption{Graphical depiction of number of waypoints (solid lines) associated with the assigned task that were visited by all robots between consecutive communication events.}
\end{figure}
accomplish its task, there exists a time instant $t'$, where all the locations in its path $P_1(t)$ are only communication points, for all $t \geq t'$. Therefore, there are no waypoints related to the assigned task between communication points. Also, the initial meeting time instants are selected as $t^m = t_0$, for all teams $T_m$, $m \in \mathcal{M}$, which are clearly infeasible. The resulting waiting times of all robots at the communication points are depicted in Figure 5.4(a). Observe that eventually the waiting time at the selected communication points is zero for all teams despite the initially infeasible meeting time instants and the temporary failure of robot 2.

To illustrate that the proposed motion plans ensure intermittent communication among the robots infinitely often, we implement a simple consensus algorithm over the dynamic network $\mathcal{G}_c$. Specifically, we assume that initially all robots generate a random number $v_i(t_0)$ and when all robots $i \in T_m$ meet at $v_j$, $j \in C_m$ they perform the following consensus update $v_i(t) = \frac{1}{|T_m|} \sum_{e \in T_m} v_e(t)$. Figure 5.4(b) shows that eventually all robots reach a consensus on the numbers $v_i(t)$, which means that
communication among robots takes place infinitely often, as proven in Theorem 5.3.3. The simulation video along with its description can be found in SimulationVideo (2017).
In this chapter, we consider robots that are tasked with accomplishing arbitrary dynamic tasks, as in Chapter 5. Similar to the previous chapters, we assume that the robots have limited communication capabilities and can only communicate when they are physically close to each other. Common in the previous chapters is that the network is decomposed into teams that are fixed with time. On the other hand, the method we propose in this chapter allows the robots to update the structure of the teams, effectively allowing them to decide who they should talk to, so that they can better accomplish newly assigned tasks. Our goal in this chapter is to design controllers that ensure that the assigned dynamic tasks are accomplished, the communication network is connected over time infinitely often, and a user-specified metric is minimized.
6.1 Problem Formulation

Consider $N \geq 1$ mobile robots operating in a workspace $W \subset \mathbb{R}^d$, $d = 2, 3$, and let

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t)),$$

(6.1)

denote the equations of motion of robot $i$, where $x_i(t) \in \mathbb{R}^d$ and $u_i(t) \in \mathbb{R}^d$ are the position and control input of robot $i$, respectively, at time $t \geq 0$. Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of all robots.

We assume that the robots that have to accomplish high-level dynamic tasks, defined as $\mathcal{H}_i = \{p_i^1, \ldots, p_i^{h_i}, \ldots, p_i^{H_i}\}$, where $p_i^{h_i} \in \mathcal{W}$, are waypoints associated with locations in space where the tasks take place, $h_i \in \{1, \ldots, H_i\}$, and $H_i \in \mathbb{N}_+$. Note that we impose no restriction on the structure of the sequence $\mathcal{H}_i$, i.e., it can be periodic or aperiodic, and $H_i$ can be finite or infinite. Moreover, we assume that the tasks $\mathcal{H}_i$ are not a priori known to the robots and instead, they are revealed over time.

Specifically, at any time $t$, every robot $i$ has access to a part of the task $\mathcal{H}_i$, defined as $\mathcal{H}_{i,\text{cur}}(t) = \{p_i^{\xi_0^0(t)}, \ldots, p_i^{\xi_f^f(t)}\} \subseteq \mathcal{H}_i$, where $\xi_0^0(t)$ through $\xi_f^f(t)$ are consecutive indices to $\{1, \ldots, H_i\}$ and point to the entries in $\mathcal{H}_i$ that are the first and the last entries in $\mathcal{H}_{i,\text{cur}}(t)$; see also Figure 6.1(a). The current tasks $\mathcal{H}_{i,\text{cur}}(t)$ can be updated as the robots navigate the workspace, by adding to them additional waypoints from $\mathcal{H}_i$. Specifically, the current task $\mathcal{H}_{i,\text{cur}}(t^+) \supseteq \mathcal{H}_{i,\text{cur}}(t)$ of robot $i$ at time $t^+$, right after an update at time $t$, is constructed as $\mathcal{H}_{i,\text{cur}}(t^+) = \mathcal{H}_{i,\text{cur}}(t) \cup \{p_i^{\xi_0^0(t)+1}, \ldots, p_i^{\xi_f^f(t^+)}\} \subseteq \mathcal{H}_i$.

The time instants $t$ when the current tasks $\mathcal{H}_{i,\text{cur}}(t)$ are updated, as well as, and the corresponding the new task specifications/waypoints $\{p_i^{\xi_0^0(t)+1}, \ldots, p_i^{\xi_f^f(t^+)}\}$ are determined on-line and are not known a priori. Also, to ensure that $\mathcal{H}_{i,\text{cur}}(t)$ are always finite, every robot $i$ deletes from $\mathcal{H}_{i,\text{cur}}(t)$ all waypoints that they have already visited.

As part of the assigned tasks, the robots also collect information that needs to
reach all other robots and possibly a user. To define a communication network among the robots, we first partition the robot team into $M \geq 1$ robot subgroups, called also teams, whose structure may change with time. Also, we require every robot to belong to at least one team. The indices $i$ of the robots that belong to the $m$-th team at time $t$ are collected in a set denoted by $\mathcal{T}_m(t)$, for all $m \in \mathcal{M} := \{1, 2, \ldots, M\}$. We define the set that collects the indices of teams that robot $i$ belongs to at time $t$ as $\mathcal{M}_i(t) = \{m|i \in \mathcal{T}_m(t), m \in \mathcal{M}\}$. Given the teams $\mathcal{T}_m(t)$, for all $m \in \mathcal{M}$, we can define the dynamic graph over these teams as follows.

**Definition 6.1.1** (Team Membership Graph $\mathcal{G}_T(t)$). The dynamic graph over the teams $\mathcal{T}_m(t)$, $m \in \mathcal{M}$ is defined as $\mathcal{G}_T(t) = (\mathcal{V}_T(t), \mathcal{E}_T(t))$, where the set of nodes $\mathcal{V}_T(t)$ is indexed by the non-empty teams $\mathcal{T}_m(t)$ and set of edges $\mathcal{E}_T(t)$ is defined as $\mathcal{E}_T(t) = \{(m,n)|\mathcal{T}_m(t) \cap \mathcal{T}_n(t) \neq \emptyset, \forall m,n \in \mathcal{V}_T(t), m \neq n\}$.

We assume that the robots have limited communication capabilities and, therefore, they can communicate only if they are physically close to each other at a common location in space, hereafter called a communication point. Specifically, we assume that there are $R \geq 1$ available communication points at locations $\mathbf{v}_j \in \mathcal{W}$, for $j = 1, \ldots, R$, and we denote by $\mathcal{C} = \{1, \ldots, R\}$ the index set of all communication points. The indices $j$ of the communication points $\mathbf{v}_j$ where communication can take place for the robotic team $\mathcal{T}_m(t)$ are collected in a finite and fixed set $\mathcal{C}_m \subseteq \mathcal{C}$, where the sets $\mathcal{C}_m$ are not necessarily disjoint. Communication between robots in a team $\mathcal{T}_m(t)$ takes place when all robots in that team are present at a common communication point. After a communication event the robots leave the communication point to accomplish their tasks or communicate with other teams. This way, a dynamic communication network is constructed among the robots, defined as follows:

**Definition 6.1.2** (Communication Network $\mathcal{G}_c(t)$). The communication network among the robots is defined as a dynamic undirected graph $\mathcal{G}_c(t) = (\mathcal{V}_c, \mathcal{E}_c(t))$, where
the set of nodes $\mathcal{V}_c$ is indexed by the robots, i.e., $\mathcal{V}_c = \mathcal{N}$, and $\mathcal{E}_c(t) \subseteq \mathcal{V}_c \times \mathcal{V}_c$ is the set of communication links that emerge among robots in team $\mathcal{T}_m(t)$, when they meet at a common communication point $\mathbf{v}_j$, for some $j \in \mathcal{C}_m$, and all $m \in \mathcal{M}$.

To ensure that information is continuously transmitted across the network of robots, we require that the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often, i.e., that all robots in team $\mathcal{T}_m(t)$ meet infinitely often at common communication points $\mathbf{v}_j$, $j \in \mathcal{C}_m$, for all teams $m \in \mathcal{V}_T$ that does not need to be fixed over time. For this, it is necessary that the teams are designed so that the dynamic team membership graph $\mathcal{G}_T(t)$ is connected for all time $t$ and that every robot belongs to at least one team, i.e., $|\mathcal{M}_i(t)| \geq 1$, for all $t$. We assume that initially the teams $\mathcal{T}_m(t_0)$, where $t_0$ denotes the initial time instant, are designed by a user so that these requirements are met. Moreover, we assume that the initial respective communication points for the teams $\mathcal{T}_m(t_0)$ are also user-specified.

The goal in this paper is to design paths $\mathcal{P}_i(t)$ for all robots $i$ so that the assigned tasks $\mathcal{H}_i^{\text{cur}}(t)$ are accomplished, the intermittent connectivity requirement is satisfied, and a user defined cost $\sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t))$ is minimized, where

$$J(\mathcal{P}_i(t)) = \sum_{k_i=1}^{K_i(t)-1} w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t)).$$

(6.2)

Specifically, the paths $\mathcal{P}_i(t)$ consist of the waypoints in $\mathcal{H}_i^{\text{cur}}(t)$ in the given order as well as communication points from the sets $\mathcal{C}_m$ associated with the teams $\mathcal{T}_m(t)$ to which robot $i$ belongs. Note that any communication points from these sets $\mathcal{C}_m$ can enter $\mathcal{P}_i(t)$ in any order. Optimization of the cost in (6.2) ensures that the communication points are selected and placed in $\mathcal{P}_i(t)$ optimally. Moreover, in (6.2), $K_i(t)$ denotes the number of waypoints in $\mathcal{P}_i(t)$, $\mathcal{P}_i^{k_i}(t)$ stands for the $k_i$-th waypoint in $\mathcal{P}_i(t)$, and $w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t))$ represents the cost to transition from $\mathcal{P}_i^{k_i}(t)$ to $\mathcal{P}_i^{k_i+1}(t)$. Hereafter, we define the transition cost $w(\mathcal{P}_i^{k_i}(t), \mathcal{P}_i^{k_i+1}(t))$ as the distance
between $\mathcal{P}_i^k(t)$ and $\mathcal{P}_i^{k+1}(t)$, i.e.,

$$w(\mathcal{P}_i^k(t), \mathcal{P}_i^{k+1}(t)) = \|\mathcal{P}_i^k(t) - \mathcal{P}_i^{k+1}(t)\|.$$  \hspace{1cm} (6.3)

Note that alternative transition costs $w$ can be defined that can capture, e.g., consumed energy or travel time. The problem that is addressed in this paper can be summarized as follows and illustrated in Figure 6.1.

**Problem 6.1.3.** Given dynamic task specifications $\mathcal{H}_{i_{\text{cur}}}(t)$ design teams $\mathcal{T}_m(t)$ and select respective communication points $v_j$, $j \in \mathcal{C}_m$ so that the robot paths $\mathcal{P}_i(t)$ for all $i \in \mathcal{N}$ satisfy: (i) the assigned tasks are accomplished, i.e., all robots $i$ go through all waypoints of $\mathcal{H}_{i_{\text{cur}}}(t)$ in the order they appear in $\mathcal{H}_{i_{\text{cur}}}(t)$; (ii) the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often; and (iii) the total cost function $\sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t))$ is minimized.

To solve Problem 6.1.3 we propose a distributed control framework that decouples the task planning and the intermittent communication problem; see also Figure 6.2. Specifically, first we design schedules (i.e., sequences) of communication events for all robots, independently of the assigned task, that ensure that the communication graph $\mathcal{G}_c(t)$ is connected over time infinitely often; see Section 6.2. These schedules are initialized given the initial teams $\mathcal{T}_m(t_0)$ and are updated locally and online as the robots design new teams; see Section 6.2. Second, we integrate the schedules of communication events with the dynamic tasks $\mathcal{H}_{i_{\text{cur}}}(t)$; see Section 6.3. Specifically, given the initial communication schedules we design initial paths $\mathcal{P}_i(t_0)$ for all $i \in \mathcal{N}$ using the initial task specifications $\mathcal{H}_{i_{\text{cur}}}(t_0)$, if they are available, and the initial user-specified communication points for the teams $\mathcal{T}_m(t_0)$ and we compute the time instants when the first communication event within every team should occur; see Section 6.3.1. The teams $\mathcal{T}_m(t)$ and the associated communication points, the time instants that the teams communicate, and the resulting paths $\mathcal{P}_i(t)$ are then updated...
Figure 6.1: Graphical illustration of the problem formulation 6.1.3. A network of $N = 3$ robots (colored dots) divided into $M = 3$ teams is depicted. The robot teams at time $t$ are designed as $T_1(t) = \{1, 2\}$, $T_2(t) = \{1, 3\}$, and $T_3(t) = \{3, 2\}$. The green polygons, the blue square, and the red star stand for the communication points in the sets $C_1$, $C_2$, and $C_3$, respectively. Figure 6.1(a) illustrates the sequences $H_{i \text{cur}}(t)$ and Figure 6.1(b) depicts the paths $P_i(t)$ that include the task waypoints of $H_{i \text{cur}}(t)$ and the communication points for all teams $T_m$, $m \in M_i$. The black rectangle stands for an obstacle.

online for all $t > t_0$ so that the assigned task is satisfied, the communication network $G_{c}(t)$ is connected over time infinitely often, and the total cost function $\sum_{i \in N} J(P_i(t))$ is minimized; see Sections 6.3.6-6.3.5.

6.2 Intermittent Connectivity Control

In this section we address the intermittent communication control problem ignoring the assigned task; see also Figure 6.2. In particular, in Section 6.2.1, we define time-dependent finite sequences $\text{sched}_i(t)$ of communication events (also called communication schedules) that determine the order in which the robots in every team $T_m(t)$ for $m \in M_i(t)$ should communicate with each other. In Section 6.2.2 we discuss how these schedules can be initialized in a decentralized way based on Algorithm 1, presented in Chapter 2. Next, in Section 6.2.3, we present how the robots can online and locally redesign their communication schedules, every time they decide to
update the set of teams they belong to. Finally, in Section 6.2.4, we define a policy that allows the robots to execute online these communication schedules so that the communication graph is intermittently connected.

6.2.1 Definition of Communication Schedules

In what follows, we define the communication schedules that determine the order in which the robots in every team $\mathcal{T}_m(t)$ for $m \in M_i(t)$ should communicate with each other.

**Definition 6.2.1** (Schedule of Communication Events). *The schedule of communication events of robot $i$ at time $t$, denoted by $\text{sched}_i(t)$, is a finite sequence defined*
as

\[ \text{sched}_i(t) = X, \ldots, X, \mathcal{M}^1_i(t), X, \ldots, X, \mathcal{M}^2_i(t), X, \ldots, X, \]

\[ \mathcal{M}^{|\mathcal{M}_i(t)|}_i(t), X, \ldots, X. \]  

(6.4)

In (6.4), \( \mathcal{M}_i^e(t) \) stands for the \( e \)-th element in the set of teams \( \mathcal{M}_i(t) \) and represents a communication event for team with index \( \mathcal{M}_i^e(t), e \in \{1, \ldots, |\mathcal{M}_i(t)|\} \). Also, the discrete states \( X \) indicate that there is no communication event for robot \( i \). The schedule \( \text{sched}_i(t) \) is updated only when robot \( i \) updates the set \( \mathcal{M}_i(t) \) of the teams it belongs to, which as discussed in Section 6.3, can happen at during a communication event. Thus, between two consecutive communication events for robot \( i \), the set \( \mathcal{M}_i(t) \) remains fixed and so does the schedule \( \text{sched}_i(t) \). The length of \( \text{sched}_i(t) \), denoted by \( L \), is the same for all robots and remains fixed for all \( t \geq t_0 \). The length \( L \) is defined in Section 6.2.2.

### 6.2.2 Initialization of Communication Schedules

In this section, we present how the robots initialize in a decentralized way the communication schedules \( \text{sched}_i(t_0) \) using the user-specified teams \( \mathcal{T}_m(t_0) \). This initialization phase requires that the robots are initially connected so that they can share information with each other.

Since every robot can be a member of more than one team, the first objective in designing the schedules \( \text{sched}_i(t_0) \) is that no teams that share any robots communicate at the same time, as this would require that the robots are present at more than one, possibly different, communication points simultaneously. We call these schedules conflict-free. The second objective in designing the schedules \( \text{sched}_i(t_0) \) is that their online execution as per Algorithm 14 is deadlock-free. Specifically, we say that there is a deadlock if there are robots \( i \in \mathcal{T}_m(t) \) that are waiting forever for robots \( j \in \mathcal{T}_m(t) \) to participate at the communication event of team \( \mathcal{T}_m \); see line 5 in Algorithm 14.
The detailed construction of conflict-free and deadlock-free communication schedules \( \text{sched}_i(t_0) \) can be found in Chapter 2. Specifically, the communication schedules \( \text{sched}_i(t_0) \) can be constructed exactly as the finite sequences \( s_i \) are constructed by Algorithm 1 in Chapter 2. The length of the constructed schedules is denoted by \( \ell \) and is common for all robots. Once all robots have initialized their communication schedules, two additional steps follow. First, the robots exchange their communication schedules so that every robot \( i \) can construct ordered sets \( \mathcal{O}_c \) that collect the communication events that exist in the \( c \)-th entry of the schedules of all robots, for all \( c \in \{1, \ldots, \ell\} \). Construction of these ordered sets requires as an input a user-defined sequence \( R = (i, \mathcal{T}_m(t_0)), (e, \mathcal{T}_n(t_0)), \ldots \) of length \( M \) that determines the order in which the communication events should appear in the sets \( \mathcal{O}_c \). The sequence \( R \) satisfies two requirements: (i) for each pair \( (i, \mathcal{T}_m(t_0)) \) it holds that \( i \in \mathcal{T}_m(t_0) \); and (ii) all teams \( \mathcal{T}_m(t_0) \) appear once in \( R \). To construct the sets \( \mathcal{O}_c \), subsequences \( R_c \subseteq R \) are constructed that collect all pairs \( (i, \mathcal{T}_m(t_0)) \in R \) that satisfy \( \text{sched}_i^c(t_0) = m \). Then, the sets \( \mathcal{O}_c \) are defined as follows.

