Context-Aware Information Gathering and Processing Towards Supporting Autonomous Systems in Industry 4.0 Scenarios

Razanne Abu-Aisheh

Thesis

of the

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Context-Aware Information Gathering and Processing Towards Supporting Autonomous Systems in Industry 4.0 Scenarios

presented by

Razanne Abu-Aisheh

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

at

Sorbonne Université

Presented on 27th February 2023.

Jury:

Nathalie Mitton	Inria, Lille , France	Reviewer
Isabelle Guérin-Lassous	ENS, Lyon , France	Reviewer
Razvan Stanica	INSA, Lyon , France	Examiner
Thomas Watteyne	Inria, Paris, France	PhD Adviser
Francesco Bronzino	ENS, Lyon , France	PhD Co-advisor
Lou Salaun	Nokia Bell Labs, Nozay , France	Industrial Advisor

Acronyms

MRS	Multi-Robot System
RSSI	Initial Received Signal Strength Indicator
IIoT	industrial Internet of Things
IoT	Internet of Things
NFA	Nearest Frontier Approach
BSO	Brain Storm Optimisation
PSO	Particle Swarm Optimisation
UI	User Interface
PDR	Packet Delivery Ratio
CPS	Cyber-Physical Systems

To anyone who ever wished for a PhD thesis to be dedicated to them

Acknowledgements

First and foremost, I am deeply grateful for the continuous support, insight and patience of my supervisors, Thomas Watteyne, Francesco Bronzino and Lou Salaun, along with Myriana Rifai who supervised me throughout the first two years of my PhD. Without them this thesis would not have been completed.

I would also like to thank my follow-up committee, Nathalie Mitton and Nikolaos Georgantas. Your feedback and suggestions have helped me refine my research and improve the quality of this thesis.

To my team mates Martina, Said, Anne, Trifun, Fil and Malisa, thank you for making the lab fun and full of life and just a cool place to be in general. Your friendship, humour, and support have made the long hours in the lab much more bearable and rewarding. I am so grateful for all the memories we have shared together.

To my friends, Bekki and Yasmeen thank you for your unwavering support and understanding. Your encouragement and positivity have kept me going during the challenging times.

And to my family, Mama, Baba, Mohannad and Manar thank you for your love and patience, I wouldn't be here if it weren't for you.

Thank you all for your contributions, support, and encouragement. This thesis is a reflection of our collective efforts, and I could not have done it without you.

Résumé

Les environnements de l'industrie 4.0 se caractérisent par la coexistence d'un ensemble diversifié de dispositifs, notamment des capteurs, des écrans à réalité mixte, des robots, des drones et des objets intelligents. Ces systèmes doivent être capables de prendre de manière autonome les décisions critiques en temps voulu nécessaires à l'exécution de tâches complexes sans intervention humaine. Une application essentielle de l'industrie 4.0 est l'exploration et la cartographie multi-robots d'environnements inconnus, en particulier dans le cadre de missions critiques telles que la détection des dangers et la recherche et le sauvetage. Ces missions partagent le besoin d'atteindre une couverture complète de l'espace explorable dans le temps le plus court possible. Pour minimiser le temps de réalisation, les robots de la flotte doivent être capables d'échanger des informations sur l'environnement de manière fiable entre eux. Cependant, les algorithmes d'exploration et de cartographie existants souffrent d'inexactitudes et d'inefficacités en raison de leur manque de connaissance contextuelle de leur environnement, notamment en termes de communications, de leur manque de flexibilité et d'adaptabilité à l'environnement, et donc, de l'ajout d'un retard inutile à la mission en cours. Dans cette thèse, nous étudions l'impact de la connaissance des communications sur la performance des expéditions d'exploration et de cartographie multi-robots, en termes de temps de réalisation. Nous évaluons les recherches existantes dans ce domaine et démontrons l'impact de la non prise en compte des problèmes de communication lors de la conception de tels algorithmes. À partir de là, nous proposons Atlas, un algorithme d'exploration et de cartographie qui prend en compte de manière native la perte de paquets, avec un taux d'achèvement de 100 Cependant, Atlas ne peut pas, à lui seul, gérer les scénarios où la connectivité est complètement perdue. Cela ajoute également un retard important à l'achèvement de la mission, car les paquets perdus sont retransmis périodiquement jusqu'à ce qu'ils soient reçus. Une solution est le placement de relais. La plupart des recherches sur le placement de relais pour les expéditions multi-robots tendent à se répartir en deux catégories. Premièrement, le placement des relais en fonction de la communication, basé sur l'indicateur de force du signal reçu (RSSI) initial, est utilisé. Cependant, cela nécessite l'exécution

d'une mission complète avant l'exploration pour trouver la position optimale des relais à placer. Deuxièmement, il faut maintenir une distance (spécifiée avant la mission) entre les relais et les robots d'exploration. Ces méthodes augmentent le temps nécessaire à l'exécution de la mission. La question de recherche devient : comment placer les relais pour maintenir une communication aussi fiable que possible, et aussi dynamiquement tout au long de la mission d'exploration sans connaissance préalable de l'environnement, de manière à réduire le retard de l'exploration et le temps de cartographie pour l'achèvement. Nous résolvons ce problème en proposant le "Connectivity Aware Relay Algorithm" (CARA), un algorithme dynamique de placement de relais sensible au contexte qui ne nécessite aucune connaissance préalable de l'environnement. Nous avons développé un simulateur open-source pour les expéditions multi-robots que nous avons utilisé pour tester les deux algorithmes contre les algorithmes de pointe. L'utilisation d'Atlas et de CARA permet de réaliser une expédition multi-robot dynamique et contextuelle qui construit de manière autonome une carte d'un environnement totalement inconnu, tout en plaçant dynamiquement des relais lorsque cela est nécessaire pour maintenir la connectivité, qui surpasse les algorithmes de pointe, en termes de temps de réalisation, d'un facteur 10.

Summary

Industry 4.0 environments are characterized by the coexistence of a diverse set of devices, including sensors, mixed-reality displays, robots, drones, and smart objects. These systems must be capable of autonomously taking critical in-time decisions necessary to perform complex tasks without human input. One essential application for Industry 4.0 is multi-robot exploration and mapping of unknown environments, especially in critical missions such as hazard detection and search and rescue. These missions share the need to reach full coverage of the explorable space in the shortest time possible. To minimize completion time, robots in the fleet must be able to exchange information about the environment reliably with one another. However, existing exploration and mapping algorithms suffer from inaccuracies and inefficiencies due to their lack of contextual awareness of their surroundings, especially in terms of communications, lacking flexibility and adaptability to the environment, and hence, adding unnecessary delay to the mission at hand. In this thesis, we investigate the impact of communication awareness on the performance of multi-robot exploration and mapping expeditions, in terms of time to completion. We evaluate existing research in the field and demonstrate the impact of not considering communication impairments when designing such algorithms. From there, we propose Atlas, an exploration and mapping algorithm that natively takes packet loss into account, with a 100% completion ratio even with Packet Delivery Ratios (PDRs) as low as 0.1. However, Atlas on its own cannot handle scenarios where connectivity is completely lost. It also adds a significant delay to the completion of the mission, as lost packets keep getting re-transmitted periodically until they are received. One solution is relay placement. Most research on relay placement for multi-robot expeditions tend to fall into two categories. First, communication-aware relay placement based on initial Received Signal Strength Indicator (RSSI) is used. However, this requires running a full mission prior to the exploration to find the optimal position for the relays to be placed. Second, maintaining a distance (specified prior to the mission) between relays and exploration robots. These methods add to the time it takes to complete the mission. The research question becomes how can we place relays to maintain communication as reliable as possible, and

also dynamically throughout the exploration mission without prior knowledge of the environment, in a way that reduces delay to the exploration and mapping time to completion. We solve this by proposing "Connectivity Aware Relay Algorithm" (CARA), a dynamic context-aware relay placement algorithm that does not require any prior knowledge of the environment. We developed an open-source simulator for multi-robot expeditions which we used to test both algorithms against state-of-the-art algorithms. Using both Atlas and CARA results in a dynamic context-aware multi-robot expedition that autonomously builds a map of a fully unknown environment, while dynamically placing relays when needed to maintain connectivity that outperforms state-of-the-art algorithms, in terms of time to completion, by a factor of 10.

Chapter 1

Introduction

This thesis contributes to the growing field of research on multi-robot coordination in unknown environments. Specifically, it focuses on utilising context aware communications to enhance the performance of a fleet of robots in exploration and mapping for critical time sensitive expeditions. This chapter provides an overall introduction into the topics related to this thesis and is organised as follows. Section 1.1 Introduces the role of autonomous systems in the context of Industry 4.0 scenarios. Section 1.2 discusses exploration and mapping as an application of autonomous systems in Industry 4.0 and highlights critical missions as a guiding application. Section 1.3 showcases the importance of communications in multi-robot coordination, specifically for exploration and mapping, as well as the impact and importance of considering lossy communications in such algorithms. Section 1.4 defines context aware systems. Section 1.5 explains the opportunities that utilising context aware communications bring to exploration and mapping algorithms, and how that can enhance the performance of a fleet of robots in such missions. Finally, Section 1.6 provides an outline of the organisation of the thesis.

1.1 Autonomous Systems in Industry 4.0

In this section we explain the concept of Industry 4.0 and walk through the significance of multi robot systems in its scenarios. We then discuss the opportunities that MRSs bring to time critical industrial services such as search and rescue and hazard detection and localisation.

Throughout history , three main industrial revolutions have evolved, leading to highly mechanised and automatised production and services. These three revolutions in order are :

1. 1st Industrial Revolution: characterised by the transition to more mechanised processes

2. 2nd Industrial Revolution: characterised by the intensive use of electrical energy

3. 3rd Industrial Revolution: characterised by the wide spread use of digital technology

The advanced digitisation of the 3rd industrial revolution, as well as smart connected objects of the Internet of Things (IoT), resulted in the emergence of the fourth industrial revolution, most commonly referred to as Industry 4.0. Industry 4.0 has been progressing exponentially in recent years [1], and is believed to cause change in more than just the principles of production, but also in the way individuals live and work fundamentally [2]. This industrial shift is characterised by automation of production and services, where interconnected machines control their own processes [3]. Such environments will be marked by the coexistence of a diverse set of devices, including sensors, mixed-reality displays, robots, drones, and smart objects. These systems must be capable of autonomously taking critical in-time decisions necessary to perform complex tasks without human input. Especially in time critical missions where human lives are involved such as search and rescue or hazard detection and localisation. Hence, the key element that characterizes industry 4.0 is the change in systems' connectivity due to IoT and Cyber-Physical Systems (CPS), resulting in an industrial age based on connectivity [4]. Multi-robot systems are examples of such systems, where a fleet of robots are connected and coordinated together to efficiently conduct a mission.

Multi-robot systems are widely used in industries that may impose a risk to human lives, such as in the chemical and nuclear industries, or in man-made or natural disasters and so on [5]. For example, after a natural disaster such as a hurricane or forest fire, the traditional response and recovery mission can be costly and even dangerous for the responders [6]. The inefficiency of rescue operations brings immeasurable losses to humans [7].