**Definition 6.2.2 (Sets \( \mathcal{O}_c \)).** The ordered set \( \mathcal{O}_c \), for all \( c \in \{1, \ldots, \ell\} \), is defined as

\[
\mathcal{O}_c = \{ \text{sched}_{R_c(1)}^c(t_0), \ldots, \text{sched}_{R_c(|R_c|)}^c(t_0) \} \tag{6.5}
\]

In (6.5), \( R_c \) is a finite sequence that is obtained by deleting the teams from all pairs \( (i, \mathcal{T}_m(t_0)) \) of the sequence \( R \) and \( R_c(r) \) stands for the \( r \)-th robot in \( R_c \); see also Example 6.2.3. We assume that the sequence \( R \) is user-defined and known by all robots. Availability of the sets \( \mathcal{O}_c \) enables the robots to locally and online construct new conflict-free and deadlock-free communication schedules when they update the sets \( \mathcal{M}_i(t) \), even if they are not connected to each other; see Section 6.2.3. Second, using the constructed sets \( \mathcal{O}_c \), every robot \( i \) adds to \( \text{sched}_i(t_0) \) additional entries. Specifically, after the \( c \)-entry of the initial schedule \( \text{sched}_i(t_0) \), \( |\mathcal{O}_c| \) entries are added and filled with \( X \)'s, for all \( c \in \{1, \ldots, \ell\} \); see also Example 6.2.3. These new entries
are used in Section 6.2.3 to redesign online locally communication schedules \( \text{sched}_i(t) \) when the teams change. The length of all schedules \( \text{sched}_i(t_0) \) after the last two steps is \( L = \ell + \sum_{c=1}^\ell \mid O_c \mid \) as discussed in Section 6.2.1.

**Example 6.2.3** (Initialization of Communication Schedules). To illustrate the construction of the initial communication schedules, consider a network of \( N = 4 \) robots, where the teams of robots are initially designed as \( \mathcal{T}_1(t_0) = \{1, 2\}, \mathcal{T}_2(t_0) = \{3, 4\}, \mathcal{T}_3(t_0) = \{2, 3\}, \) and \( \mathcal{T}_4(t_0) = \{1, 4\} \). The schedules constructed as per Algorithm 1 presented in Chapter 2 are \( \text{sched}_1(t_0) = 1, 4, \text{sched}_2(t_0) = 1, 3, \text{sched}_3(t_0) = 2, 3, \text{sched}_4(t_0) = 2, 4 \) with length \( \ell = 2 \). Using these schedules and user-defined sequence of robots \( R = (1, \mathcal{T}_1(t_0)), (3, \mathcal{T}_2(t_0)), (3, \mathcal{T}_3(t_0)), (4, \mathcal{T}_4(t_0)) \), subsequences \( R_c \) are constructed as \( R_1 = (1, \mathcal{T}_1(t_0)), (3, \mathcal{T}_2(t_0)) \) and \( R_2 = (3, \mathcal{T}_3(t_0)), (4, \mathcal{T}_4(t_0)) \). Next, using the subsequences \( R_c \), the sets \( O_c \) are constructed as \( O_1 = \{ \text{sched}_1^1(t_0), \text{sched}_3^1(t_0) \} = \{1, 2\}, O_2 = \{ \text{sched}_2^2(t_0), \text{sched}_3^2(t_0) \} = \{4, 3\} \). Then, using the sets \( O_c \), the communication schedules are updated by adding \( X \)'s as follows

\[
\begin{bmatrix}
\text{sched}_1(t_0) \\
\text{sched}_2(t_0) \\
\text{sched}_3(t_0) \\
\text{sched}_4(t_0)
\end{bmatrix} =
\begin{bmatrix}
1, X, X, 4, X, X \\
1, X, X, 3, X, X \\
2, X, X, 3, X, X \\
2, X, X, 4, X, X
\end{bmatrix}.
\] (6.6)

**Remark 6.2.4** (Discrete states \( X \)). In the communication schedules \( \text{sched}_i(t_0) \), defined in Definition 6.2.1, the states \( X \) indicate that there is no communication event placed at an entry of \( \text{sched}_i(t_0) \) with index \( n_i \in \{1, \ldots, L\} \). These states are used to ensure that the communication event for a team \( \mathcal{T}_m(t_0) \) is placed at an entry of \( \text{sched}_i(t_0) \) with index \( n_i \) that is the same for all robots \( i \in \mathcal{T}_m(t_0) \); see also Example 6.2.3. Nevertheless, as it will be shown in Corollary 6.4.9, in Section 6.4, it is the order of communication events in \( \text{sched}_i(t) \) that is critical to ensure intermittent communication, and not the indices \( n_i \). This is due to the waiting actions of robots during the execution of \( \text{sched}_i(t) \); see line 5 in Algorithm 14.
6.2.3 Online Redesign of Communication Schedules

In this section, we discuss how the robots locally redesign their communication schedules, if they decide to join new teams or leave existing teams. Note that it is possible for a robot to leave a team without joining any new team, to join a new team without leaving any of the teams it already belongs to, or to join a new team and leave a team it already belongs to at the same time. Again, the goal is to obtain new communication schedules that are conflict-free and deadlock-free. This redesign phase occurs online and is communication-free, i.e., it does not require communication among any robots in the network.

Assume that at time $t$, robot $i$ updates the set $\mathcal{M}_i(t)$ of teams it belongs to and, therefore, it needs to redesign its communication schedule. Hereafter, we denote by $\text{sched}_i(t^+)$ the revised communication schedule of robot $i$ right after updating $\mathcal{M}_i(t)$. In what follows, we discuss how the communication schedules are redesigned if robot $i$ decides to (i) join a new team $\mathcal{T}_e(t)$, $e \notin \mathcal{M}_i(t)$ or (ii) leave an existing team $\mathcal{T}_e(t)$, $e \in \mathcal{M}_i(t)$; see also Algorithm 13. In both cases, we first initialize the revised communication schedules as $\text{sched}_i(t^+) = \text{sched}_i(t)$ (line 1, Alg. 13).

Assume that robot $i$ decides to join a new team $\mathcal{T}_e(t)$, $e \notin \mathcal{M}_i(t)$. Then, to construct the respective revised communication schedules, we need to define the one-to-one functions

$$g_e(\mathcal{O}_c(k)) = \alpha(c) + k,$$  \hspace{1cm} (6.7)

for all $c \in 1, ..., \ell$, where $\mathcal{O}_c(k)$ is the $k$-th element of $\mathcal{O}_c$ and $\alpha(c)$ is defined as

$$\alpha(c) = \sum_{p=1}^{c-1} (|\mathcal{O}_p| + 1).$$  \hspace{1cm} (6.8)

\footnote{Notice that the indices $\alpha(c)$ point to entries of $\text{sched}_i(t)$ where communication events have been placed during the construction of $\text{sched}_i(t_0)$. In other words, the indices in the interval $[\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|]$ point to the entries that are added to $\text{sched}_i(t_0)$ after its initialization and are initially filled with $X$’s.}
Note that the functions \( g_c \) map the communication event for a team with index \( e \in \mathcal{O}_c \) to a unique entry of \( \text{sched}_i(t^+) \) with index that belongs to \( [\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|] \subseteq \mathbb{N}_+ \). Using the functions \( g_c \), robot \( i \) can locally construct its revised communication schedule \( \text{sched}_i(t^+) \) based on the following two rules that hold for all \( c \in \{1, \ldots, \ell\} \).

- **First rule:** Assume that \( e \in \mathcal{O}_c \) is such that \( e \notin \mathcal{M}_i(t) \). Then, the communication event for the team with index \( e \) is placed at the entry of \( \text{sched}_i(t^+) \) with index \( g_c(e) \) (line 3, Alg. 13).

- **Second rule:** If there exists a communication event for a team with index \( b \in \mathcal{O}_c \cap \mathcal{M}_i(t) \) then this communication event is placed in \( \text{sched}_i(t^+) \) at the entry with index \( g_c(b) \) and the entry in which \( b \) was previously placed is filled with an \( X \) (lines 4-7, Alg. 13).

The first rule ensures that the schedules \( \text{sched}_i(t^+) \) are *conflict-free*, since the entry in \( \text{sched}_i(t^+) \) with index \( g_c(e) \) is reserved for the communication event \( e \) and no other communication event can be placed at this entry. The second rule ensures that the schedules \( \text{sched}_i(t^+) \) are *deadlock-free*, as it will be shown in Section 6.4. The online redesign of the schedules \( \text{sched}_i(t) \) and the importance of the second rule are also explained in Examples 6.2.6 and 6.2.7, respectively.

Next, assume that robot \( i \) decides to leave a team \( T_e(t), e \in \mathcal{M}_i(t) \). Then, the revised schedule \( \text{sched}_i(t^+) \) is constructed by replacing the event \( e \) in \( \text{sched}_i(t) \) with an \( X \) (lines 8-9, Alg. 13). In line 9 of Algorithm 13, the index \( n_e^i \) points to the entry of \( \text{sched}_i(t^+) \), where the communication event for team \( T_e \) has been placed.

**Remark 6.2.5** (Algorithm 13). *Recall from Section 6.2.2 that at time \( t_0 \) the communication event for a team \( T_h(t_0), h \in \mathcal{M}_i(t_0) \cap \mathcal{O}_c \) is placed at an entry of \( \text{sched}_i(t_0) \) with index \( \alpha(c), c \in \{1, \ldots, \ell\} \). If at time \( t > t_0 \), robot \( i \) decides to join another team \( T_e(t) \), where \( e, h \in \mathcal{O}_c \), then both \( e \) and \( h \) will be placed at the entries in \( \text{sched}_i(t) \)
Algorithm 13: Online Redesign of Communication Schedules

Input: \( e \in \mathcal{O}_c, g_c, \text{sched}_i(t) \)

Output: \( \text{sched}_i(t^+) \)

1. \( \text{sched}_i(t^+) = \text{sched}_i(t) \);
2. if Team \( \mathcal{T}_c(t), e \notin \mathcal{M}_i(t) \) joined then
   3. \( \text{sched}^{c_i(e)}_i(t^+) = e \) (First rule);
   4. if \( \text{sched}^{\alpha_c(c)}_i(t^+) \neq X \) then
      5. \( b = \text{sched}^{\alpha_c(c)}_i(t^+) \);
      6. \( \text{sched}^{\alpha_c(c)}_i(t^+) = X \);
      7. \( \text{sched}^{c_i(b)}_i(t^+) = b \) (Second rule);
3. if Team \( \mathcal{T}_c(t), e \in \mathcal{M}_i(t) \) left then
   4. \( \text{sched}^{c_i(e)}_i(t^+) = X \)

with indices \( g_c(e) \) and \( g_c(h) \), \( g_c(e) \neq g_c(h) \), by construction of Algorithm 13. Also, by construction of Algorithm 13, neither of the events \( h \) and \( e \) are ever going to be placed back at the entry with index \( \alpha_c(c) \). In fact, there will be an \( X \) at the entry of \( \text{sched}_i(t') \) with index \( \alpha_c(c) \) for all \( t' > t \). Finally, notice since all robots \( i \in \mathcal{N}_c \) construct the same sets \( \mathcal{O}_c \), the index \( g_c(e), e \in \mathcal{M}_i \), is the same for all robots \( i \in \mathcal{N}_c \).

Example 6.2.6 (Construction of \( \text{sched}_i(t^+) \)). Consider the network of 4 robots of Example 6.2.3 with the following initial communication schedules

\[
\begin{bmatrix}
\text{sched}_1(t_0) \\
\text{sched}_2(t_0) \\
\text{sched}_3(t_0) \\
\text{sched}_4(t_0)
\end{bmatrix}
= \begin{bmatrix}
1, X, X, 4, X, X \\
1, X, X, 3, X, X \\
2, X, X, 3, X, X \\
2, X, X, 4, X, X
\end{bmatrix}.
\]

(6.9)

Recall from Example 6.2.3 that \( \mathcal{O}_1 = \{1, 2\} \) and \( \mathcal{O}_2 = \{4, 3\} \). Given the sets \( \mathcal{O}_1 \) and \( \mathcal{O}_2 \), the functions \( g_1 \) and \( g_2 \) are defined as \( g_1(1) = 2, g_1(2) = 3, g_2(4) = 5, \) and \( g_2(3) = 6 \). Assume that the first time that robots update their teams and, as a result their schedules \( \text{sched}_i(t) \), is at the time instant \( t \). Thus, the schedules in (6.9) are fixed from \( t_0 \) until \( t \). At time \( t \), assume that robot 2 decides to join team \( \mathcal{T}_2(t) = \{3, 4\} \), \( 2 \notin \mathcal{M}_2(t) \) and robot 3 decides to leave team \( \mathcal{T}_2(t), 2 \in \mathcal{M}_1(t) \). Notice that for team \( \mathcal{T}_2(t) \) it holds that \( 2 \in \mathcal{O}_1 \). Then applying the first two rules to \( \text{sched}_2 \)
and replacing in \textit{sched}₃ the first entry that satisfies \textit{sched}₃(t) = 2 with \(X\), we get the following revised communication schedules

\[
\begin{bmatrix}
\text{sched}_1(t^+) \\
\text{sched}_2(t^+) \\
\text{sched}_3(t^+) \\
\text{sched}_4(t^+)
\end{bmatrix} = \begin{bmatrix}
1, & X, & X, & 4, & X, & X \\
X, & 1, & 2, & 3, & X, & X \\
X, & X, & X, & 3, & X, & X \\
2, & X, & X, & 4, & X, & X
\end{bmatrix}.
\] (6.10)

Note that at time \(t^+\), we have that \(\mathcal{T}_2(t^+) = \{2, 4\}\) and that robots 1, 4 are not aware of the change in team \(\mathcal{T}_2(t^+)\). Observe also in (6.10) that robots 1, 4 did not update their schedules at time \(t\), since they did not update the sets of teams they belong to.

**Example 6.2.7** (Second rule of Algorithm 13). Continuing the previous example, assume that at a subsequent time instant \(t' \geq t\) robot 4 decides to join team \(\mathcal{T}_1(t') = \{1, 2\}\) and robot 1 decides to leave \(\mathcal{T}_1(t')\). Then, applying again the first two rules to \(\text{sched}_4(t')\) and replacing in \(\text{sched}_1(t')\) the first entry that satisfies \(\text{sched}_1(t') = 1\) with \(X\), we get the following revised communication schedules.

\[
\begin{bmatrix}
\text{sched}_1(t'^+) \\
\text{sched}_2(t'^+) \\
\text{sched}_3(t'^+) \\
\text{sched}_4(t'^+)
\end{bmatrix} = \begin{bmatrix}
X, & X, & X, & 4, & X, & X \\
X, & 1, & 2, & 3, & X, & X \\
X, & X, & X, & 3, & X, & X \\
X, & 1, & 2, & 4, & X, & X
\end{bmatrix}.
\] (6.11)

Notice that without applying the second rule in the Examples 6.2.6-6.2.7, the resulting communication schedules at time \(t'^+\) would be

\[
\begin{bmatrix}
\text{sched}_1(t'^+) \\
\text{sched}_2(t'^+) \\
\text{sched}_3(t'^+) \\
\text{sched}_4(t'^+)
\end{bmatrix} = \begin{bmatrix}
X, & X, & X, & 4, & X, & X \\
1, & X, & 2, & 3, & X, & X \\
X, & X, & X, & 3, & X, & X \\
2, & 1, & X, & 4, & X, & X
\end{bmatrix}.
\] (6.12)

Observe that communication according to the schedules (6.12) would create a deadlock. The reason is that in \(\text{sched}_2\) the communication event of team \(\mathcal{T}_1\) should occur before the communication event of team \(\mathcal{T}_2\), while in \(\text{sched}_4\) the communication event of team \(\mathcal{T}_1\) should occur after the communication event of team \(\mathcal{T}_2\). This situation is avoided in the schedules (6.11).
Algorithm 14: Online Execution of $\text{sched}_i(t)$, $i \in \mathcal{T}_m(t)$

1  $z_i = 0$;
2  $n_i = 1$;
3  for $t = 0 : \infty$ do
4    if $\text{sched}^{n_i}_i(t) = m$ then
5      Wait until $\text{sched}^{n_j}_j(t) = m$, for all $j \in \mathcal{T}_m(t)$;
6      Construct set $\mathcal{M}_i(t^+)$ and schedule $\text{sched}_i(t^+)$;
7      $z_i = z_i + 1$;
8      $n_i = n_i^{m}(t^+) + 1$;
9    else if $\text{sched}^{n_i}_i(t) = X$ then
10      $n_i = n_i + 1$;
11    if $n_i > L$ then
12      $n_i = 1$;
13    Next event is $\text{sched}^{n_i}_i(t)$;

6.2.4 Online Execution of Communication Schedules

The execution of $\text{sched}_i(t)$ is described in Algorithm 14. In Algorithm 14, we denote by $\text{sched}^{n_i}_i(t)$ the entry in the finite sequence $\text{sched}_i(t)$ with index $n_i \in \{1, \ldots, L\}$. Moreover, in Algorithm 14, the index $n_i$ is updated so that it points to the next communication event in $\text{sched}_i(t)$ that robot $i$ needs to participate and is initialized as $n_i = 1$, for all robots $i \in \mathcal{N}$ [line 2, Alg. 14]. If $\text{sched}^{n_i}_i(t) = m$, $m \in \mathcal{M}_i(t)$, then robot $i$ needs to communicate with robots in team $\mathcal{T}_m(t)$ and waits until $\text{sched}^{n_j}_j(t) = m$, for all $j \in \mathcal{T}_m(t)$ [lines 4-5, Alg. 14]. Once this happens, the robots in team $\mathcal{T}_m(t)$ communicate, construct the sets $\mathcal{M}_i(t^+)$ and the respective schedules $\text{sched}_i(t^+)$ that they will have right after this update [lines 4-6, Alg. 14]; see also Section 6.3. Next, robot $i$ updates the index $n_i$ so that it points to the next event in the schedule $\text{sched}_i(t^+)$ that it needs to participate. Particularly, the next event for robot $i$ is $\text{sched}^{n_i}_i(t^+)$, where $n_i = n_i^{m}(t^+) + 1$ if $n_i^{m}(t^+) + 1 \leq L$ [line 8, Alg. 14], or $n_i = 1$ if $n_i^{m}(t^+) + 1 > L$ [lines 11-12, Alg. 14], where the index $n_i^{m}(t^+)$ points to the entry in the schedule $\text{sched}_i(t^+)$ where the communication event for team $\mathcal{T}_m(t)$ has been.

Note that the indices $n_i$ are updated asynchronously across the network of robots.
placed. In case $\text{sched}_i^n(t) = X$, then robot $i$ does not update the set $\mathcal{M}_i(t)$ or the schedule $\text{sched}_i(t)$ [line 9, Alg. 14] and updates the index $n_i$ as $n_i = n_i + 1$ if $n_i + 1 \leq L$ [line 10, Alg. 14], or $n_i = 1$ if $n_i + 1 > L$ [lines 11-12, Alg. 14]. Between consecutive communication events, the robots can move free of communication constraints in the workspace to accomplish their tasks. Finally, in Algorithm 14, $z_i \in \mathbb{N}_+$ counts the number of communication events that robot $i$ has participated [line 7, Alg. 14], which is initialized as $z_i = 0$ [line 1, Alg. 14]. The index $z_i$ is only used in the proof of Theorem 6.4.8 in Section 6.4.

**Example 6.2.8 (Execution of $\text{sched}_i(t)$).** Consider the following schedule of robot 2 at time $t$ $\text{sched}_2(t) = 1, X, X, 3, X, X$. Assume that at time $t$, it holds that $n_2 = 1$ and, therefore, $\text{sched}_2^{n_2}(t) = 1$, i.e., robot 2 needs to communicate with robots in team $\mathcal{T}_1(t)$. Assume that the robots in team $\mathcal{T}_1(t)$ communicate, robot 2 updates its schedule, and its new schedule at time $t^+$, i.e., right after robot 2 leaves the communication point $v_j$ is $\text{sched}_2(t^+) = X, 1, 2, 3, X, X$. Observe that in the new schedule we have that $n_2^{1}(t^+) = 2$. Therefore, according to line 8 in Algorithm 14, the next communication event for robot 2 is $\text{sched}_2^{n_2^{1}(t^+)+1} = 2$, i.e., robot 2 needs to communicate with robots in team $\mathcal{T}_2(t^+)$. 

### 6.3 Integrated Path Planning and Intermittent Communication Control

In this section, we synthesize paths $\mathcal{P}_i(t)$ that satisfy the assigned tasks $\mathcal{H}_i^{\text{cur}}(t)$, the intermittent connectivity requirement, and minimize the total cost $\sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t))$. To achieve this, we design teams and select communication points that are introduced in the dynamic tasks $\mathcal{H}_i^{\text{cur}}(t)$ so that the total cost $\sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t))$ is minimized. In Section 6.3.1, we present the distributed initialization of the paths $\mathcal{P}_i(t_0)$. In Section 6.3.2, we discuss when the robots can update their paths by adding task waypoints
or communication points. Next, in Sections 6.3.3-6.3.5, we discuss how the robots select the communication points for their teams, the time instants when they should communicate, who they should talk to, and how these decisions are integrated with the paths $\mathcal{P}_i(t)$. Finally, in Section 6.3.6, we discuss how the paths $\mathcal{P}_i(t)$ are executed online.

6.3.1 Initialization of Paths

Using the schedules $\text{sched}_i(t_0)$, we design the initial paths $\mathcal{P}_i(t_0)$ that include (i) all waypoints in $\mathcal{H}_i^{\text{cur}}(t_0)$ in the order they appear in $\mathcal{H}_i^{\text{cur}}(t_0)$, and (ii) the user-specified communication points $v_j$, $j \in \mathcal{C}_m$, for all teams $\mathcal{T}_m(t_0)$, $m \in \mathcal{M}_i(t_0)$. Thus, the number of waypoints in $\mathcal{P}_i(t_0)$ is $K_i(t_0) = |\mathcal{H}_i^{\text{cur}}(t_0)| + |\mathcal{M}_i(t_0)|$, where $|\mathcal{H}_i^{\text{cur}}(t_0)|$ stands for the number of waypoints in $\mathcal{H}_i^{\text{cur}}(t_0)$. This initialization phase does not require the robots to be connected with each other; see also Figure 6.2.

Specifically, first the paths $\mathcal{P}_i(t_0)$ are initialized as $\mathcal{P}_i(t_0) = \mathcal{H}_i^{\text{cur}}(t_0)$. If the task specifications $\mathcal{H}_i^{\text{cur}}(t_0)$ are not available, then the paths $\mathcal{P}_i(t_0)$ are initialized as $\mathcal{P}_i(t_0) = \emptyset$. Then, the paths $\mathcal{P}_i(t_0)$ are updated by incorporating into them the user-specified communication points $v_j$, $j \in \mathcal{C}_m$, for all teams $\mathcal{T}_m(t_0)$, $m \in \mathcal{M}_i(t_0)$. The index $k^m_i \in \{1, \ldots, K_i(t_0)\}$ of the entry in $\mathcal{P}_i(t_0)$ where the communication point $v_j$, $j \in \mathcal{C}_m$ for team $\mathcal{T}_m(t_0)$ will be placed can be selected either arbitrarily or optimally so that the cost function $J(\mathcal{P}_i(t_0))$ is minimized. The only requirement is that the communication points are introduced in $\mathcal{P}_i(t_0)$ in the order the respective communication events appear in $\text{sched}_i(t_0)$, for all $i \in \mathcal{N}$. In this way, we ensure that the communication events during the execution of the paths $\mathcal{P}_i(t_0)$ will occur in the order determined by the $\text{sched}_i(t_0)$. Note that during the initialization phase, the $X$’s that appear in $\text{sched}_i(t_0)$ are ignored and are not introduced in the paths $\mathcal{P}_i(t_0)$.
6.3.2 **Online Construction of Paths**

At any time $t$ every robot $i$ can update the current task $\mathcal{H}_i^{\text{cur}}(t)$ by appending additional waypoints from $\mathcal{H}_i$, as discussed in Section 5.1. The additional waypoints are appended to the paths $\mathcal{P}_i(t)$, as well. Moreover, when the robots $i \in \mathcal{T}_m(t)$ meet at the respective communication point that appears in their paths $\mathcal{P}_i(t)$, they communicate and coordinate to update their corresponding paths $\mathcal{P}_i(t^+)$, the next communication point for team $\mathcal{T}_m(t^+)$, the set of teams $\mathcal{M}_i(t^+)$, and the respective communication schedules $\text{sched}_i(t^+)$ that they will have at the time instant $t^+$, i.e., right after leaving this communication point.

Note that this coordination process can be formulated as a centralized nonlinear optimal control problem with the objective to minimize the total cost $\sum_{i \in \mathcal{T}_m(t)} J(\mathcal{P}_i(t^+))$ and constraints that ensure that the robots $i \in \mathcal{T}_m(t)$ (i) will select a common next communication point for $\mathcal{T}_m(t^+)$; (ii) will compute a common time instant when they should communicate again so that the resulting waiting time is zero; (iii) will design new sets $\mathcal{M}_i(t^+)$ that satisfy $|\mathcal{M}_i(t^+)| \geq 1$ while preserving connectivity of the team membership graph $\mathcal{G}_T(t^+)$, as required in Section 5.1; and (iv) will incorporate the communication points for all teams $e \in \mathcal{M}_i(t^+)$ as per the schedule $\text{sched}_i(t^+)$. Note that solving this optimal control problem requires global information due to the constraint (iii), which may not available at time $t$ in an intermittent communication setup. To address this problem, we propose a distributed method that involves three steps and is summarized in Algorithm 15; see also Figure 6.2. First, the robots select the next communication point for team $\mathcal{T}_m(t)$. Second, given the selected communication point, they compute the time instant when they should communicate there, so that all robots in $\mathcal{T}_m(t)$ can arrive simultaneously at the selected communication point. Third, given the selected next communication point and the respective next meeting time instant, the robots in $\mathcal{T}_m(t)$ coordinate with each other to select, among
Algorithm 15: Coordination during a communication event for team $T_m(t)$ at time $t$

**Input:** Sets $C_e$, $\forall e \in M$, $P_i(t)$, $M_i(t)$, $sched_i(t)$, $Csched_i(t)$, for all $i \in T_m(t)$

**Output:** $P_i(t^+)$, $M_i(t^+)$, $sched_i(t^+)$, for all $i \in T_m(t)$

1. Select next communication point and incorporate it into $P_i(t^+)$, for all $i \in T_m(t)$ as per the solution of (6.13) (Section 6.3.3);
2. Compute next meeting time instant $t^m$ for team $T_m$ (Algorithm 16, Section 6.3.4);
3. Design sets $M_i(t^+)$ and construct respective schedules $sched_i(t^+)$ and paths $P_i(t^+)$, for all $i \in T_m(t)$ (Algorithm 17, Sections 6.3.5);

the teams $e \in \cup_{i \in T_m(t)} M_i(t)$, which teams they should join or leave in order to further decrease the total cost $\sum_{i \in T_m(t)} J(P_i(t^+))$ while ensuring that every robot belongs to at least one team, i.e., $|M_i(t^+)| \geq 1$, and that the team membership graph $G_T(t^+)$ remains connected. This coordination process is summarized in Algorithm 15 and discussed in detail in the following sections.

6.3.3 Selection of Next Communication Point: Where?

To select the next communication point $v_j$, $j \in C_m$ for team $T_m(t)$ and incorporate it into $P_i(t)$ giving rise to the paths $P_i(t^+)$, the robots $i \in T_m(t)$ solve the following integer program (line 1, Alg. 15).

\[
\begin{align*}
\text{minimize} & \quad \sum_{v_j \in C_m, \{k_i^m\}_{i \in T_m(t)}} J(P_i(t^+)) \\
\text{subject to} & \quad P_i^{k_i^m}(t^+) = v_j, \forall i \in T_m(t), \\
& \quad k_i^m > k_i^{LC}(t), \forall i \in T_m(t), \\
& \quad k_i^m \geq \min(k_i^{task}(t^+) + 1, K_i(t^+)) \\
& \quad k_i^m \leq k_i^{before}(t), \forall i \in T_m(t), \text{where} \\
& \quad K_i(t^+) \geq k_i^{before}(t) \geq \min(k_i^{task}(t^+) + 1, K_i(t^+)).
\end{align*}
\]
Figure 6.3: Graphical illustration of optimization problem (6.13). Robots 1 and 2 (black dots) of team $T_1(t)$ meet at the selected communication point $P_1(t) = P_3(t)$ (green rhombus) and coordinate to select the next communication point for team $T_1(t^+)$. Red and blue squares stand for the waypoints that robots 1 and 2 have to visit to accomplish their tasks, respectively. The communication points for teams $T_2(t)$ and $T_3(t)$ are represented by stars. The resulting paths $P_1(t)$ and $P_2(t^+)$ comprise the red and blue, both solid and dashed, lines. The gray line stands for an edge in the path $P_1(t)$ that does not exist in the path $P_1(t^+)$ due to the introduction of the communication point for team $T_1(t^+)$. The parameters $k_{i}^{\text{before}}(t)$ are selected as $k_{1}^{\text{before}} = 8$ and $k_{2}^{\text{before}} = 6$, while $k_{1}^{\text{LC}}(t) = 6$, $k_{2}^{\text{LC}}(t) = 5$, $k_{1}^{\text{task}}(t) = k_{2}^{\text{task}}(t) = 2$, $K_{1}(t^+) = 10$, and $K_{2}(t^+) = 10$. The schedules of robots 1 and 2 are determined in Example 6.2.3. Observe that the communication points appear in $P_i(t^+)$ as determined by $\text{sched}_i(t^+)$, for both robots.

In the optimization problem (6.13), the paths $P_i(t^+)$ are initialized as $P_i(t^+) = P_i(t)$. In (6.13a), $J(P_i(t^+))$ stands for the cost of the path $P_i(t^+)$, defined in (6.2), and $k_{i}^{m}$ represents the index of the entry in $P_i(t^+)$ where the selected communication point $v_j, j \in \mathcal{C}_m$ will be placed, i.e., $P_i^{k_{i}^{m}}(t^+) = v_j$. According to the objective function (6.13a), the communication point $v_j$ is selected so that the sum of the costs of $P_i(t^+)$ for all robots $i \in T_m(t)$ is minimized. The first constraint (6.13b) requires that all robots $i \in T_m(t)$ will select the same communication point $v_j, j \in \mathcal{C}_m$ for the
next communication event of team $T_m(t)$ and incorporate it into the entry of $P_i(t^+)$ with index $k_i^m$. The second constraint (6.13c) ensures that all communication points for every team $T_e(t^+)$, $e \in \mathcal{M}_i(t)$, appear in $P_i(t^+)$ in the order that the indices of teams $e \in \mathcal{M}_i(t)$ appear in $\text{sched}_i(t)$. In particular, in the second constraint, the index $k_{iC}^l(t)$ points to the entry in $P_i(t)$ where the last communication point has been introduced. This constraint requires that robot $i$ will participate at the next communication event for team $T_m(t)$ only after it has visited all other communication points that appear in $P_i(t)$, for all robots $i \in T_m(t)$. This combined with the fact that the communication points appear in the paths $P_i(t)$ in the order determined by $\text{sched}_i(t)$, for all $i \in \mathcal{N}$ entails that the communication points are introduced into the paths $P_i(t^+)$ in an order determined by $\text{sched}_i(t)$, as well, for all $i \in \mathcal{N}$; see also Figure 6.3. As discussed in Remark 6.2.4, and as it will be shown in Proposition 6.4.5, this constraint ensures that the network never reaches a deadlock configuration and guarantees intermittent communication. Notice that the discrete states $X$ that appear in the schedules $\text{sched}_i(t)$ are ignored and are not introduced in the paths $P_i(t^+)$. The third constraint requires robot $i$ to visit at least one task waypoint before communicating again with robots in team $T_m(t)$; see also Figure 6.3. As it will be shown in Theorem 6.4.10, this ensures that the assigned task will be accomplished, i.e., that every robot $i$ will eventually go through all task waypoints that appear in $H_{i\text{cur}}^c(t)$, for all $t \geq t_0$. The last constraint requires that $k_i^m \leq k_i^\text{before}(t)$, where $k_i^\text{before}(t)$ is an index of entries in $P_i(t)$. The index $k_i^\text{before}(t)$ is selected so that the inequality $K_i(t^+) \geq k_i^\text{before}(t) \geq \min(k_i^\text{task}(t^+), K_i(t^+))$ is satisfied to ensure feasibility of (6.13). Notice that without this constraint, the optimization problem (6.13) can become computational expensive to solve for large teams $T_m(t)$ and for long paths $P_i(t)$. Finally, notice that the indices $k_i^m$ are not required to be the same

---

3 Note that we refer to the teams and the communication schedules at $t$ and not $t^+$, since they have not been updated yet.
for all robots in team $\mathcal{T}_m(t)$. As soon as the next communication point for team $\mathcal{T}_m(t)$ has been selected, the paths $\mathcal{P}_i(t^+)$ are constructed as per the solution of (6.13).

6.3.4 Selection of Next Meeting Time Instant: When?

In Section 5.1 we assumed that communication between robots in team $\mathcal{T}_m(t)$ happens only when all robots in that team are simultaneously present a common communication location $v_j$, $j \in \mathcal{C}_m$. Nevertheless, this control policy can lead to large waiting times which can be undesirable in case of time-critical missions, or it can even be infeasible if the robots cannot stay stationary, e.g., due to their dynamics. To avoid waiting delays, once the robots $i \in \mathcal{T}_m(t)$ select their next communication point, they also compute the time instant $t^m$ at which they will communicate again, so that the waiting time at that next communication point is zero (line 2, Alg.15).