The practical implementation of multi-robot systems can significantly reduce the risk to human life and health when working in adverse or dangerous conditions. By using multiple robots, if one robot fails, there are still a number of operational robots left to continue the mission, which adds redundancy to the system [8]. In addition, a larger spatial area can be covered more efficiently if multiple robots are deployed, as they can complete a mission faster when the tasks are split across various robots working together towards a common goal. One of the major challenges for MRSs is to design appropriate coordination strategies between the robots that enable them to perform operations efficiently in terms of time and working space [9]. In this thesis we work on contributing solutions towards solving such challenges by designing End-to-End multi-robot coordination algorithms with the aim of reducing the time to completion while maintaining reliable performance.

1.2 Exploration and Mapping with Multi Robot Systems

Robotic exploration and mapping of unknown environments is fundamental for several real-world applications [10]. Including finding survivors in a collapsed building after an earthquake, mapping out a building before entering in military applications, or exploring underwater caves [11]. In such missions, a fleet of robots is inserted into an unknown area, the robots explore the area, and while doing so, collectively create a map of it.

This is essential in critical missions such as hazard detection and search and rescue [12]. These missions share the need to reach full coverage of the explorable space, leaving no parts unexplored [13]. Completeness of the robot-built maps, as well as the speed at which this is accomplished, are major challenges [14]. Completing the expedition as fast as possible is of high importance, delays could even imply human losses, for example in fire detection expeditions.

Regardless of the final application or mission at hand, or even whether the fleet is composed of crawling, flying, or swimming robots, the robots will most likely need to conduct some form of exploration and mapping. Creating a good map depends on many things: the ability for the robots to move well, their ability to sense the obstacles, the reliability of their communication, and the performance of the navigation algorithm that drives the exploration and mapping expedition.

Exploration of post-disaster environments for target detection is a risky, time and resource consuming mission. We envision autonomous fleets of micro-robots to be tremendously beneficial in terms of minimizing exploration time and reducing human exposure to risks [15]. A fleet of micro-robots is defined as a group of at least three robotic entities that cooperate together to achieve a common global goal with limited to zero human operated control. During these operations, micro-robots strategically search the environment to collect the most informative data from their surroundings. Thus, the ability to leverage the collected data to the benefit of other members of the fleet is critical to minimizing search completion time.

For this thesis, we start by conducting a "hands-on survey" of the literature of exploration and mapping in terms of exploration strategy for map building. We therefore develop a simulator specifically for comparing exploration and mapping algorithms, implement what we believe are the most relevant proposals, and compare their performance. We discover that existing efficient proposals only generate complete maps when the fleet is very dense (e.g. hundreds of robots deployed on a medium-sized office floor). We therefore design Atlas, a systematic exploration and mapping algorithm specifically designed for sparse swarms, which creates complete maps even in the extreme case of a single robot.

1.3 Impact of Communication on Exploration and Mapping

While not all multi-robot exploration applications require a central base station or “orchestrator”, many do [16]. In critical missions, typically, robots in a fleet wirelessly communicate information about the environment they are exploring to a central “orchestrator”. Having centralised situational awareness at an orchestrator is often required for the effective supervision of the mission [17], as the rescuers will need this information to conduct the "rescue" part of the mission. The study of this problem is often simplified by assuming that communication between robots is possible between any two locations. However, such a strong assumption does not necessarily reflect real-life conditions where imperfections in the communication channel can impact the performance of the system [18].

Communication is one of the most crucial elements of multi-robot coordination, as the efficiency and performance of the mission depend entirely on the real-time data collected by the robots and the timely exchange of it. To minimize completion time, robots in the fleet must be able to exchange information about the environment reliably with each other with minimum delay. However, in most harsh environments such as those of critical missions, communications can be delayed, disrupted, or even non-existent due to interference from the environment or from a limitation of the robotic platform [19].

The Atlas algorithm is an algorithm which runs at the orchestrator and coordinates the action of individual robots in the fleet as they carry out a mapping expedition. Atlas minimizes the time to fully map an unexplored area, and the number of micro-robots necessary to complete the mapping. While Atlas does not require continuous communication between all the members of the fleet, it assumes ideal communication: there are no packet losses or other limitations regarding communication.

We use Atlas as a representative multi-robot coordination algorithm, to evaluate the impact of network connectivity on mapping algorithms. Specifically, we study the impact of lossy communication on the speed of the mapping by Atlas. We then modify Atlas so it handles communication failures while maintaining the guarantee of mapping completion. We develop and use a discrete-event continuous time simulator that includes realistic communication conditions to evaluate the completion time even for extremely lossy environments.

1.4 Defining Context Awareness

Context is any information that can be used to characterize an entity, its condition, or its surrounding situation, if the information is considered relevant to the

interaction between the entities in a system. An entity can be a person, place, material, object or robot [20]. Context Awareness can be viewed as the ability of systems to intelligently adapt to their changing environment based on the system's understanding of it's context [21]. In other words, it can be seen as a system's relationship with the environment around it based on it's understanding of it's surroundings and it's role in relation to them. Such awareness allows for proactive system adaptation based on context, thereby, enhancing the performance of such systems.

A system that is context aware can change it's behaviour as appropriate when changes happen in the environment, making the system more flexible and less dependent on prior knowledge of the environment. A collaborative context-aware system is a system that comprises of a group of entities, capable of sensing, inferring, and actuating, that communicate in order to achieve a common goal [22]. In this thesis, we look into context aware communications as a means to improve the performance of multi-robot systems in exploration and mapping. Fig 1.1 shows communication awareness in relation to Industry 4.0.

1.5 Context Aware Communications: A Means to Enhance Multi Robot Exploration and Mapping

When exploring communication-restricted environments, it is essential to maintain reliable connectivity for as far as the robots spread out. We therefore want to expand the exploration range as far as possible, while maintaining connectivity between all robots in the fleet and the orchestrator. Defining multiple roles (including communication relays) has shown to be a worthy strategy to address this problem [23].