The computation of the time instant $t^m$, called also 
 meeting time instant, is described in Algorithm 16. First, every robot $i \in \mathcal{T}_m(t)$ computes the last time instant $L_i \in \mathbb{R}_+$ that it needs to participate in a communication event during the execution of the path $\mathcal{P}_i(t)$, i.e., $L_i = \max_{e \in M_i(t)}(t^e)$ (line 2, Alg.16). This communication event takes place at the communication point $\mathcal{P}_i^{k_i^{LC}(t)}(t) = \mathcal{P}_i^{k_i^{LC}(t)}(t^+)$, where $k_i^{LC}(t)$ was defined in Section 6.3.3. Second, given the next communication point $v_j$, $j \in \mathcal{C}_m$, for team $\mathcal{T}_m(t)$, determined by the solution of (6.13), every robot $i \in \mathcal{T}_m(t)$ computes the minimum time required to travel from the location $\mathcal{P}_i^{k_i^{LC}(t)}(t)$ to $v_j = \mathcal{P}_i^{\text{index}^m}(t^+)$, denoted by $t_i^{tr}$ (line 3, Alg.16). Then, any time instant $t^m \geq \max_{i \in \mathcal{T}_m(t)}(L_i + t_i^{tr})$, is a feasible time instant.\footnote{Note that here we assume that if robot $i$ can arrive at a location $v_j$ at time instant $L_i + t_i^{tr}$, then it can also arrive at $v_j$ at $t^m \geq L_i + t_i^{tr}$ which is a reasonable assumption as long as the the robot velocities are not bounded below or have sufficiently small lower bounds.} Here, feasibility of $t^m$ means that there exists a controller which given the robot dynamics can drive robot $i$ from the communication point $\mathcal{P}_i^{k_i^{LC}(t)}(t^+)$ to $v_j$ within $t^m - L_i$ time units, for all robots $i \in \mathcal{T}_m(t)$. Design of such a control input for arbitrary robot dynamics is out of the scope of this paper. In this
Algorithm 16: Computation of meeting time instant $t^m$

**Input:** Paths $P_i(t^+)$, robot dynamics $\dot{x}_i(t) = f_i(x_i(t), u_i(t))$, for all robots $i \in T_m$, $v_j \in C_m$  

**Output:** Meeting time instant $t^m$

1. for $i \in T_m$ do
   2. Compute time at which the latest communication event for robot $i$ will occur: $L_i = \max_{e \in M_i(t)}(t^e)$;  
   3. Given dynamics $\dot{x}_i(t) = f_i(x_i(t), u_i(t))$ and the next communication point $v_j$, $j \in T_m$, compute the minimum required travel time, denoted by $t_i^{tr}$, from $P_i^{k_i}(t^+)$ to $v_j$;
   4. Next meeting time instant for team $T_m$: $t^m = \max_{i \in T_m}(L_i + t_i^{tr})$;

work, we select $t^m = \max_{i \in T_m}(L_i + t_i^{tr})$ (line 4, Alg. 16).

**Remark 6.3.1** (Initial Meeting Time Instants). Given the initial paths $P_i(t_0)$, feasible initial meeting time instants $t^m$ can be designed if every robot $i \in N$ runs Algorithm 16 for each team $T_m$, $m \in M_i(t_0)$ in the order determined by $\text{sched}_i(t_0)$.

Notice that all robots in team $T_m(t_0)$ will run Algorithm 16 for team $T_m(t_0)$ at the same time, by construction of $\text{sched}_i(t_0)$, and will collectively compute $t^m$. In this initialization process, (i) in line 2, we set $t^e = t_0$ for all $t^e$, $e \in M_i(t_0)$, that have not been computed at previous iterations of Algorithm 16, and (ii) in line 3, we compute the travel time from the communication point that appears right before the communication point of team $T_m(t_0)$ in $P_i(t_0)$ (or $x_i(t_0)$ if this communication point is not defined) to the communication point of $T_m(t_0)$. We show via simulations that even if the initial meeting time instants are selected arbitrarily and, as a result, they are not necessarily feasible, resulting in non-zero waiting times, the waiting time will eventually become zero for all teams $T_m(t_0)$, $m \in M$. By construction of Algorithm 16, if the initial meeting time instants are feasible, then the waiting time will always be zero for all teams $m \in M$.  

153
6.3.5 Update of Teams: Who talks to who?

Once the robots $i \in \mathcal{T}_m(t)$ decide where and when they will communicate next, they coordinate to construct new sets $\mathcal{M}_i(t^+)$ of the teams that they will belong to after they leave the current communication point and the respective new communication schedules $\text{sched}_i(t^+)$ (line 3, Alg. 15). The sets $\mathcal{M}_i(t^+)$ are constructed iteratively as described in Algorithm 17.

First, every robot $i \in \mathcal{T}_m(t)$ initializes the sets $\mathcal{M}_i(t^+)$ and the corresponding communication schedules as $\mathcal{M}_i(t^+) = \mathcal{M}_i(t)$ and $\text{sched}_i(t^+) = \text{sched}_i(t)$, respectively (line 2, Alg. 17). Then, robots $i \in \mathcal{T}_m(t)$ check if leaving a team $\mathcal{T}_e(t)$ and getting replaced in $\mathcal{T}_e(t)$ by robot $j \in \mathcal{T}_m(t) \setminus \{i\}$ decreases the cost $\sum_{r \in \mathcal{T}_m(t)} J(P_r(t^+))$. This is repeated sequentially for all teams $\mathcal{T}_e(t), e \in \mathcal{M}_i(t) \setminus \{m\}$ (line 4, Alg. 17). This process is also executed sequentially across all robots $i \in \mathcal{T}_m(t)$ (line 3, Alg. 17). In particular, the following three steps are required to determine if robot $i$ should leave team $\mathcal{T}_e(t), e \in \mathcal{M}_i(t) \setminus \{m\}$ (lines 5-22, Alg. 17).

**Leaving team $\mathcal{T}_e(t), e \in \mathcal{M}_i(t)$**

First, robot $i$ computes the cost of traveling towards the communication point of team $\mathcal{T}_e(t)$ (line 5, Alg. 17). To compute this cost, robot $i$ constructs the paths

$$\mathcal{Q}^i_e = [P_i^{k_i^e - 1}(t^+), P_i^{k_i^e}(t^+), P_i^{k_i^e + 1}(t^+)] \subseteq \mathcal{P}_i(t^+),$$

(6.14)

and

$$\overline{\mathcal{Q}}^e_i = [P_i^{k_i^e - 1}(t^+), P_i^{k_i^e}(t^+)] \subseteq \mathcal{Q}^e_i,$$

(6.15)

where recall that $k_i^e$ stands for the index of the entry in the path $\mathcal{P}_i(t^+)$ where the communication point for $\mathcal{T}_e(t)$ has been placed.\(^5\) For instance, in Figure 6.3, the red dashed lines and the gray solid line stand for the paths $\mathcal{Q}^1_i$ and $\overline{\mathcal{Q}}^1_i$, respectively.

\(^5\) In case $k_i^e = K_i(t^+)$, then the paths $\mathcal{Q}^e_i$ and $\overline{\mathcal{Q}}^e_i$ are defined as $\mathcal{Q}^e_i = [P_i^{k_i^e - 1}(t^+), P_i^{k_i^e}(t^+)]$ and $\overline{\mathcal{Q}}^e_i = \emptyset$. 

154
Given $Q_i^e$ and $Q_i^e$, the additional distance that robot $i$ has to travel to communicate with robots in team $T_e(t)$ is (line 5, Alg. 17)

$$J_i^e = J(Q_i^e) - J(Q_i^e) \geq 0,$$

where the cost function $J$ is defined in (6.2). Next, robot $i$ constructs the set of teams $\mathcal{M}_i$, the communication schedule $\text{sched}_i$, and the path $\mathcal{P}_i$ that it will have at time $t^+$ in case it leaves team $T_e(t)$ (line 6, Alg. 17). The set $\mathcal{M}_i$ is constructed as $\mathcal{M}_i = M_i(t^+) \{ e \}$. The schedule $\text{sched}_i$ is constructed as discussed in Section 6.2.3. The path $\mathcal{P}_i$ is constructed by deleting the communication point of team $T_e(t)$ from $P_i(t^+)$, i.e., the entry $P_i^{k+}(t^+)$. Recall that before updating the sets of teams, it holds that the communication points in $P_i(t^+)$ appear in the order determined by $\text{sched}_i(t^+)$; see Section 6.3.3. Therefore, the same also holds for $\mathcal{P}_i$ and $\text{sched}_i$, since they are constructed by deleting the communication point and event related to a team $T_e(t)$ from $P_i(t^+)$ and $\text{sched}_i(t^+)$, respectively. Next, robot $i$ checks if after deleting the communication point of team $T_e(t)$, it can still ensure that it can arrive at the communication points of the teams included in $\mathcal{P}_i$ at the designed meeting time instants. If not, then robot $i$ does not leave $T_e(t)$ and checks if it can leave another team from the set $M_i(t)$ (line 7, Alg. 17). Otherwise, robot $i$ checks if leaving $T_e(t)$ and getting replaced in this team by another robot $j \in T_m(t) \{ i \}$ decreases the cost $\sum_{r \in T_m(t)} J(P_r(t^+))$, as follows (lines 8-22, Alg.17).

Cost of joining team $T_e(t)$

Second, to check if replacing $i$ with another robot $j \in T_m(t) \{ i \}$ in team $T_e(t)$ decreases the cost $\sum_{r \in T_m(t)} J(P_r(t^+))$, all robots $j \in T_m(t) \{ i \}$ compute (in parallel) the additional cost $J_j^e$ of traveling towards the communication point associated with team $T_e(t)$ (lines 8-10, Alg.17). The cost $J_j^e$ is defined as

$$J_j^e = J(Q_j^e) - J(Q_j^e) \geq 0, \forall j \in T_m(t) \{ i \},$$

155
Algorithm 17: Update of Teams

Input: $M_i(t)$, $\text{sched}_i(t)$, $P_i(t^+)$, for all $i \in T_m(t)$

Output: $M_i(t^+)$, $\text{sched}_i(t^+)$, $P_i(t^+)$, for all $i \in T_m(t)$

1. $\kappa = 0$;
2. Initialize $M_i(t^+) = M_i(t)$, $\text{sched}_i(t^+) = \text{sched}_i(t)$ $\forall i \in T_m(t)$;
3. for $i \in T_m(t)$ do
   for $e \in M_i(t)\setminus\{m\}$ do
      Compute cost $J_i^e$ as per (6.16);
      Compute $\overline{M}_i$, $\overline{\text{sched}}_i$, and $\overline{P}_i$;
      if $i$ can respect meeting time instants then
         for $j \in T_m\setminus\{i\}$ do
            Compute $\overline{M}_j$, $\overline{\text{sched}}_j$, and $\overline{P}_j$;
            Given $\overline{P}_j$, compute cost $J_j^e$ as per (6.17);
         Construct the set $R_{j\rightarrow i}$;
         Find $j^* = \min_{j \in R_{j\rightarrow i}} J_j^e$;
         if $J_{j^*}^e < J_i^e$ then
            $\kappa = \kappa + 1$;
            $M_i(t^+) = M_i$ (leave team $T_e(t)$);
            $\text{sched}_i(t^+) = \overline{\text{sched}}_i$;
            $P_i(t^+) = \overline{P}_i$;
            $M_{j^*}(t^+) = \overline{M}_{j^*}$ (join team $T_e(t)$);
            $\text{sched}_{j^*}(t^+) = \overline{\text{sched}}_{j^*}$;
            $P_{j^*}(t^+) = \overline{P}_{j^*}$;
         if $|T_e| = 1$ then
            Robot $j^*$ leaves team $T_e$;
   end
end

where the paths $Q_j^e$ and $\overline{Q}_j^e$ are defined as

$$Q_j^e = [\overline{P}_j^{k_j^e-1}, \overline{P}_j^{k_j^e}, \overline{P}_j^{k_j^e+1}],$$  \hspace{1cm} (6.18)

and

$$\overline{Q}_j^e = [\overline{P}_j^{k_j^e-1}, \overline{P}_j^{k_j^e+1}].$$ \hspace{1cm} (6.19)

In (6.18)-(6.19), $\overline{P}_j$ denotes the path that robots $j \in T_m(t)\setminus\{i\}$ will have to follow at time $t^+$ if they join team $T_e(t)$ and $k_j^e$ is the index of the entry of $\overline{P}_j$ where the (already selected) communication point of team $T_e(t)$ will be placed.\textsuperscript{6}

\textsuperscript{6} Note that the only difference between the paths $P_j$ and $P_j(t^+)$ is that the first one contains the communication point for team $T_e(t)$ while the second one may not.
In what follows, we discuss how robots \( j \in \mathcal{T}_m(t) \setminus \{i\} \) compute the cost (6.17). First, they construct the set of teams \( \mathcal{M}_j \) and the corresponding communication schedules \( \text{sched}_j \) that these robots will have at time \( t^+ \), if they join team \( \mathcal{T}_e(t) \). Specifically, the sets \( \mathcal{M}_j \) are constructed as \( \mathcal{M}_j = \mathcal{M}_j(t^+) \cup \{e\} \) and the communication schedules \( \text{sched}_j \) are constructed as discussed in Section 6.2.3. Once the robots construct the communication schedules \( \text{sched}_j \), they construct the paths \( P_j \) in order to compute the cost (6.17). Note that it is possible that there exist robots \( j \in \mathcal{T}_m(t) \setminus \{i\} \) that satisfy \( e \in \mathcal{M}_j(t^+) \). Such robots have already placed the communication point of team \( \mathcal{T}_e(t) \) in their respective paths \( P_j(t^+) \) at an entry with index \( k^e_j \), by construction of the paths \( P_j(t^+) \). Therefore, for these robots, it holds that \( \mathcal{M}_j = \mathcal{M}_j(t^+) \), \( \text{sched}_j = \text{sched}_j(t^+) \), and \( P_j = P_j(t^+) \). Given \( P_j \), robots \( j \in \mathcal{T}_m(t) \cap \mathcal{T}_e(t) \setminus \{i\} \) can compute the cost \( J^e_j \) according to (6.17).

To construct the paths \( P_j \), for robots \( j \in \mathcal{T}_m(t) \setminus \mathcal{T}_e(t) \), we need to compute the index \( k^e_j \) of the entry in their paths \( P_j \) where the communication point of team \( \mathcal{T}_e(t) \) will be placed. This is computed by solving the following local integer program:

\[
\begin{align*}
\text{minimize} & \quad J^e_j(k^e_j) \\
\text{subject to} & \quad P^e_{j^e} = v_a, \\
& \quad k^e_j \in I^e_j.
\end{align*}
\] (6.20)

where the path \( P_j \) is initialized as \( P_j = P_j(t^+) \). In (6.20), the first constraint requires that the already selected communication point \( v_a, a \in \mathcal{C}_e \), of team \( \mathcal{T}_e(t) \) is placed at the \( k^e_j \)-th entry of \( P_j \). The second constraint requires that the index \( k^e_j \) is selected so that the schedule \( \text{sched}_j \) is respected. Specifically, in the second constraint, the set \( I^e_j \) collects all the indices of entries in \( P_j \) at which the communication point \( v_a \), \( a \in \mathcal{C}_e \) can be placed so that \( \text{sched}_j \) is respected and is defined as follows

\[
I^e_j = \{k_j \mid k^\text{prev}_j < k_j \leq k^\text{next}_j\}. \] (6.21)
In (6.21), $k_j^{\text{prev}}$ and $k_j^{\text{next}}$ are indices of entries in $\mathcal{P}_j$ that point to communication points associated with the communication events that should occur before and after the communication event of team $\mathcal{T}_e$, as per $\mathbf{sched}_j$. In words, the index $k_j^{e}$ is selected so that the cost $J^e_j(k_j^e)$, defined in (6.17), is minimized and the path $\mathcal{P}_j$ respects the communication schedule $\mathbf{sched}_j$. Next, the robots $j \in \mathcal{T}_m(t)\backslash \mathcal{T}_e(t)$ construct the path $\mathcal{P}_j$ as per the solution of (6.20) and compute the corresponding cost $J^e_j$.

**Joining team $\mathcal{T}_e(t)$**

Third, once the robots $j \in \mathcal{T}_m(t)\backslash \{i\}$ construct the paths $\mathcal{P}_j$, they check if, given their dynamics (6.1), they can arrive at the communication points that appear in $\mathcal{P}_j$ at the selected meeting time instants. The robots $j \in \mathcal{T}_m(t)\backslash \{i\}$ that can respect these meeting time instants are collected in the set $\mathcal{R}_{j\to i}$ and are candidate robots to replace robot $i$ in team $\mathcal{T}_e(t)$ (line 11, Alg. 17). Next, given the costs $J^e_j$, we select the robot $j^* = \min_{j \in \mathcal{R}_{j\to i}} J^e_j$ (line 12, Alg. 17). If $J^e_{j^*} < J^e_i$, then robot $j^*$ (if it exists) replaces robot $i$ in team $\mathcal{T}_e(t)$ (line 13, Alg. 17). Finally, if such a replacement occurs, then robots $j^*$ and $i$ update (i) the set of teams they will belong to at time $t^+$ as $\mathcal{M}_i(t^+) = \overline{\mathcal{M}}_i$ and $\mathcal{M}_j(t^+) = \overline{\mathcal{M}}_j$; (ii) their communication schedules as $\mathbf{sched}_i(t^+) = \overline{\mathbf{sched}}_i$, $\mathbf{sched}_{j^*}(t^+) = \overline{\mathbf{sched}}_{j^*}$; and (iii) their paths as $\mathcal{P}_i(t^+) = \overline{\mathcal{P}}_i$, $\mathcal{P}_{j^*}(t^+) = \overline{\mathcal{P}}_j$ (lines 15-20, Alg. 17). Note that every time a team $\mathcal{T}_e(t)$, $e \in \cup_{r \in \mathcal{T}_m(t)} \mathcal{M}_r(t)\backslash \{m\}$ changes, the iteration index $\kappa$, initialized as $\kappa = 0$ (line 1, Alg. 17), is updated as $\kappa = \kappa + 1$ (line 1, Alg. 17). This iteration index is only used in the proof of Proposition 6.4.2. Finally, note that if after robot $j^*$ joins $\mathcal{T}_e(t^+)$, it holds that $\mathcal{T}_e(t^+) = \{j^*\}$, then robot $j^*$ deletes the event $e$ from $\mathbf{sched}_{j^*}(t^+)$ and the respective communication point $v_a$, $a \in \mathcal{C}_e$ from $\mathcal{P}_{j^*}(t^+)$ (line 22, Alg. 17). Then, it holds that $\mathcal{T}_e(t^+) = \emptyset$ and the node $e$ is deleted from the team membership graph $\mathcal{G}_T(t^+)$. The above process is repeated for all teams $e \in \mathcal{M}_i(t)\backslash \{m\}$ and sequentially across all robots $i \in \mathcal{T}_m(t)$. When Algorithm 17 terminates, then all robots $i \in \mathcal{T}_m(t)$
leave the current communication point and delete it from the path $P_i(t^+)$ (line 7, Alg. 18).

Finally, observe that Algorithm 17 requires that all the sets $M_i(t^+)$ include the team with index $m$; see line 4. This ensures that $|M_i(t^+)| \geq 1$, as required in Section 5.1. Moreover, as it will be shown in Proposition 6.4.1, this ensures that the graph $G_T(t^+)$ remains connected as long as $G_T(t)$ is connected, without requiring knowledge of any changes that simultaneously may occur in this graph during communication events in other teams, as also required in Section 5.1.

**Remark 6.3.2** (Update of team $T_e(t^+)$). Note that robots $r \in T_e(t^+) \setminus T_m(t)$ are not aware of the new members $j \in T_m(t)$ that joined $T_e(t^+)$ during the communication event in team $T_m(t)$. Thus, when a robot in team $T_e(t^+)$ arrives at the respective communication point, it has to notify all other robots in this team about the robots it has replaced. This ensures that robots in team $T_e(t^+)$ will not wait indefinitely at the communication points for old members of this team.

**Remark 6.3.3** (Deletion of team $T_e(t^+)$). Recall that at iteration $\kappa > 0$ of Algorithm 17, a team $T_e(t^+)$ is deleted from the graph $G_T(t^+)$ and robot $j^*$ leaves team $T_e(t^+)$ only if $T_e(t^+) = \{j^*\}$. Also, recall that the team $T_e(t^+)$ may be updated simultaneously by more than one team that communicates at time $t$. As a result, robots in team $T_m(t)$ are not aware of the members of team $T_e(t^+) \setminus T_m(t)$; see also Remark 6.3.2. Nevertheless, robot $j^*$ can locally check if it is the only member of team $T_e(t^+)$ without requiring global knowledge. Specifically, $T_e(t^+) = \{j^*\}$ can only happen if during the communication event of team $T_m(t)$ at time $t$, robots $i \in T_m(t)$ and $j^* \in T_m(t)$ are the only members of $T_e(t)$, i.e., $T_e(t) = \{i, j^*\}$ and, therefore, in this case $T_e(t)$ is updated only by robots in team $T_m(t)$. Note also that robots can join new teams only by substituting existing robots in this team. Therefore, it is not possible that a robot $r \notin T_m(t)$ joins the team $T_e(t) = \{i, j^*\}$ at time $t$ without robots $i$ and $j^*$ knowing it.
Algorithm 18: Online Execution of $\mathcal{P}_i(t)$, $t \geq t_0$

Input: Sets $\mathcal{C}_m$, $\forall m \in \mathcal{M}$, $\mathcal{P}_i(t_0)$

1. while $\text{Terminate}=0$ do
   2. Move towards $\mathcal{P}_i^1(t)$;
   3. if $(x_i(t) = \mathcal{P}_i^1(t)) \land (\mathcal{P}_i^1(t) = \mathbf{v}_j, j \in \mathcal{C}_m), m \in \mathcal{M}_i(t)$ then
      4. Wait until all robots of team $\mathcal{T}_m(t)$ arrive;
      5. Update paths, teams, and schedules (Algorithm 15);
   6. if $(x_i(t) = \mathcal{P}_i^1(t))$ then
      7. Delete the visited waypoint $\mathcal{P}_i^1(t)$ from $\mathcal{P}_i(t)$;

This prevents cases where robot $j^*$ would erroneously delete team $\mathcal{T}_e(t^*)$.

6.3.6 Online Execution of Paths

In this section, we discuss how the robots execute the paths $\mathcal{P}_i(t)$; see also Algorithm 18. Specifically, given paths $\mathcal{P}_i(t)$, $t \geq t_0$, every robot $i$ moves towards the first unvisited waypoint of their paths $\mathcal{P}_i(t)$, which is always $\mathcal{P}_i^1(t)$, since the robots delete from $\mathcal{P}_i(t)$ the waypoints that they have already visited (line 2, Alg. 18).

When robot $i$ reaches the waypoint $\mathcal{P}_i^1(t)$, it checks whether this waypoint is related to the assigned task or to a communication point associated with a team $\mathcal{T}_m(t)$, $m \in \mathcal{M}_i(t)$ (line 3, Alg. 18). If it is related to the assigned task, then robot $i$ deletes the waypoint $\mathcal{P}_i^1(t)$ from the path $\mathcal{P}_i(t)$ (line 7, Alg. 18), as there is no need to store it anymore, and moves towards the next waypoint $\mathcal{P}_i^1(t)$ (line 2, Alg. 18). Otherwise, if $\mathcal{P}_i^1(t)$ is a communication point, i.e., $\mathcal{P}_i^1(t) = \mathbf{v}_j, j \in \mathcal{C}_m, m \in \mathcal{M}_i(t)$, then robot $i$ communicates with all other robots in team $\mathcal{T}_m(t)$ and coordinate to update their corresponding paths $\mathcal{P}_i(t^*)$, the next communication point for team $\mathcal{T}_m(t^*)$, the set of teams $\mathcal{M}_i(t^*)$, and the respective communication schedules $\text{sched}_i(t^*)$ that they will have at the time instant $t^*$, i.e., right after leaving this communication point (line 5, Alg. 18). Note that uncertainty and exogenous disturbances may affect the arrival times of the robots at the communication points. Thus, if robot $i \in \mathcal{T}_m(t)$ is not able to arrive at $\mathbf{v}_j, j \in \mathcal{C}_m$ at the meeting time instant $t^m$ computed in Section
6.3.4, then it executes the following control policy (line 4, Alg. 18).

**Definition 6.3.4** (Control policy at communication points). *Every robot \( i \) that arrives at a communication location \( P_i^1(t) = v_j, j \in C_m, m \in M_i(t) \), waits there indefinitely, or until all other robots in the team \( T_m(t) \) arrive.*

### 6.4 Correctness

In this section, we present results pertaining to correctness of the proposed control scheme. First, we show that the graph \( G_{T_m(t)} \) remains connected for all time if the teams are updated as per Algorithm 17, as required in Section 5.1. The proof of the following proposition can be found in Appendix A.

**Proposition 6.4.1** (Connectivity of \( G_{T_m(t)} \)). *The dynamic team membership graph \( G_{T_m(t)} \) remains connected for all \( t > t_0 \) as long as the teams \( T_m(t), m \in M \) are updated by Algorithm 17 and the graph \( G_{T_m(t_0)} \) is connected.*

Next, we show that updating the teams according to Algorithm 17 decreases the total cost \( \sum_{i=1}^{N} J(P_i(t^+)) \).

**Proposition 6.4.2** (Optimality of Teams). *Let the teams \( T_{mb}(t) \), where \( b \in \{1, \ldots, B\} \) for some \( 0 < B < M \), communicate at time \( t \) and let \( \hat{P}_i(t^+) \) and \( P_i(t^+) \) denote the paths of robots \( i \in T_{mb}(t) \) right before and right after the update of teams by Algorithm 17, respectively. Then, it holds that \( \sum_{i \in \mathcal{N}} J(\hat{P}_i(t^+)) - \sum_{i \in \mathcal{N}} J(P_i(t^+)) \geq 0 \).*

*Proof.* This result holds by construction of Algorithm 17. First, we show that \( \sum_{i \in T_{mb}(t)} J(\hat{P}_i(t^+)) - \sum_{i \in T_{mb}(t)} J(P_i(t^+)) \geq 0 \) for all teams \( T_{mb}(t) \), where \( b \in \{1, \ldots, B\} \) and then we show that this is equivalent to \( \sum_{i \in \mathcal{N}} J(\hat{P}_i(t^+)) - \sum_{i \in \mathcal{N}} J(P_i(t^+)) \geq 0 \).

Recall that the iteration index \( \kappa \) in Algorithm 17 is initialized as \( \kappa = 0 \) and is updated as \( \kappa = \kappa + 1 \) every time a team \( T_e \) is updated, where \( e \in \cup_{i \in T_{mb}(t)} M_i(t) \setminus \{mb\} \).
We denote the path of robot $i$ at iteration $\kappa$ by $P_{i,\kappa}(t^+)$, i.e., $P_{i,0} = \hat{P}_i(t^+)$ and $P_{i,\hat{\kappa}} = P_i(t^+)$, where $\hat{\kappa}$ stands for the total number of iterations $\kappa$ that Algorithm 17 runs until it terminates. Also, we denote the team $T_e$ at iteration $\kappa$ by $T_{e,\kappa}$.

Assume that at iteration $\kappa$ robot $i$ is replaced by robot $j^*$ in team $T_e$. By construction of the path $P_{i,\kappa}$, we get the cost of $P_{i,\kappa}$ is

$$J(P_{i,\kappa}) = J(P_{i,\kappa-1}) - J_i^e,$$  

(6.22)

where the cost $J_i^e$ is defined in (6.16). Similarly, by construction of $P_{j^*,\kappa}$, we have that the cost of $P_{j^*,\kappa}$ is

$$J(P_{j^*,\kappa}) = \begin{cases} J(P_{j^*,\kappa-1}), & \text{if } j^* \in T_{e,\kappa-1} \\ J(P_{j^*,\kappa-1}) + J_{j^*}^e, & \text{otherwise,} \end{cases}$$  

(6.23)

where $J_{j^*}^e$ is defined in (6.17). Combining (6.22) and (6.23) and using the fact that $J_i^e > J_{j^*}^e$ by assumption, and $J_i^e \geq 0$ by definition, we get that

$$J(P_{i,\kappa}) + J(P_{j^*,\kappa}) < J(P_{i,\kappa-1}) + J(P_{j^*,\kappa-1}).$$  

(6.24)

For all other robots in team $T_{mb}(t)$, we have that $J(P_{r,\kappa}) = J(P_{r,\kappa-1})$, since these robots do not update their paths at iteration $\kappa$. Combining this with inequality (6.24), we get that $\sum_{r \in T_{mb}(t)} J(P_{r,\kappa}) < \sum_{r \in T_{mb}(t)} J(P_{r,\kappa-1})$. Following the same logic from $\kappa = 0$ to $\kappa = \hat{\kappa}$ yields $\sum_{r \in T_{mb}(t)} J(P_{r,0}) < \cdots < \sum_{r \in T_{mb}(t)} J(P_{r,\kappa}) < \cdots < \sum_{r \in T_{mb}(t)} J(P_{r,\hat{\kappa}})$, i.e., $\sum_{i \in T_{mb}(t)} J(\hat{P}_i(t)) - \sum_{i \in T_{mb}(t)} J(P_i(t)) > 0$. Note that if $\hat{\kappa} = 0$, i.e., if after executing Algorithm 17, it holds $\mathcal{M}_i(t^+) = \mathcal{M}_i(t)$, for all $i \in T_{mb}(t)$, then we have that $\sum_{i \in T_{mb}(t)} J(\hat{P}_i(t)) - \sum_{i \in T_{mb}(t)} J(P_i(t)) = 0$. Therefore, we get that $\sum_{i \in T_{mb}(t)} J(\hat{P}_i(t^+)) - \sum_{i \in T_{mb}(t)} J(P_i(t^+)) \geq 0$. Since this results holds for all $b \in \{1, \ldots, B\}$, we get

$$\sum_{b \in \{1, \ldots, B\}} \sum_{i \in T_{mb}(t)} J(\hat{P}_i(t^+)) - \sum_{b \in \{1, \ldots, B\}} \sum_{i \in T_{mb}(t)} J(P_i(t^+)) \geq 0$$  

(6.25)
Observe that all other robots \( r \in \mathcal{N} \setminus \bigcup_{b \in \{1, \ldots, B\}} \mathcal{T}_{mb}(t) \) do not update their teams, since they do not participate in any communication event, i.e., \( J(\hat{\mathcal{P}}_r(t)) = J(\mathcal{P}_r(t)) \).

Observe also that \( \mathcal{T}_{mb}(t) \cap \mathcal{T}_{mb}(t) = \emptyset \), for all \( b \neq \bar{b}, b, \bar{b} \in \{1, \ldots, B\} \), since by assumption the robots in \( \mathcal{T}_{mb}(t) \) and \( \mathcal{T}_{mb}(t) \) communicate simultaneously at time \( t \).

Combining these two observations with (6.25), we get

\[
\sum_{i \in \mathcal{N}} J(\hat{\mathcal{P}}_i(t^+)) - \sum_{i \in \mathcal{N}} J(\mathcal{P}_i(t^+)) \geq 0,
\]

completing the proof.

Next, in Theorems 6.4.8 and 6.4.10, we show that if the robots follow paths \( \mathcal{P}_i(t) \) constructed and executed as per Algorithm 15, both the intermittent connectivity requirement and the assigned task are satisfied. To show these results, we first need to show that the system is \textit{deadlock-free}. Specifically, we assume that there is a \textit{deadlock}, if there are robots in any team \( \mathcal{T}_m(t) \) that are waiting forever at a communication point for the arrival of all other robots in team \( \mathcal{T}_m(t) \) due to the control policy defined in Definition 6.3.4. To show that the system is deadlock-free we need first to state the following results. The proofs of the these results are provided in Appendix A.

First, we introduce the definition of \textit{equivalent schedules} that will be used in the proofs of the following results.

\begin{definition}[Equivalent schedules] Consider two communication schedules denoted by \( \mathbf{sched}_i(t) \) and \( \overline{\mathbf{sched}}_i(t) \) for robot \( i \). If the order in which the communication events for teams \( \mathcal{T}_m(t), m \in \mathcal{M}_i(t) \) occur is the same for both \( \mathbf{sched}_i(t) \) and \( \overline{\mathbf{sched}}_i(t) \), but the indices \( n^m_i \) and \( \overline{n^m_i} \) that satisfy \( \mathbf{sched}_i^{n^m_i}(t) = m \) and \( \overline{\mathbf{sched}}_i^{\overline{n^m_i}}(t) = m \), respectively, are possibly different, then these schedules are called equivalent. This equivalence relationship is denoted by \( \mathbf{sched}_i(t) \sim \overline{\mathbf{sched}}_i(t) \).
\end{definition}
Next, we show that there exist schedules $\text{sched}_i(t)$ that are equivalent to the schedules $\text{sched}_i(t)$ constructed by Algorithm 13 and satisfy the two rules of Algorithm 1 in Chapter 2 for all $i \in \mathcal{N}$ and for all $t > t_0$. Recall that the first rule requires that the index of the entry of $\text{sched}_i(t)$ where the communication event for a team $\mathcal{T}_m$ will be placed should be the same for all $i \in \mathcal{T}_m$. The second rule requires that the communication events of any two teams $\mathcal{T}_m(t)$, $\mathcal{T}_n(t)$, where $m \in \mathcal{M}_i$, $n \in \mathcal{M}_j$, $m \neq n$, $i \neq j$, should not be placed at an entry in $\text{sched}_i(t)$ and $\text{sched}_j(t)$, respectively, with the same index. Recall also that Algorithm 1 in Chapter 2 constructs (fixed) communication schedules $\text{sched}_i(t)$, assuming fixed teams, that are executed indefinitely.

**Proposition 6.4.4** (Equivalent Schedules). There always exist schedules $\overline{\text{sched}}_i(t)$ that are equivalent to the schedules $\text{sched}_i(t)$ constructed by Algorithm 13 and satisfy the two rules of Algorithm 1 presented in Chapter 2, for all $i \in \mathcal{N}$.

**Proposition 6.4.5** (Deadlock-free–Fixed Teams). Consider fixed sets of teams $\mathcal{M}_i$ and respective fixed communication schedules $\text{sched}_i(t)$ constructed by Algorithm 1. Then, the schedules $\text{sched}_i(t)$ are deadlock-free if they are executed as per Algorithm 14.

The proof of the above result is identical to the first part of the proof of Proposition 6.4.6 in Chapter 3 and, therefore, is omitted. Note that the execution of fixed schedules as per Algorithm 14 coincides with the execution of these schedules described in Chapter 2. Next, using Propositions 6.4.4 and 6.4.5, we show that the system is deadlock-free as the robots update their communication schedules $\text{sched}_i(t)$ and the teams $\mathcal{M}_i(t)$ as per Algorithms 13 and 17, respectively. Specifically, using 6.4.5 we show that if the robots communicate according to the equivalent schedules that satisfy the two rules of Algorithm 1, which always exist as shown in Proposition 6.4.4, the system is deadlock-free.
6.4.4, then the system is deadlock-free. Then, we show that this results also holds if the robots communicate according to the schedules designed by Algorithm 13.

**Proposition 6.4.6** (Deadlock-free–Dynamic Teams). Consider dynamic sets of teams $\mathcal{M}_i(t)$ and schedules $\text{sched}_i(t)$ determined by Algorithms 17 and 13, respectively. If the paths $\mathcal{P}_i(t)$ are constructed and executed as per Algorithm 15, for all $t \geq t_0$, then the system is deadlock-free.