The majority of research on relay placement in such cases tends to fall into two categories. First, communication-aware relay placement based on initial Received Signal Strength Indicator (RSSI) is used. However, this requires running a full mission prior to the exploration to find the optimal position for the relays to be placed. Second, maintaining a distance (specified prior to the mission) between relays and exploration robots.

While these methods do improve the quality of the communication, they add to the time it takes to complete the mission. Running a separate exploration mission to find the optimal relay position based on building communication models of the area, followed by a mapping mission, adds significant delay between the moment the fleet of robots is placed in the unknown environment and the moment the map is fully built. Maintaining a certain distance between all robots and all relays

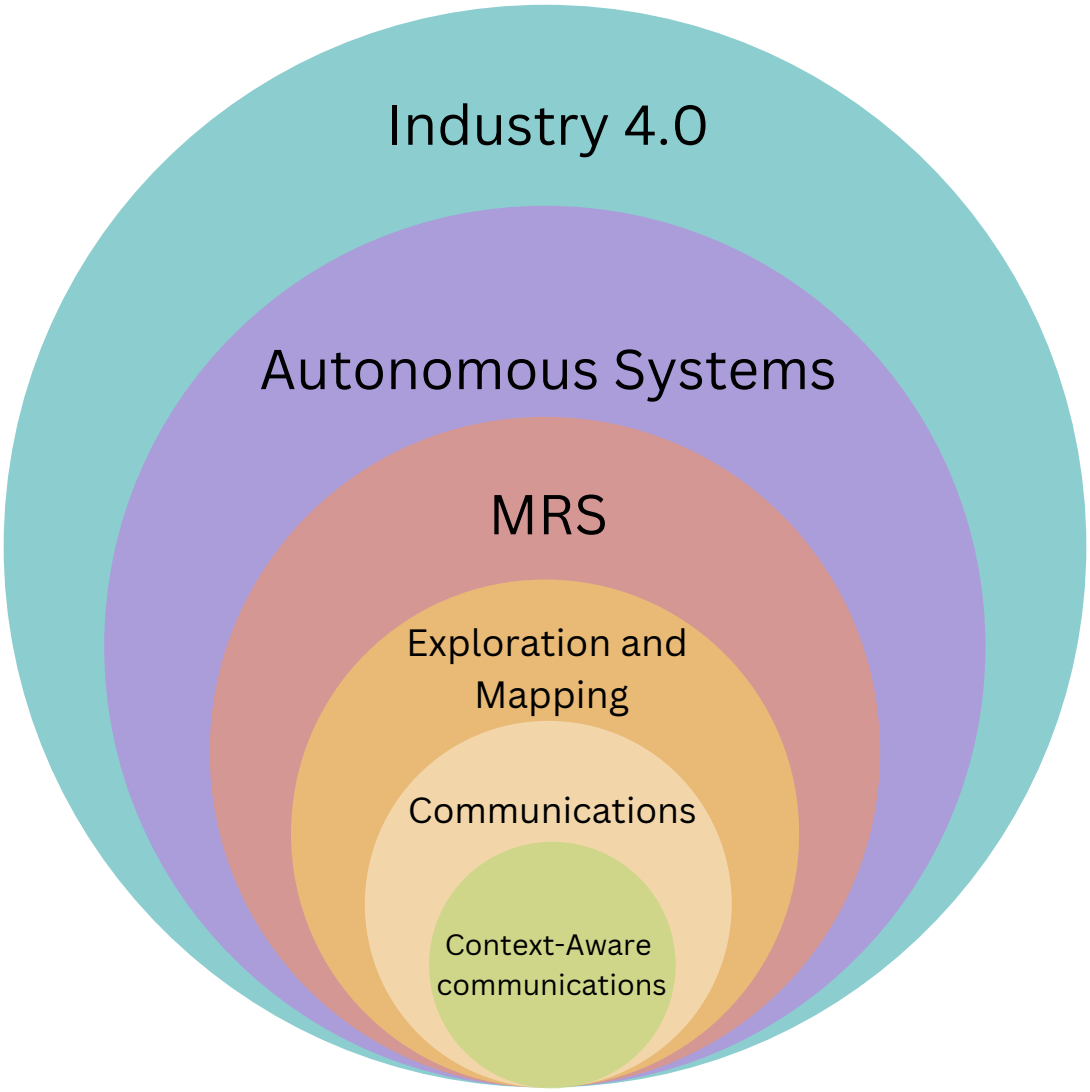


Figure 1.1: Communication-awareness in relation to Industry 4.0.

requires a high number of relays, which may not all necessarily be needed. This reduces the number of exploration robots, leading to a longer time to completion, given that the higher the number of exploration robots, the faster the time to completion. The main issue at hand is reducing time to completion and these methods do not contribute to that.

The research question becomes how can we place relays *(i)* to maintain communications as reliable as possible and *(ii)* dynamically throughout the exploration mission without prior knowledge of the environment, in a way that minimises delay to the exploration and mapping time to completion.

As a solution to this problem, we propose CARA (Connectivity Aware Relay Algorithm), a dynamic context-aware relay placement algorithm that does not require any prior knowledge of the environment yet positions relays based on the estimated quality of the communications.

Communication awareness, or context-aware communication, is when the multi-robot system has the ability to adapt to changes in the quality of communication throughout the mission. Utilising this in relay placement has the benefit of adapting to whatever environment the mission is held-in in real-time to optimise performance, making it more flexible to environmental changes. This leads to placing a more optimal number of relays adapted to what is required to maintain reliable communications throughout the mission. By avoiding placing redundant relays, we maximise the number of exploration relays, hence, reducing the time to completion.