**Remark 6.4.7** (Bounded waiting times). Proposition 6.4.6 shows also that the waiting times of robots at the communication points are bounded.

Next, using Proposition 6.4.6, we show that the communication network $\mathcal{G}_c(t)$ is connected over time infinitely often.

**Theorem 6.4.8** (Intermittent Connectivity). Consider dynamic sets of teams $\mathcal{M}_i(t)$ and schedules $\text{sched}_i(t)$ determined by Algorithms 17 and 13, respectively. If the paths $\mathcal{P}_i(t)$ are constructed and executed as per Algorithm 15, for all $t \geq t_0$, then the communication network $\mathcal{G}_c(t)$ is connected over time infinitely often.

**Proof.** To show that $\mathcal{G}_c(t)$ is connected over time infinitely often, it suffices to show that all non-empty teams $\mathcal{T}_m(t)$ communicate infinitely often. This is because every robot belongs to at least one team $\mathcal{T}_m(t)$, by construction of the sets $\mathcal{M}_i(t)$, and the team membership graph $\mathcal{G}_\mathcal{T}(t)$ remains connected for all $t \geq t_0$, as shown in Proposition 6.4.1. To show this, it suffices to show that time interval between two communication events for all non-empty teams $\mathcal{T}_m(t)$ is finite. Throughout this proof we make use of the index $z_i$ introduced in Algorithm 14. Recall that $z_i$ is updated every time the robot participates in a communication event, i.e., it counts the number of communication events that robot $i$ has participated.

Assume that at time $t^m$ robots $i \in \mathcal{T}_m(t^m)$ communicate. In what follows, we show that despite any subsequent changes in the structure of the teams and in the
communication schedules that may occur after the time instant \( t^m \), there exists a finite time interval \( \Delta t \), so that at the time instant \( t^m_{\text{next}} = t^m + \Delta t \) the robots in \( \mathcal{T}_m(t^m_{\text{next}}) \) will communicate again. Notice that due to the online update of teams, it is possible that \( \mathcal{T}_m(t^m_{\text{next}}) \neq \mathcal{T}_m(t^m) \). To show that there exists such a finite time interval \( \Delta t \), we first show that the robots \( r \in \mathcal{T}_m(t^m_{\text{next}}) \) will participate in a finite number of communication events, i.e., i.e., that the indices \( z_r \) will increase a finite number of times, within the time interval \([t^m, t^m_{\text{next}}]\). To show this, recall first that the communication event for team \( \mathcal{T}_m(t), m \in \mathcal{O}_c \) is placed at an entry in \( \text{sched}_r(t) \) with index either \( \alpha(c) \) or \( g_c(m) > \alpha(c) \), by construction of Algorithm 13. Also, once \( m \) is placed in \( \text{sched}_r(t) \) at an entry with index \( g_c(m) \), it will always be there for robot \( r \in \mathcal{T}_m(t^m_{\text{next}}) \); see also Remark 6.2.5. Therefore, since the index of the entry in \( \text{sched}_r(t) \) where the communication event \( m \) is placed will eventually remain fixed, we conclude that as robots \( r \in \mathcal{T}_m(t^m_{\text{next}}) \) execute \( \text{sched}_r(t) \) as per Algorithm 14, the communication event \( m \) will be ‘reached’ after a finite number of increments of the indices \( z_r \).

Using the above result, we show that there exists a finite interval \( \Delta t \) so that at the time instant \( t^m_{\text{next}} = t^m + \Delta t \) the robots in \( \mathcal{T}_m(t^m_{\text{next}}) \) will communicate again. First, notice that the traveling time between consecutive communication events is finite, since between consecutive communication points in paths \( \mathcal{P}_i(t) \) there exists a finite number of task waypoints, by construction. Also, the waiting time at each communication point is bounded due to Proposition 6.4.6. Also, the number of communication points that robot \( r \) has to visit until it communicates with robots in team \( \mathcal{T}_m(t^m_{\text{next}}) \) is finite as shown before. Thus, all robots \( r \in \mathcal{T}_m(t^m_{\text{next}}) \) will arrive at the respective communication point after finite time interval \( \Delta t \) completing the proof. \( \square \)

The proofs of Proposition 6.4.6 and Theorem 6.4.8 imply that it is the order of
communication events that ensure intermittent connectivity and not the indices of entries in \( \text{sched}_i \) at which these communication events are placed. This is formally stated in the following corollary.

**Corollary 6.4.9 (Equivalent schedules).** Consider two equivalent communication schedules denoted by \( \text{sched}_i(t) \) and \( \overline{\text{sched}}_i(t) \) for all robots \( i \in \mathcal{N} \) that may change with time \( t \). If execution of the schedules \( \overline{\text{sched}}_i(t) \) as per Algorithm 14 ensures that the network is deadlock-free and intermittently connected infinitely often, then so does the execution of \( \text{sched}_i(t) \) and vice versa.

**Theorem 6.4.10 (Task).** The construction and execution of \( \mathcal{P}_i(t) \) as per Algorithm 18 ensures that all robots will satisfy their assigned task, i.e., there exists a finite time instant \( t' \geq t \) by which all robots \( i \in \mathcal{N} \) will have visited all task waypoints that appear in the path \( \mathcal{P}_i(t) \), for all \( t \geq t_0 \), and for all \( i \in \mathcal{N} \).

**Proof.** To show this result, it suffices to show that in the current path \( \mathcal{P}_i(t) \) there exists a waypoint \( \mathcal{P}_i^{k_i}(t) \) that can be reached by robot \( i \) at a finite time instant \( t' > t \) and there exists at least one task waypoint between the current location \( \mathcal{P}_i^1(t) \) and \( \mathcal{P}_i^{k_i}(t) \), for all \( t \geq t_0 \) and for all \( i \in \mathcal{N} \).

Assume that robots \( i \in \mathcal{I}_m(t^m) \) currently communicate and design new paths \( \mathcal{P}_i(t^{m,+}) \) where \( \mathcal{P}_i^{k_i}(t^{m,+}) \) denotes the next communication point of team \( \mathcal{I}_m(t^m) \). Recall that due to (6.13a), we have that at time \( t^m \) there exists at least one task waypoint (if there is any) between the current communication point \( \mathcal{P}_i^1(t^{m,+}) \) and the next communication point \( \mathcal{P}_i^{k_i}(t^{m,+}) \) of team \( \mathcal{I}_m \), for all \( i \in \mathcal{I}_m(t^m) \). In what follows, we divide the robots \( i \in \mathcal{I}_m(t^m) \) into two groups. The first group concerns the robots that will decide to stay in team \( \mathcal{I}_m \) until communication within this team

---

7 Here with slight abuse of notation, we assume that \( \mathcal{P}_i^1(t^{m,+}) \) denotes the communication point where the robots in \( i \in \mathcal{I}_m(t^m) \) communicated at time \( t^m \). In fact this location is deleted at time \( t^{m,+} \) from the paths \( \mathcal{P}_i(t^{m,+}) \), for all \( i \in \mathcal{I}_m(t^m) \); see Algorithm 18.
 occurs again. These robots are collected in the set $S_m \subseteq \mathcal{T}_m(t^m)$. The second group concerns the robots that will leave team $\mathcal{T}_m$ on their way towards their next communication point $\mathcal{P}^m_{i^m}(t^{m,+})$. These robots are collected in the set $L_m \subseteq \mathcal{T}_m(t^m)$. For robots $i \in S_m$, we know that there exists a finite time instant $t' > \bar{t}^m$ at which they will reach the communication point $\mathcal{P}^m_{i^m}(t^{m,+})$ due to Theorem 6.4.8. On the other hand, robots $i \in L_m$ will delete the communication $\mathcal{P}^m_{i^m}(t^{m,+})$ but they will reach the waypoint $\mathcal{P}^m_{i^m-1}(t^{m,+})$ in finite time again since these robots will reach in finite time all communication points that appear in $\mathcal{P}_i(t^{m,+})$ after the waypoint $\mathcal{P}^m_{i^m-1}(t^{m,+})$ due to Theorem 6.4.8. Since task waypoints are never removed from the designed paths, we have that robots $i \in S_m$ will visit at least one task waypoint as they move towards $\mathcal{P}^m_{i^m}(t^{m,+})$. Similarly, we get that robots $i \in L_m$ will also visit at least one task waypoint as they move towards $\mathcal{P}^m_{i^m-1}(t^{m,+})$.

Thus, we have shown that once robot $i$ participates at a communication event $m$ at time $t^m$, it will visit at least one task waypoint in finite time. Recall that due to Theorem 6.4.8, robot $i$ will participate in an infinite number of communication events. Since $\mathcal{P}_i(t)$ has a finite number of task waypoints, we conclude that all task waypoints in $\mathcal{P}_i(t)$ will be visited in finite time, for all $t \geq t_0$ and for all $i \in \mathcal{N}$, completing the proof. 

\[ \Box \]

6.5 Simulation Studies

In this section, a simulation study is provided that illustrates our approach for a network of $N = 15$ robots that reside in a $10 \times 10$ square workspace free of obstacles. Robots are initially categorized into $M = 12$ teams as follows: $\mathcal{T}_1(t_0) = \{1, 2, 9\}$, $\mathcal{T}_2(t_0) = \{3, 4, 5\}$, $\mathcal{T}_3(t_0) = \{6, 13\}$, $\mathcal{T}_4(t_0) = \{1, 3, 14\}$, $\mathcal{T}_5(t_0) = \{2, 5, 6, 11\}$, $\mathcal{T}_6(t_0) = \{4, 12, 14\}$, $\mathcal{T}_7(t_0) = \{5, 9, 15\}$, $\mathcal{T}_8(t_0) = \{4, 9, 12\}$, $\mathcal{T}_9(t_0) = \{6, 7, 10, 15\}$, $\mathcal{T}_{10}(t_0) = \{7, 8, 11\}$, $\mathcal{T}_{11}(t_0) = \{8, 10, 11, 12\}$, and $\mathcal{T}_{12}(t_0) = \{7, 10, 13\}$ resulting in a
connected graph $G_{T}(t_0)$. In the workspace, there are $R = 60$ communication points that are randomly located in $W$, where we select $|C_m| = 5$, for all $m \in \mathcal{M}$ and $C_m \cap C_n = \emptyset$, for all $m, n \in \mathcal{M}$. Also, we assume that the robot dynamics are given by $\dot{x}_i(t) = u_i(t), \|u_i(t)\| \leq u_{i\text{max}}$.

Robot 1 has to follow a finite path with $H_1 = 30$ waypoints, which is randomly generated at the beginning capturing point-to-point navigation tasks Karaman and Frazzoli (2011) or complex tasks captured by co-safe LTL formulas Bhatia et al. (2010), such as sequencing tasks Fainekos et al. (2005) that require visitation of regions of interest in a specific order. For all the other robots we select $H_i = \infty$. Specifically, we assume that robots 2 and $N$ have to follow periodic paths forever to accomplish their assigned tasks. The periodic paths are randomly generated at the beginning resembling in this way surveillance tasks Nardi et al. (2016), informative path planning Lan and Schwager (2016), or complex tasks captured by Linear Temporal Logic formulas, such as coverage Fainekos et al. (2005) and data gathering Guo and Zavlanos (2017). Also, the periodic path $H_2(t)$ goes through a user that receives the collected information. All the other robots have to follow infinite and aperiodic paths. These robots initially construct finite paths which are randomly generated and can be extended at random time instants by a random number of waypoints resembling reactive motion planning tasks in unknown or dynamic environments as, e.g., in Guo and Dimarogonas (2015); Alonso-Mora et al. (2017), or receding horizon planning approaches Hollinger and Singh (2010); Ulusoy and Belta (2014).

The initial communication schedules for the teams $T_m(t_0)$ have the following form:

---

8 For periodic paths, the length $K_i(t^+)$ of the path $P_i(t^+)$ in (6.13d) can be selected arbitrarily large, since it can be viewed as an infinite and known path.
The initial meeting time instants are selected as $t^m = 0$ for all teams $\mathcal{T}_m(t_0)$, $m \in \mathcal{M}$, which are clearly infeasible. The resulting waiting times at the communication points for the first 20 communication events of all teams are depicted in Figure 6.4(a). Observe that despite the initially infeasible meeting time instants the waiting time at the selected communication points is eventually zero for all teams except for two.
Figure 6.5: Graphical depiction of the consensus of numbers \( v_i(t) \).

The non-zero waiting time for these teams is due to robot 1. Specifically, robot 1 has a finite task to accomplish and, therefore, there exists a time instant \( t' \) after which the path of this robot consists of only communication points. Also, during the construction of the set \( M_1(t) \) by Algorithm 17, robot 1 often decides to be a member of only one team. If at time \( t \geq t' \) the robots of the only team that robot 1 belongs to select their next communication point to be the same as the current one, then robot 1 will not leave this communication point until the other robots in this team return back to it resulting in a non-zero waiting time. Note that in this case it is not possible to ensure zero waiting time. To prevent such cases, if robot 1 cannot remain stationary due to its dynamics, we need to ensure that \( M_i(t) \geq 2 \) and that \( C_m \cap C_n \neq \emptyset \), for all pairs of teams \( T_m(t), T_n, m, n \in M \). The resulting waiting times, if the teams are fixed are also depicted in Figure 6.4(b). Observe in Figure 6.4(b) that the waiting time is zero for all teams despite the finite task of robot 1. The reason is that in this case robot 1 always belongs to two teams \( T_1(t_0) \) and \( T_4(t_0) \), with \( C_1 \cap C_4 \neq \emptyset \). As a result, robot 1 always has to move between two distinct
communication points for teams $T_1(t_0)$ and $T_4(t_0)$ for all time $t \geq t'$ and never stays idle at a communication point waiting for other robots.

To illustrate that the proposed motion plans ensure intermittent communication among the robots, we implement a simple consensus algorithm over the dynamic network $G_c(t)$. Specifically, we assume that initially all robots generate a random number $v_i(t_0)$ and when all robots $i \in T_m(t)$ meet at $v_j$, $j \in C_m$ they perform the following consensus update $v_i(t) = \frac{1}{|T_m(t)|} \sum_{r \in T_m} v_r(t)$. Figure 6.5 shows that eventually all robots reach a consensus on the numbers $v_i(t)$, which means that communication among robots takes place infinitely often, as proven in Theorem 6.4.8. Figure 6.6(a) illustrates the evolution of the second smallest eigenvalue $\lambda_2(G_T(t))$ of the Laplacian matrix of the dynamic graph $G_T(t)$ every time $G_T(t)$ gets updated. Notice that $\lambda_2(G_T(t)) > 0$, for all $t \geq t_0$ which means that $G_T$ remains connected for all time, as expected due to Proposition 6.4.1. Figure 6.6(b) shows the evolution of the maximum degree of $G_T(t)$ with time. Figure 6.7(a) shows the size of the teams at the end of every communication event, i.e., after Algorithm 17 has terminated.
Figure 6.7: Figure 6.7(b) shows the teams that robots 1 (red diamond) and 14 (blue square) belong to at the end of every communication event in any team. Figure 6.7(a) shows size of each team $T_m(t)$ at the end of every communication event in any team.

Observe in this figure that the teams $T_6$ and $T_{11}$ are eventually empty. Figure 6.7(b) shows the teams that robots 1 and 14 belong to at the end of each communication event. Notice that robot 14 has been a member of all teams $T_m(t)$, $m \in \mathcal{M}$ at least once. Also, notice that robot 1 often decides to belong to only one team, e.g., after the 450-th communication event it belongs to either $T_8$ or $T_{10}$.

Figure 6.8(a) shows the total cost $\sum_{i=1}^{N} J(P_{i}^{1:k_i}(t_f))$ with respect to the index $k_i \in \{1, 2, \ldots, \min_{i \in N} K_i(t_f)\}$, where $t_f$ refers to the time instant when the numerical experiment was terminated, and $P_{i}^{1:k_i}(t_f)$ is a subsequence of $P_{i}(t_f)$ that collects the first $k_i$ waypoints of $P_{i}(t_f)$. Observe in Figure 6.8(a) that the total distance $\sum_{i=1}^{N} J(P_{i}^{1:k_i}(t_f))$ that the robots need to travel is smaller if the teams are updated as per Algorithm 17 than if the teams remain fixed as constructed at $t_0$. Note that in order to plot Figure 6.8(a), we never deleted waypoints from $P_{i}(t)$ that have been visited, as required in line 7 of Algorithm 18, i.e., $P_{i}(t) = x_i(t_0)$.

Figure 6.8(b) shows the cost difference $\sum_{i=1}^{N} J(\hat{P}_i(t^+)) - \sum_{i=1}^{N} J(P_i(t^+))$ every time a communication event within any team occurs, where $\hat{P}_i(t^+)$ and $P_i(t^+)$ de-
Figure 6.8: Figure 6.8(a) depicts the total traveled distance $\sum_{i=1}^{N} J(\mathcal{P}_{i}^{1:k_i}(t_f))$ with respect to the index $k_i \in \{1, 2, \ldots, \min_{i \in N} K_i(t_f)\}$. Figure 6.8(b) illustrates the cost difference $\sum_{i=1}^{N} J(\hat{\mathcal{P}}_i(t^+)) - \sum_{i=1}^{N} J(\mathcal{P}_i(t^+))$ every time a communication event in any team occurs.