1.6 Organisation of the Thesis

The remainder of this thesis is organised as follows.

In chapter 2 we survey existing research on multi robot coordination specifically for exploration and mapping for critical missions such as search and rescue or hazard localisation/detection. Industry 4.0 requires new approaches in connection, control, and maintenance of robotic systems. We survey state-of-the-art research on robotic systems in Industry 4.0. and highlight the importance of using multiple robots together rather than just one. We then explain the multiple elements, considered in research, for designing multi robot systems which are architecture, coordination, communications and control. We use those elements as a guide to classify related work on multi-robot exploration and mapping algorithms. First,in terms of exploration strategies, then in terms of communication assumptions made when designing exploration and mapping algorithms. We then highlight current trends used for maintaining reliable communications throughout multi-robot expeditions, with a focus on relay placement. We conclude this chapter by summarizing the related work, and listing the key contributions of this thesis.

Chapter 3 introduces in detail both versions of the open-source simulation platform we developed to test exploration and mapping algorithms for multi-robot systems. We explain why we chose simulation as a way to implement and test our algorithms. Followed by a detailed explanation of the first simulator, it's limitations in terms of lack of communication features and how that led to the improved second version. We then provide an explanation of the advantages and new features of the updated version which includes communication and networking features.

Exploration and mapping is a fundamental capability of a swarm of robots: robots enter an unknown area, explore it, and collectively build a map of it. Existing exploration and mapping algorithms tend to either be inefficient, or rely on having a dense swarm of robots. Chapter 4 introduces Atlas, an exploration and mapping algorithm for sparse swarms of robots, which completes a full exploration even in the extreme case of a single robot. We develop an open-source simulator and show that Atlas outperforms the state-of-the-art in terms of exploration speed and completeness of the resulting map.

Such exploration and mapping algorithms require reaching full coverage of the explorable space to build a complete map of the environment. To minimize completion time, robots in the swarm must be able to exchange information about the environment with each other. However, communication between swarm members is often assumed to be perfect, an assumption that does not reflect real-world conditions, where impairments can affect the Packet Delivery Ratio (PDR) of the wireless links. Chapter 5 studies how communication impairments can have a drastic impact on the performance of a robotic swarm. We present Atlas 2.0, an exploration algorithm that natively takes packet loss into account. We simulate the effect of various PDRs on robotic swarm exploration and mapping in three different scenarios. Our results show that the time it takes to complete the mapping mission increases significantly as the PDR decreases: on average, halving the PDR triples the time it takes to complete mapping. We emphasise the importance of considering methods to compensate for the delay caused by lossy communication when designing and implementing algorithms for multi robot coordination.

One of the ways to compensate for lossy communications, expand coverage and maintain reliable communications in multi robot coordination is placing relays. Existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots. These approaches lack flexibility and adaptability to the environment. Chapter 6 introduces the "Connectivity Aware Relay Algorithm" (CARA), a dynamic context-aware relay placement algorithm that does not require any prior knowledge of the environment. We compare CARA against a

state-of-the-art distance based relay placement algorithm. Results demonstrate that CARA outperforms the state-of-the-art algorithm in terms of time to completion by a factor of 10 as it places, on average, half the number of relays.

Chapter 7 concludes this manuscript and discusses the avenues for future work it opens. The list of publications and software contributions made as part of this work are listed in Chapter 8.

Chapter 2

State of the Art

Key Takeaways: In this chapter we survey existing research on multi robot coordination specifically for exploration and mapping for critical missions such as search and rescue or hazard localisation/detection. Industry 4.0 requires new approaches in connection, control, and maintenance of robotic systems. We survey state-of-the-art research on robotic systems in Industry 4.0. and highlight the importance of using multiple robots together rather than just one. We then explain the multiple elements, considered in research, for designing multi robot systems which are architecture, coordination, communications and control. We use those elements as a guide to classify related work on multi-robot exploration and mapping algorithms. First,in terms of exploration strategies, then in terms of communication assumptions made when designing exploration and mapping algorithms. We then highlight current trends used for maintaining reliable communications throughout multi-robot expeditions, with a focus on relay placement. We conclude this chapter by summarizing the related work, and listing the key contributions of this thesis.

2.1 Introduction

In this chapter we survey existing research on multi robot coordination specifically for exploration and mapping for critical missions such as search and rescue or hazard localisation/detection. Section 2.2 Introduces Industry 4.0 and the role of multi-robot systems in it. Section 2.3 describes the multiple elements, considered in research, for designing Multi robot systems. These elements include architecture, coordination, communication and control. Section 2.4 includes a survey of related work on multi-robot exploration and mapping algorithms in terms

of exploration strategies, concluding that frontier-based coordinated algorithms are more efficient and better suited for time sensitive exploration and mapping missions. Section 2.5 shows an overview of various communication assumptions made when designing exploration and mapping algorithms. We highlight the lack of work on algorithms that consider realistic communication assumptions and their impact on the performance of the algorithm. Section 2.6 presents methods for maintaining reliable communications throughout multi-robot expeditions, with a focus on relay placement. We demonstrate how most existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots lacking flexibility and adaptability to the environment. Finally, Section 2.7 summarizes the related work, and lists the key contributions of this thesis.

2.2 Multi Robot Systems in Industry 4.0

Industry 4.0 requires new approaches in connection, control, and maintenance of robotic systems. In this section we survey state-of-the-art research on robotic systems in Industry 4.0.