Note the path that robot $i$ has constructed right before and after it updates its set $\mathcal{M}_i(t)$ of teams as per Algorithm 17, respectively. Notice that $\sum_{i=1}^{N} J(\hat{\mathcal{P}}_i(t^+)) - \sum_{i=1}^{N} J(\mathcal{P}_i(t^+)) \geq 0$ for all time $t \geq t_0$, as expected due to Proposition 6.4.2.
Conclusions and Future Work

7.1 Summary

In this dissertation, we designed the first distributed intermittent connectivity framework that (i) ensures that the communication network is connected over time, infinitely often (ii) is flexible enough to account for arbitrary dynamic tasks, and (iii) can be applied to large-scale networks.

The great challenge in developing intermittent connectivity protocols for networks of mobile robots is to decide (i) which robots talk to which, (ii) where, and (iii) when, so that the communication network is connected over time infinitely often. To address these challenges, in Chapter 2, we decomposed the network into small groups of robots, also called teams, so that every robot belongs to at least one team and that there is a path, i.e., a sequence of teams, where consecutive teams have non-empty intersections, connecting every two teams of robots, so that information can propagate in the network. Given such fixed teams, we first design communication schedules, independent of their assigned tasks, that determine when every team should communicate, so that the communication network is connected over time infinitely often. The designed communication schedules ensure that all teams communicate infinitely
often, i.e., that the communication network is connected over time infinitely often. Between communication events the robots can move in the workspace free of communication constraints to accomplish their assigned tasks. Theoretical guarantees and numerical experiments corroborate the proposed framework.

In Chapters 3-5, given user-specified fixed teams, we proposed distributed integrated path planning and intermittent connectivity frameworks that determine both where and when every team should communicate so that the assigned task is accomplished, the communication network is connected over time infinitely often, and a user-specified metric, such as total traveled distance or consumed energy, is minimized. Specifically, we considered high-level complex tasks captured by temporal logic formulas, state-estimation tasks, and time-critical dynamic tasks. We showed that employing the proposed intermittent connectivity framework for such tasks results in significant performance gains compared to the existing solutions in the literature that maintain communication for all time. Theoretical guarantees, numerical and experimental studies support the proposed distributed control algorithms.

Finally, in Chapter 6 we proposed a fully autonomous intermittent connectivity framework that can handle arbitrary dynamic tasks and also allows the robots to locally and online update the structure of the teams and construct the respective communication schedules, effectively allowing them to decide who they should talk to, so that they can better accomplish newly assigned tasks. The structure of the teams, the associated communication locations, and the time instants when communication within teams will occur are integrated online with task planning giving rise to paths, i.e., sequences of waypoints, that ensure that the assigned task is accomplished, the communication network is connected over time infinitely often, and a user specified metric is minimized. This is the first fully autonomous, distributed, and online intermittent connectivity framework that can handle arbitrary dynamic tasks and also controls the topology of the intermittently connected robot network to better
accomplish those tasks. At the same time, the proposed framework scales well with the size of the robot network. Theoretical guarantees and numerical experiments corroborate the proposed distributed control scheme.

7.2 Future Work

We have identified the following three topics as possible future research directions.

7.2.1 Realistic Communication Models for Underwater Applications

In underwater applications, such as ocean exploration, the robots typically communicate over aquatic acoustic channels that are not practical when the robots need to be deployed tens of kilometers apart due to limited bandwidth, or in applications that take place in cluttered and dispersed environments where long distances and environmental interference. Maintaining all-time connectivity in these applications is rather conservative and may prevent robots from accomplishing their goals as the robot mobility is always restricted by connectivity/proximity constraints. Therefore, in such applications, the proposed intermittent connectivity framework is much preferred over all-time connectivity methods. Integration of the proposed intermittent connectivity framework with realistic acoustic communication channels for underwater applications is an interesting future research direction. Challenges that need to be addressed are to (i) obtain models for acoustic channels, (ii) take into account uncertainty of channels, (iii) integrate such uncertain communication models with robot mobility, and (iv) design energy efficient paths, i.e., paths along which robot do not have to fight against strong currents, that lead to connected configurations.

7.2.2 Robustness to Robot Failures

The proposed intermittent connectivity framework can account for uncertainty in the traveling times between consecutive communication events by requiring the robots
to wait for all other members of their teams. However, this control policy at the communication points may result in a deadlock, if a robot fails and never arrives at the selected communication points. Extending the proposed control scheme so that it can account for robot failures would add an additional degree of robustness to the system. This would require to either introduce maximum waiting times at the communication points or design probabilistic models for the robots that capture their reliability and lifetime.

7.2.3 Uncertain and Dynamic Environments

Typically, robots operate in uncertain, unknown, and dynamic environments such as crowded human environments or underwater environments. In such environments, \textit{a priori} determined communication locations may become temporarily unavailable and, as a result, intermittent communication within a team may become impossible. To account for such uncertainties, probabilistic models are necessary that return a probability distribution over possible communication locations or, in other words, they return the locations where successful communication is most likely to happen.
Appendix A

Proofs of Chapter 6

A.1 Proof of Proposition 6.4.1

To prove this result, first recall that there is an edge between two nodes \( m \) and \( n \) of \( G_T(t) \) if \( T_m(t) \cap T_n(t) \neq \emptyset \). Recall also that the teams \( T_m(t) \) can be updated only during a communication event. Let \( t = t_1, t_2, \ldots, t_k, \ldots \) be an infinite sequence of time instants when at least one team communicates and at least one robot \( i \) in these teams decides to switch from the set of teams \( M_i(t_k) \) to \( M_i(t_{k+1}^+) \) determined by Algorithm 17 giving rise to the team membership graph \( G_T(t_{k+1}^+) \). Observe that for all \( t \in (t_{k-1}, t_k] \) the graph \( G_T(t) \) remains unaltered. Therefore, if \( G_T(t_{k-1}^+) \) is connected, so is the graph \( G_T(t) \) for all \( t \in (t_{k-1}, t_k] \) for all \( k \in \mathbb{N}_+ \). Therefore, to show this result it suffices to show that \( G_T(t_{k}^+) \) is also connected, for all \( k \in \mathbb{N}_+ \).

Assume that at time \( t_k > 0 \) the graph \( G_T(t_k) \) is connected meaning that there exists a path from any node \( m \) to any other node \( n \) in \( G_T(t_k) \). Let \( T_{mb}(t_k), b \in \{1, \ldots, B\}, M \geq B \geq 1 \) be teams that communicate at time \( t_k \). By construction of Algorithm 18, when communication within a team \( T_{mb}(t_k) \) occurs, the robots in \( i \in T_{mb}(t_k) \) coordinate to construct their sets \( M_i(t_{k+1}^+) \) as per Algorithm 17. In what
follows, we assume that there exists at least one robot $i \in T_{mb}(t_k)$, for at least one $b \in \{1, \ldots, B\}$, that constructs a set $\mathcal{M}_i(t_k^+) \neq \mathcal{M}_i(t_k)$. If there does not exist such a robot $i$, then $G_T(t_k^+) = G_T(t_k)$ and then $G_T(t_k^+)$ is connected by assumption. In other words, we assume that at time $t_k$ there exists a robot $j \in T_{mb}(t_k)$, for at least one $b \in \{1, \ldots, B\}$, that replaces robot $i \in T_{mb}(t)$ in team $T_e(t_k^+)$ so that $T_e(t_k^+) \neq T_e(t_k)$. Notice that since in general there can be more than one team communicating at time $t_k$, it is possible that the team $T_e(t_k)$ is updated by more than one team simultaneously.

There are three main steps to show that the graph $G_T(t_k^+)$ is connected. First, we determine the subgraphs of $G_T(t_k^+)$ that are affected by the update of teams $T_e(t_k)$. Second, we show that these subgraphs remain connected after updating team $T_e(t_k)$. Third, using the second part we show by contradiction that the graph $G_T(t_k^+)$ is connected.

First, notice that replacing robot $i \in T_{mb}(t_k)$ with robot $j \in T_{mb}(t_k)$ in team $T_e(t_k^+)$ can possibly delete only the edges between the node $e$ and all nodes $n$ in $G_T(t_k)$ that satisfy $i \in T_e(t_k) \cap T_e(t_k)$. We collect such nodes $n$ in the set $\mathcal{N}_e^i = \{n \in V_T(t_k) \mid n \neq e, \ i \in T_e(t_k) \cap T_e(t_k)\}$. To show that only the edges $(n, e) \in E_T(t_k)$ where $n \in \mathcal{N}_e^i$ can be deleted, consider two nodes $h, o \in V_T(t_k)$ such that $(h, o) \in E_T(t_k)$. Assume first that $h \neq e$, and $o \neq e$. Then, if $T_h(t_k) \cap T_o(t_k) \neq \emptyset$, then $T_h(t_k^+) \cap T_o(t_k^+) \neq \emptyset$ still holds even after the replacement of robot $i$ in team $T_e(t_k^+)$, since the structure of the teams $T_h$ and $T_o$ is not affected. Next, assume that either $h = e$ or $o = e$. Without loss of generality assume that $h = e$ and $o \notin \mathcal{N}_e^i$. Since $o \notin \mathcal{N}_e^i$ we have that $i \notin T_h(t_k) \cap T_o(t_k)$. Consequently, this means that the edge between nodes $h$ and $o$ cannot be affected by a replacement of robot $i$ in team $T_e(t_k^+)$. On the other hand, if $h = e$ and $o \in \mathcal{N}_e^i$, then the edge $(h, o)$ may be deleted at time $t_k^+$. For example, this can happen if $T_h(t_k) \cap T_o(t_k) = \{i\}$. Notice also that after robot $j$ joining team $T_e(t_k^+)$ new edges may appear in $G_T(t_k^+)$. However, addition of edges cannot affect
the connectivity of $\mathcal{G}_T(t^+_k)$ and, therefore, they are not examined in this proof. Next, we define the subgraph $\mathcal{G}^i_e = \{\mathcal{V}^i_e, \mathcal{E}^i_e\} \subseteq \mathcal{G}_T(t^+_k)$, where the set of nodes $\mathcal{V}^i_e$ is defined as $\mathcal{V}^i_e = \{e\} \cup \mathcal{N}^i_e$ and the set of edges $\mathcal{E}^i_e$ contains the edges between these nodes as they are determined by the set $\mathcal{E}_T(t^+_k)$. Notice that the set $\mathcal{E}^i_e$ collects all edges in $\mathcal{G}_T(t^+_k)$ that can possibly be deleted due to the substitution of robot $i$ by robot $j$ in team $\mathcal{T}_e(t^+_k)$ by definition of the set $\mathcal{N}^i_e$.

Second, we show that the graph $\mathcal{G}^i_e = \{\mathcal{V}^i_e, \mathcal{E}^i_e\}$ is connected by showing that all nodes $n \in \mathcal{V}^i_e \setminus \{m_b\}$ are connected to the node $m_b$ at $t^+_k$. Recall that $\mathcal{T}_{m_b}(t_k)$ denotes a team that communicates at time $t_k$ and that $m_b \in \mathcal{V}^i_e$ by definition of $\mathcal{V}^i_e$ and $\mathcal{N}^i_e$. In particular, we first show that there exists an edge $(m_b, e) \in \mathcal{E}^i_e$ and then we show that there exist edges $(m_b, n) \in \mathcal{E}^i_e$ for all $n \in \mathcal{V}^i_e \setminus \{e, m_b\}$. Notice that $(m_b, e) \in \mathcal{E}(t^+_k)$ since it holds that $j \in \mathcal{T}_{m_b}(t^+_k) \cap \mathcal{T}_e(t^+_k)$, since robot $j \in \mathcal{T}_{m_b}(t^+_k)$ joins team $\mathcal{T}_e(t^+_k)$ to replace robot $i$. Next we focus on edges $(m_b, n)$, for all $n \in \mathcal{V}^i_e \setminus \{e, m_b\}$. Observe that it holds that $i \in \mathcal{T}_n(t^+_k)$ and $i \in \mathcal{T}_{m_b}(t^+_k)$ for all $n \in \mathcal{V}^i_e \setminus \{e, m_b\}$, since robot $i$ does not leave these teams. Therefore, all nodes $n \in \mathcal{V}^i_e \setminus \{m_b\}$ are still connected to node $m_b$ at $t^+_k$. Thus, since all nodes $\mathcal{V}^i_e$ are connected to the node $m_b \in \mathcal{V}^i_e$, the subgraph $\mathcal{G}^i_e$ is connected.

Third, using the result shown in the second part of this proof, we show by contradiction that the graph $\mathcal{G}_T(t^+_k)$ remains connected. Specifically, assume that there exist two nodes $a_1$ and $a_2$ such that there is no path in $\mathcal{G}_T(t^+_k)$ that connects them. This implies that the path that was connecting these two nodes in $\mathcal{G}_T(t_k)$ was going through at least one of the deleted edges. Specifically, assume that such a path was going through the edge $(c_1, c_2) \in \mathcal{E}(t^+_k)$ and $(c_1, c_2) \notin \mathcal{E}_T(t^+_k)$. Since $(c_1, c_2)$ is a link that was deleted at time $t^+_k$, this means that the nodes $c_1, d_1$ belong to at least one of the subgraphs $\mathcal{G}^i_e$ due to the first part of this proof.\footnote{Recall that it is possible that a team $\mathcal{T}_e$ is updated by more than one teams $\mathcal{T}_{m_b}$ at the same time. For example, assume that teams $\mathcal{T}_{m_b}$ and $\mathcal{T}_{m_6}$ update team $\mathcal{T}_e$ simultaneously. Then, notice} This equivalently, means...
that there is no path in any of these subgraphs $G^i_e$ that connects the nodes $c_1$ and $c_2$ which contradicts the fact that all subgraphs $G^i_e$ are connected shown in the second part of this proof. Therefore, the graph $G_T(t_k^+)$ is connected if the graph $G(t_k)$ is connected.

Note that if after replacing robot $i$ with robot $j$ in $T_p(t_k)$, it holds that $T_e(t_k^+)$, then Algorithm 17 deletes node $e$ from $G_T(t_k^+)$. Note that before deleting this node, it holds that $G_T(t_k^+)$ is connected as shown before. In what follows, we show by contradiction that after the deletion of node $e$, the graph $G_T(t_k^+)$ remains connected. Specifically, assume that after deleting node $e$, the graph $G_T(t_k^+)$ gets disconnected. This means that before deleting node $e$, there exist two nodes $a_1$ and $a_2$ in $G_T(t_k^-)$ that are connected only through this node $e$. This implies that before deleting node $e$, it holds that $(a_1, e) \in \mathcal{E}(t_k^+)$, i.e., $T_e(t_k^+)$, i.e., $j \in T_{a_1}(t_k^+) \cap T_e(t_k^+)$. Similarly, we get that $(e, a_2) \in \mathcal{E}(t_k^+)$, i.e., $j \in T_{a_2}(t_k^+) \cap T_e(t_k^+)$. Since $T_e(t_k^+) = \{j\}$, this equivalently means that $j \in T_{a_1}(t_k^+) \cap T_{a_2}(t_k^+)$, i.e., $(a_1, a_2) \in \mathcal{E}_T(t_k^+)$ which contradicts the assumption that $a_1$ and $a_2$ are connected in $G_T(t_k^+)$ only through this node $e$.

Thus, the graph $G_T(t_k^+)$ remains connected after the deletion of nodes $e$ that satisfy $|T_e(t_k^+)| = 1$. Consequently, we get that $G_T(t)$ remains connected for all $t > t_0$ as long as the graph $G_T(t_0)$ is connected and the teams are updated as described in Algorithm 17 completing the proof.

A.2 Proof of Proposition 6.4.4

To show this result, it suffices to construct schedules $\text{\underline{sched}}_i(t)$ that satisfy the two rules required for the construction of the sequences $s$, by Algorithm 1 presented in Chapter 2 while ensuring that the communication events appear in both $\text{\underline{sched}}_i(t)$ that the resulting subgraphs $G^i_e$ and $G^r_e$, where $i \in T_{m_i}$ and $r \in T_{m_r}$ are not necessarily disjoint because it is possible that $i \in T_f$, with $f \in N^i_e$ and $f \in N^r_e$ (and/or $r \in T_q$, with $q \in N^i_e$ and $q \in N^r_e$). In this case, node $f$ (and/or node $q$) belongs to both $G^i_e$ and $G^r_e$. 

182
and \( \text{sched}_i(t) \) in the same order. We construct the schedules \( \overline{\text{sched}}_i(t) \) incrementally. Specifically, we initialize them as \( \overline{\text{sched}}_i(t) = \text{sched}_i(t) \) for all \( i \in \mathcal{N} \). Note that due to this initialization, it trivially holds that \( \overline{\text{sched}}_i(t) \sim \text{sched}_i(t) \). In what follows, we modify the \( \overline{\text{sched}}_i(t) \) so that the two rules of Algorithm 1 are satisfied, while preserving the property \( \overline{\text{sched}}_i(t) \sim \text{sched}_i(t) \).

If no robot \( i \in \mathcal{N} \) has updated its set \( \mathcal{M}_i(t) \) of teams until time \( t > t_0 \), then we have that \( \text{sched}_i(t) = \text{sched}_i(t_0) \), for all \( i \in \mathcal{N} \). Then, the schedules \( \overline{\text{sched}}_i(t) \) already satisfy the two rules of 1; see Section 6.2.2. Also, the schedules \( \overline{\text{sched}}_i(t) \) trivially satisfy \( \overline{\text{sched}}_i(t) \sim \text{sched}_i(t) \), since \( \overline{\text{sched}}_i(t) = \text{sched}_i(t) \). In what follows, we consider cases where at least one robot has updated its set \( \mathcal{M}_i(t) \) and, consequently, its communication schedule. In such cases, first we check if the events that are placed at the entries of \( \overline{\text{sched}}_i(t) \) with indices that belong to \( [\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|] \), for all \( c \in \{1, \ldots, \ell\} \) respect the first and the second rule of Algorithm 1. Recall that these indices point to entries in \( \overline{\text{sched}}_i(t) \) that at time \( t_0 \) are filled with X’s; see Section 6.2.2. If the two rules of Algorithm 1 are not satisfied, then we update the schedules \( \overline{\text{sched}}_i(t) \) so that these rules are satisfied while preserving the order at which the communication events occur in \( \text{sched}_i(t) \). Second, we follow the same process for the entries of all schedules \( \overline{\text{sched}}_i(t) \) with indices \( \alpha(c) \), for all \( c \in \{1, \ldots, \ell\} \).