Industrial robotics play an important role in modern industrial automation technology [24]. Klaus Schwab [25] divided the industrial revolution into four main trends: unmanned vehicles, 3D printing, advanced robotics and new materials. While Russmann et al. [26] identify the main technologies transforming industrial processes as autonomous robots, simulation, horizontal and vertical system integration, the industrial Internet of Things (IIoT), cybersecurity, the cloud, additive manufacturing, augmented reality and big data. The Industry 4.0 technology we focus on in this thesis is autonomous robots due to the wide variety of applications they bring as well as the potential to encompass most of the other technologies adopted in IoT.

There is a wide variety of research on single robot systems for industrial applications. King et al. [27] proposed an online probabilistic motion planning and trajectory estimation navigation technique with collision avoidance for single autonomous space craft. Zhou et al. [28] propose a collision avoidance algorithm for robotic arms to avoid colliding with one another. Chiang et al. [29] and Haghtalab et al. [30] propose algorithms for motion planning in single robots for navigation purposes.

While such applications can be carried out efficiently by a single robot, others require the use of multiple robots working together towards a common goal to complete the mission in a more efficient and effective manner, such as time sensitive missions where robots are used to detect victims or localise hazards.

Using a MRS over a single robot provides many benefits [31], as robots can

work simultaneously on the same mission leading to a faster time to completion. In addition, having multiple robots makes the system more fault tolerant ; if some robots fail, there are still others working on the mission at hand.

2.3 Design Elements for Multi-Robot Systems

The design of algorithms for multi robot systems to complete a specific task, such as exploration and mapping, requires considering several elements, shown in Fig. 2.1. Each of these elements impacts the overall performance of the multi robot system as they impact the relationship between the robotic system and environment it is in. [32]

2.3.1 MRS Architecture

The first design element considered in MRS is the architecture. This refers to whether the decision making part of the system is *centralised* or *decentralised*. Fig. 2.2 illustrates the difference between both architectures.

In decentralised algorithms, each robot is responsible for its own control and makes it's own decisions according to information shared with it by other robots that are nearby. The behavior of the fleet as a whole is "emergent": many local (often simple) interactions between neighbor robots yield the overall behavior. Bio-inspired algorithms are often decentralised, mimicking for example the behavior of ants in a colony.

Mathews et al. [33] propose an algorithm inspired by nature. Their algorithm uses simple local rules based on animal aggregation behaviours such as flocking of birds and schooling of fish to for formation control to maintain coverage and connectivity between robots in a swarm.

Hunt et al. [34] use the concept of dynamic 'boldness' levels from the social behaviour of spiders to explore risky environments in a way that adapts to the size of the group. Here Boldness is represented as a continuous variable associated with the risk appetite of robots to explore regions more distant from a central base.

In centralised algorithms, the overall behavior of the swarm is explicitly driven by a central controller. This controller (e.g. a computer on the side of the area) receives information from the robots (the position of each robots, the partial map each has been able to explore, ...), and remotely controls the movement of each robot.

Matoui et al. [35] propose a centralised architecture for the trajectory planning of a multi-robot system using an artificial potential field method.

Sayed et al. [36] propose a centralised algorithm for a multi-robot system of Hexapod walking robots to map a field hospital environment. A six wheeled mobile

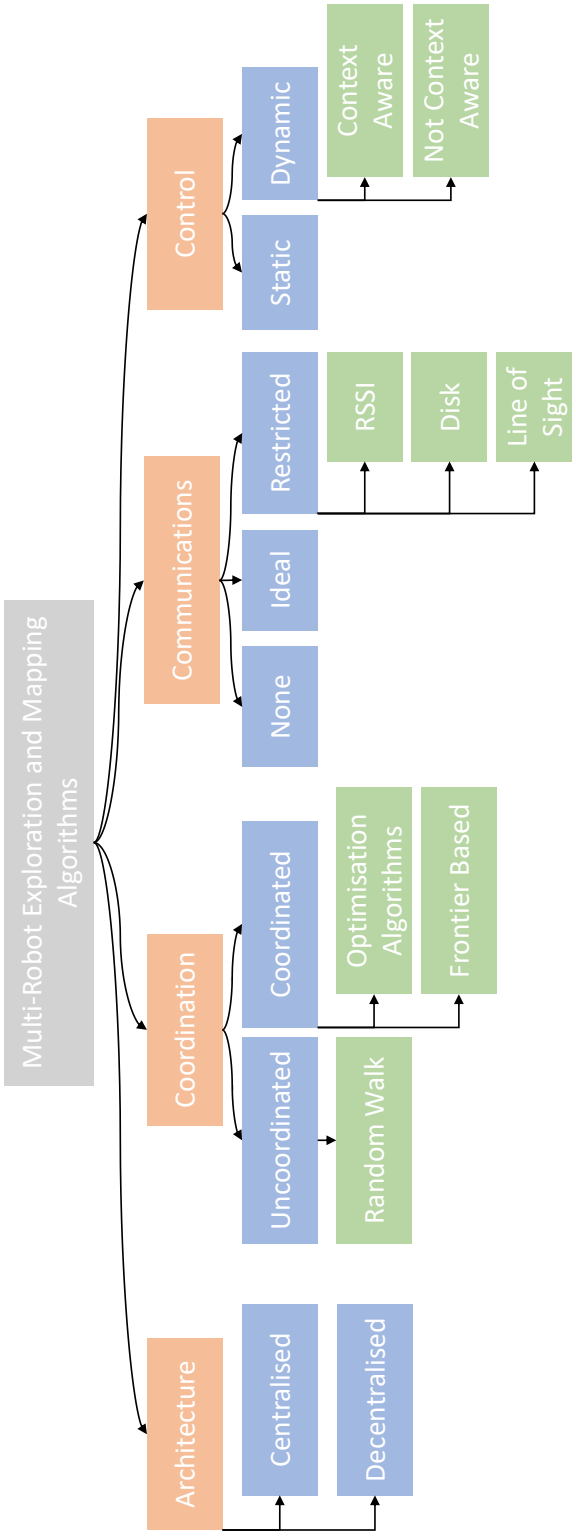


Figure 2.1: Elements considered in multi robot exploration and mapping

robot will then act as a medical cargo delivery that enters based on the predefined map and path.

Despite the fact that centralised systems create a single point of failure, there are cases where this architecture is considered as more efficient in controlling a group of robots [37]. In multi robot architecture, a controller with visibility over the state of all robots in the system can control every robot in a manner that makes the overall behaviour more efficient for specific applications.

2.3.2 MRS Coordination

One of the main elements that impact the performance of a multi robot system, and how efficiently they carry out their mission collectively, is whether or not they are coordinated. In *uncoordinated* algorithms, each robot in the fleet carries out its tasks without coordinating with other robots. This makes for a very algorithmically simple solution, but yields inefficiencies. In the context of exploration and mapping algorithms, if the robots are uncoordinated then multiple robots might for example end up exploring the same area, leading to added delay to the completion of the mission. In *coordinated* algorithms, the task of each robot is planned to explicitly complement that of the others. This involves robots sharing information as they carry out their mission.

Haire et al. [38] propose two algorithms for a swarm of autonomous underwater robots, one coordinated, and the other uncoordinated. They compare both algorithms to evaluate which method performs better. They refer to their uncoordinated algorithm as the lawnmower search. In this method, the robots are evenly distributed along one side of a ship vessel at the waterline. Each robot performs a search that stretches under the vessel until the waterline on the other side of the ship hull is reached. At that point, the robot moves forward along the ship hull and performs the same search under the vessel once more until the original side is reached. This pattern then repeats until the entire hull has been examined. The sweeping search is what the authors call their coordinated algorithm. This algorithm is similar to the the lawnmower algorithm, the only difference is that in this approach the robots are instructed to stay within sensor range of one another while performing their search. Their comparison shows that the coordinated algorithm outperforms the uncoordinated algorithm in terms of accuracy and time to completion.

2.3.3 MRS Communications

The level of coordination of robots in a fleet depends entirely on their ability to communicate information they obtained from the environment to one another.

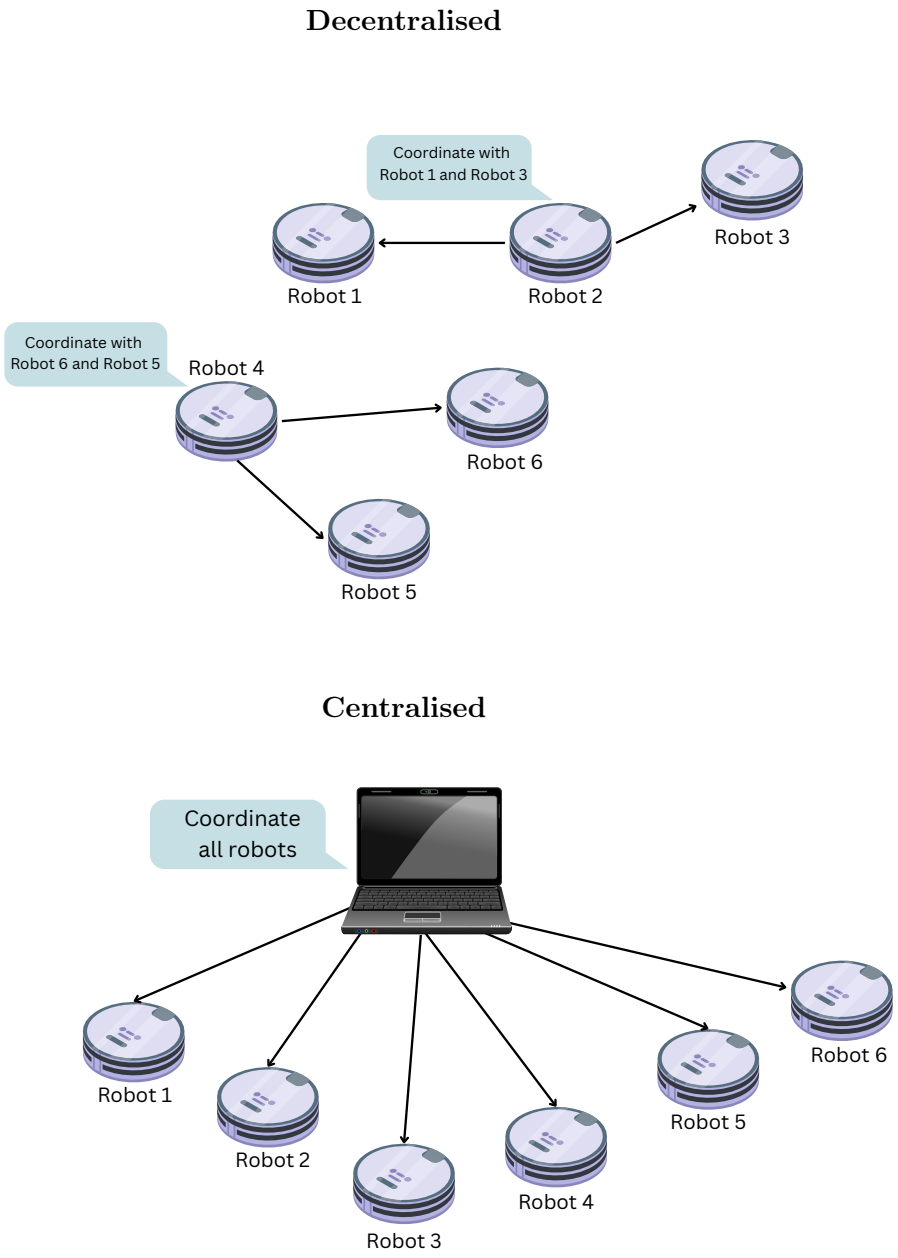


Figure 2.2: A visual explanation of the difference between decentralised and centralised MRS architectures

Amigoni et al. [39] categorize communication for multi robot exploration to three categories: robots are *not* required to communicate, every robot must be able to communicate with all other robots at all times, or communication is only required either periodically or by particular events such as the discovery of new information.