First, we focus on the events that are placed at the entries of \( \overline{\text{sched}}_i(t) \) with indices that belong to \( [\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|] \), for all \( c \in \{1, \ldots, \ell\} \). Specifically, consider robots \( i \in \mathcal{T}_m(t) \), \( m \in \mathcal{O}_c \), that have placed the event \( m \) at the entry with index \( g_c(m) \) either due to the first or the second rule of Algorithm 13. We collect such robots in the set \( \mathcal{A}_m \). Also, consider robots \( j \in \mathcal{T}_m(t) \) that have placed the event \( m \) at the entry with index \( \alpha(c) \), i.e., at the entry where the event \( m \) was originally placed at \( t_0 \).\(^2\) We collect such robots in the set \( \mathcal{A}_m \). Notice that \( \mathcal{A}_m \cup \mathcal{A}_m = \mathcal{T}_m(t) \).

\(^2\) By construction of Algorithm 13, this implies that robot \( j \in \mathcal{T}_m(t) \) has not joined any other
since the event $m$ can be placed at the entry of $\text{sched}_i(t)$ with index either $\alpha(c)$ or $g_c(m)$ in case robot $j$ joins another team that belongs to $\mathcal{O}_c$; see the two rules of Algorithm 13. By definition of the function $g_c$ it holds that $g_c(m) \neq \alpha(c)$; see Section 6.2.3. Thus, the first rule for the event $m$ is not satisfied. However, the fact that robots $j \in \tilde{A}_m$ have placed the event $m$ at the entry with index $\alpha(c)$ implies that all entries of $\text{sched}_j(t)$ with indices in $[\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|]$ are filled with $X$’s by construction of Algorithm 13. Thus, in $\text{sched}_j(t)$, for all $j \in \tilde{A}_m$, we can replace the entry with index $a(c)$ with $X$ and place the event $m$ at the entry of $\text{sched}_j(t)$ with index $g_c(m)$. This way $\text{sched}_j(t)$ preserves the order at which communication events occur in $\text{sched}_j(t)$. The event $m$ in the schedules $\text{sched}_i(t)$, for all $i \in A_m$ remains at the entry $g_c(m)$. Consequently, after modifying the schedules $\text{sched}_j(t)$, for all $j \in \tilde{A}_m$, the first rule for the event $m$ is satisfied. The second rule of Algorithm 1 is trivially satisfied at entries with index $\text{ind} \in [\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|]$, for all $c \in \{1, \ldots, \ell\}$, across all schedules $\text{sched}_i(t)$, $i \in \mathcal{N}$. The reason is that the entry with index $\text{ind} \in [\alpha(c) + 1, \alpha(c) + |\mathcal{O}_c|]$ is reserved for a single and unique team with index $g_c^{-1}(\text{ind}) \in \mathcal{M}$.

Second, we focus on the events that are placed at the entries of the schedules $\text{sched}_i(t)$ with indices $\alpha(c)$, for all $c \in \{1, \ldots, \ell\}$. Similarly to the previous case, consider robots $i \in A_m$ and robots $j \in \tilde{A}_m$, where $m \in \mathcal{O}_c$ for at least one $c \in \{1, \ldots, \ell\}$. In this case, the first rule for the event $m$ is not satisfied, as discussed before. Following the same logic as before, we can update the schedules $\text{sched}_j(t)$, for all $j \in \tilde{A}_m$, so that the first rule is satisfied. The second rule of Algorithm 1 is satisfied at the $\alpha(c)$-th entry of $\text{sched}_i(t)$, for all $i \in \mathcal{N}$ and for all $c \in \{1, \ldots, \ell\}$ due to the second rule of Algorithm 13 and no further modification of $\text{sched}_i(t)$ is required. To show this, assume that at time $t$ the second rule of Algorithm 1
is satisfied at the entries with indices \( \alpha(c) \), for all \( c \in \{1, \ldots, \ell\} \). In what follows, we show by contradiction that the second rule will never be violated at the entries with indices \( \alpha(c) \). There are two scenarios that can eventually happen and violate the second rule of Algorithm 1 at the considered entries. The first one can occur if due to Algorithm 13, a robot \( i \) places at time \( t \) a communication event \( h \) at an entry of \( \text{sched}_i(t) \) (and consequently of \( \overline{\text{sched}}_i(t) \)) with index \( \alpha(c) \) while it holds \( \mathcal{T}_m(t) \cap \mathcal{T}_h(t) \neq \emptyset \). However, this can never happen since Algorithm 13 never places any event at an entry with index \( \alpha(c) \); see also Remark 6.2.5. As for the second scenario, consider teams \( \mathcal{T}_m(t) \) and \( \mathcal{T}_e(t) \) such that \( \mathcal{T}_m(t) \cap \mathcal{T}_e(t) = \emptyset \). Also, consider robots \( i \in \bar{\mathcal{A}}_m \) and \( j \in \bar{\mathcal{A}}_e \), \( m, e \in \mathcal{O}_c \), i.e., the events \( m \) and \( e \) are placed at an entry of \( \overline{\text{sched}}_i(t) \) and \( \overline{\text{sched}}_j(t) \), respectively, with the same index \( \alpha(c) \). The second rule will be violated, if at time \( t^+ > t \), it holds that \( \mathcal{T}_m(t^+) \cap \mathcal{T}_e(t^+) \neq \emptyset \) and both events \( m, e \) remain at the entry of their schedules that they were at time \( t \). To show that this can never happen, assume that during the communication event of a team \( \mathcal{T}_h(t) \), where \( i \in \mathcal{T}_h(t) \cap \mathcal{T}_m(t) \), robot \( i \in \bar{\mathcal{A}}_m \) decides to join team \( \mathcal{T}_e(t) \). Therefore, it holds that \( i \in \mathcal{T}_m(t^+) \cap \mathcal{T}_e(t^+) \). Thus, if the events \( m \) and \( e \) remain at the entry of \( \overline{\text{sched}}_i(t^+) \) and \( \overline{\text{sched}}_j(t^+) \), respectively, with the same index \( \alpha(c) \) then the second rule of Algorithm 1 is violated. However, such a scenario is prevented from the second rule of Algorithm 13 that will move the event \( m \) in \( \overline{\text{sched}}_i(t) \) from the entry with index \( \alpha(c) \) to the entry with index \( \text{g}_c(m) \).

To sum up, the schedules \( \overline{\text{sched}}_i(t) \) are initialized as \( \overline{\text{sched}}_i(t) = \text{sched}_i(t) \) and they are updated by moving the event \( m \in \mathcal{O}_c \) to the entry of \( \overline{\text{sched}}_i(t) \) with index \( \text{g}_c(m) \), if there exists another robot \( j \) that has placed the same event at the entry of \( \overline{\text{sched}}_j(t) \) with index \( \text{g}_c(m) \), for all \( m \in \mathcal{M}_i(t) \). Note that the construction of the schedules \( \overline{\text{sched}}_i(t) \) requires all robots to form a connected network to share information, which is not the case during the online construction of the schedules \( \text{sched}_i(t) \).
by Algorithm 13. By construction, the resulting schedules \(\overline{\text{sched}}_i(t)\) preserve the order at which communication events occur in \(\text{sched}_i(t)\), i.e., \(\overline{\text{sched}}_i(t) \sim \text{sched}_i(t)\) and satisfy the two rules of Algorithm 1, for all \(i \in \mathcal{N}\), completing the proof.

A.3 Proof of Proposition 6.4.6

In what follows, we show that the network can never reach a stationary configuration, i.e., a configuration where at least one robot waits indefinitely at a communication point for the arrival of other robots, when the paths \(\mathcal{P}_i(t)\) are executed as per Algorithm 18. The proof this result consists of two parts. In the first part we show that the system is deadlock-free when the robots execute the equivalent schedules \(\overline{\text{sched}}_i(t)\) constructed in Proposition 6.4.4 as per Algorithm 14. In other words, we show that there is no robot \(i\) with \(\overline{\text{sched}}_i^\text{eq}_i(t) = m\) that waits indefinitely until all other robots in team \(\mathcal{T}_m(t)\) participate at the communication event \(m\), for all \(t \geq t_0\); see also line 5 in Algorithm 14. In the second part, we show that if the system is deadlock-free under the execution of \(\overline{\text{sched}}_i(t)\), then so is when the robots execute and construct the paths \(\mathcal{P}_i(t)\) according to Algorithm 18 using the communication schedules \(\text{sched}_i(t)\).

Recall first that the teams and the schedules can be updated only during a communication event. Let \(t = t_1, t_2, \ldots, t_k, \ldots\) be an infinite sequence of time instants when at least one team communicates and at least one robot \(i\) in these teams decides to switch from the schedule \(\text{sched}_i(t_k)\) to \(\text{sched}_i(t_k^+),\) constructed as per Algorithm 13, and correspondingly switch from the equivalent schedule \(\overline{\text{sched}}_i(t_k)\) to \(\overline{\text{sched}}_i(t_k^+)\). Recall that the equivalent schedules are constructed so that they respect the two rules of Algorithm 1; see the proof of Proposition 6.4.4. Assume that all robots execute the equivalent schedules \(\overline{\text{sched}}_i(t)\) instead of \(\text{sched}_i(t)\). Observe that for all \(t \in (t_{k-1}, t_k]\) the schedules \(\text{sched}_i(t)\) and, consequently the equivalent sched-
ules $\text{sched}_i(t)$, do not change, for all $i \in \mathcal{N}$ and for all $k \in \mathbb{N}_+$. Therefore, since the schedules $\text{sched}_i(t)$ (and the teams) remain fixed for all $t \in (t_{k-1}, t_k]$ and satisfy the two rules of Algorithm 1, we conclude that that the network is deadlock-free for all $t \in (t_k, t_{k+1}]$ and for all $k \in \mathbb{N}_+$ if the robots execute them as per Algorithm 14 due to Proposition 2.4.2. Therefore, to complete the first part of the proof, it suffices to show that the system remains deadlock-free as the robots switch from the schedules $\text{sched}_i(t_k)$ to $\text{sched}_i(t_k^+)$. 

In what follows, we show that switching from the schedules $\text{sched}_i(t_k)$ to $\text{sched}_i(t_k^+)$ still ensures that the system is deadlock-free. We collect all teams that communicate at time instant $t_k$ in the set $\mathcal{D}(t_k)$. Let $m \in \mathcal{D}(t_k)$. Then, at time $t_k$ the current communication event that for robots $i \in \mathcal{T}_m(t_k)$ is $\text{sched}_i^{n^m_i(t_k)}(t_k) = m$, where $n^m_i(t_k)$ denotes the index of the entry in $\text{sched}_i(t_k)$ where the communication event $m$ is placed. At time $t_k$, robots in team $\mathcal{T}_m(t_k)$ communicate and at least one robot $i \in \mathcal{T}_m(t_k)$ constructs a new schedule $\text{sched}_i(t_k^+)$, by assumption. Note that since at least one robot $i$ constructs a new schedule $\text{sched}_i(t_k^+)$, all other robots in the network update their schedules (even if they did not update their set of teams) so that the two rules of Algorithm 1 are satisfied; see the proof of Proposition 6.4.4. Assume that the next communication event for robot $i$ is $\text{sched}_i^{n^h_i(t_k^+)}(t_k^+) = h$ determined by Algorithm 14, for some $h \in \mathcal{M}_i(t_k^+)$. Notice that the previous communication event $m$ is not removed from $\text{sched}_i(t_k^+)$, since $m \in \mathcal{M}_i(t_k^+)$ by construction of Algorithm 17, and is placed at an entry of $\text{sched}_i(t_k^+)$ with index $n^m_i(t_k^+)$. Thus, the transition from the event $\text{sched}_i^{n^m_i(t_k)}(t_k) = m$ to $\text{sched}_i^{n^h_i(t_k^+)}(t_k^+) = h$ is the same as the transition from $\text{sched}_i^{n^m_i(t_k^+)}(t_k^+) = m$ to $\text{sched}_i^{n^h_i(t_k^+)}(t_k^+) = h$ and no other communication

3 Notice that the index $n^m_i(t_k^+)$ is possibly different from the index $n^m_i(t_k)$ due to the second rule of Algorithm 13; see Example 6.2.8.
events are interleaved between $m$ and $h$ in $\text{sched}_i(t_k^+)$. Note that such a transition is deadlock-free since all schedules $\overline{\text{sched}}_i(t_k^+)$ satisfy the two rules of Algorithm 1, as shown in Proposition 6.4.4. Therefore, as shown in Proposition 2.4.2, all transitions within such schedules are deadlock-free. Since this is true for all $t_k$, $k \in \mathbb{N}_+$, for all robots $i \in \mathcal{N}$ and for all $t \geq t_0$ due to Proposition 6.4.4, we conclude that the network is deadlock-free under the execution of the schedules $\text{sched}_i(t)$ as per Algorithm 14, for all $t \geq t_0$.

Next, recall that by construction of $\mathcal{P}_i(t_k^+)$ in Algorithm 17, the order in which the communication points for teams in $\mathcal{M}_i(t_k^+)$ appear in $\mathcal{P}_i(t_k^+)$ is the same as the order in which the respective communication events appear in the schedule $\text{sched}_i(t_k^+)$. Also, recall that the communication events in both $\overline{\text{sched}}_i(t_k^+)$ and $\text{sched}_i(t_k^+)$ appear in the same order; see Definition 6.4.3. Therefore, the order in which the communication points for teams in $\mathcal{M}_i(t_k^+)$ appear in the paths $\mathcal{P}_i(t_k^+)$ and the order in which the respective communication events appear in the schedules $\overline{\text{sched}}_i(t_k^+)$ is the same, as well. Moreover, during the execution of the paths $\mathcal{P}_i(t_k^+)$ as per Algorithm 18, robots $i$ wait only at communication points, until all the other robots associated with the respective team arrive there, exactly as in the execution of the schedules $\overline{\text{sched}}_i(t_k^+)$ discussed in Algorithm 14. Therefore, execution of schedules $\overline{\text{sched}}_i(t)$ according to Algorithm 14 is equivalent to the execution of paths $\mathcal{P}_i(t)$ according to Algorithm 18. This result along with the fact that the system is deadlock-free under the execution of the schedules $\overline{\text{sched}}_i(t)$, as per Algorithm 14, which was shown before, entail that the system is deadlock-free for all $t \geq t_0$, as well, when all paths $\mathcal{P}_i(t)$ are executed as per Algorithm 18, which completes the proof.

188
Bibliography


Freundlich, C., Lee, S., and Zavlanos, M. M. (December 2016), “Distributed estima-
tion and control for robotic sensor networks,” in 55th Conference on Decision and
Control (CDC), pp. 3518–3523, Las Vegas, NV, IEEE.

Ghaffarkhah, A. and Mostofi, Y. (2010), “Channel learning and communication-
aware motion planning in mobile networks,” in IEEE American Control Confer-
ence, pp. 5413–5420, Baltimore, Maryland, USA.


Godsil, C. and Royle, G. (2001), Algebraic Graph Theory, New York: Springer-
Verlag.

local LTL specifications,” The International Journal of Robotics Research, 34, 218–
235.

Guo, M. and Dimarogonas, D. V. (December 2013), “Reconfiguration in motion plan-
ing of single-and multi-agent systems under infeasible local LTL specifications,”
in IEEE 52nd Annual Conference on Decision and Control (CDC), pp. 2758–2763,
Florence.

straints and Intermittent Communication,” in Robotics and Automation (ICRA),
IEEE International Conference on, Singapore (to appear).

wireless sensor networks for event monitoring,” IEEE Transactions on Parallel
and Distributed Systems, 23, 475–482.

Hollinger, G. and Singh, S. (May 2010), “Multi-robot coordination with periodic con-
nectivity,” in IEEE International Conference on Robotics and Automation (ICRA),
pp. 4457–4462, Anchorage, Alaska.

Hollinger, G., Singh, S., et al. (2012), “Multirobot coordination with periodic con-

“Distributed Data Fusion for Multirobot Search,” IEEE Transactions on Robotics,
31, 55–66.

Robotic Information Gathering,” in Robotics: Science and Systems, pp. 72–983,
Citeseer.


Biography

Ioannis (Yiannis) Kantaros was born in Tripoli, Greece on December 19, 1989. He received the Diploma in Electrical and Computer Engineering in 2012 from the University of Patras, Patras, Greece and the M.S. in Mechanical Engineering from Duke University, Durham, NC in 2017. The current publication is toward the doctorate degree in Mechanical Engineering at Duke University, Durham, NC. Yiannis received the Best Student Paper award at the 2nd IEEE Global Conference on Signal and Information Processing in 2014, and a fellowship from Gerondelis Foundation in 2016.

Recent publications are: Stergiopoulos et al. (2012a,b); Kantaros et al. (2014, 2015); Kantaros and Zavlanos (2014b,a, 2015, 2016a,c,b,d, 2017a); Kantaros et al. (2017b); Kantaros and Zavlanos (2017b,c, 2018c); Khodayi-mehr et al. (2017); Kantaros and Zavlanos (2018b,a); Kantaros et al. (2017a)