Algorithms that do not require connectivity are usually uncoordinated in terms of behaviour. As for the exploration algorithms that do require coordination, some are designed under the assumption of ideal communications with no losses.

Following are examples of multi-robot algorithms with the assumption of ideal communications. Vielfaure et al. [40] propose a decentralized exploration algorithm that leverages distributed belief maps (DBMs) to maximize coverage and decrease robot failure probability using risk-awareness. While they model communication costs and how it improves scalability, they do not consider lossy communications or how the algorithm would handle a situation where there is no connectivity.

Hvezda et al. [41] propose a novel probabilistic approach for multi robot coordination based on the Rapidly Exploring Random Tree algorithm (RRT) for discrete environments. Here, the authors do not consider or mention communications whatsoever.

Gosrich et al. [42] develop a decentralised control policy for a multi robot system using a Graph Neural Network which uses inter-robot communication to leverage non-local information for control decisions. The authors assume that each robot can communicate with any robot in the region (i.e., the communication radius is equal to or greater than the diameter of the region)

Yue et al. [43] presented an approach for collaborative UAV-UGV mapping in GNSS-denied environments. Lossy communications are not considered in this paper.

Other works consider restricted communications, a more realistic representation of an environment in which a critical mission would be carried out. Following are a few examples.

Smith et al. [44] present a map inference-based coordination algorithm for distributed multi-robot exploration. Exploring robots sample potential positions to explore from the observed and inferred portions of the map and use an internal market to select their target location to move to for exploration. To resolve conflicting goal poses between agents, each agent broadcasts their target position and travel cost in an open auction. The robots keep a record of their recent bids of local robots allowing coordination to continue when robots break communication. This paper considers lossy communications, they test their algorithm with various communication capabilities. From unrestricted global communication between all agents to range restricted line-of-sight communication. The range for line of sight communication was varied from 5 to 100 unit.

2.3.4 MRS Control

The control of a fleet of robots depends on the system's understanding of the environment and level of reactivity to changes in it, this can be either static or dynamic. In static control, prior knowledge of the environment is required. Control policies and rules are then placed based on this knowledge. These policies do not adapt to different environments and lack flexibility. Many exploration and mapping algorithms in the literature are only validated on pre-recorded datasets, and do not consider flexibility problems encountered in practical applications, such as communication delay and packet loss for example [45]. On the other had, dynamic control is a reactive form of coordination. Meaning that conducting the task depends on analysing the information about the environment gained throughout the mission and adjusting tasks accordingly.

Context awareness is a method of dynamic control with higher ambient intelligence. This method of control aims at extracting contextual information and controlling the system in a way that reacts to aspects of the environment, lead by questions of What, Who, Where, When, Why and How [46]. Contextual information can be any information used to characterize the situation a robot is in, including state of the hardware, environment, available resources, quality of the communications, etc [47]. While there is a wide variety of research on context-awareness for smartphones the Internet of Things (IoT) with human users [48]–[50], only a few address context awareness in robotic systems and how the robots themselves would be context aware [51].

2.4 Multi Robot Exploration and Mapping Algorithms

There are several ways to categorize mutli-robot exploration and mapping algorithms. We use the taxonomy, provided in the previous section, to classify existing research on exploration and mapping, with a focus on critical missions, such as hazard detection and search and rescue. In this section, we focus on classifying according to system architecture and coordination, to give an overall understanding of existing methods of exploration and mapping.

There are three main trends in exploration algorithms:

- Swarm optimisation algorithms such as Brain Storm Optimisation (BSO), Particle Swarm Optimisation (PSO), etc
- Random-walk and it's variants
- Frontier-based exploration

Bio-mimicking is often used to design navigation algorithms for swarms, using the movement of groups of insects or bacteria as inspiration. Yang et al. [52] propose a Bacteria Chemotaxis algorithm, which is distributed. Their proposed framework is for target search and trapping using swarm robots. The robots use their initial positions as a local coordinate system. They tessellate the area as a Voronoi diagram, with a robot in each cell. Each robot then explores its Voronoi cell to find targets.

Ghassemi et al. [53] propose a decentralized algorithm that extends Gaussian process modeling to update over trajectories and integrates physical robot constraints and other robots' decisions to perform informative path planning, simultaneously mitigating knowledge uncertainty and getting closer to the source by exchanging information amongst themselves. The authors simulate three case studies to test their algorithm. The first study is a parametric analysis on how the exploitation coefficient of Bayes-Swarm affects its performance, where this parameter is 1 when the robots focus on going straight towards a target and don't explore the environment, and 0 when the robots have no target and are fully in exploration mode. The second study is a scalability analysis to investigate the performance of Bayes-Swarm across multiple swarm sizes. In the third study, Bayes-Swarm is run using chosen default values to analyse its performance regarding different source distributions (i.e. single-modal and multi-modal response surfaces); results are compared with that of standard exhaustive search and random walk methods. The comparison metrics used are the total time taken to complete the exploration and the success rate of completing the exploration. The authors conclude that the Bayes-Swarm performs significantly better than exhaustive search and random-walk approaches in all four case studies.

Li et al. [54] propose a distributed algorithm based on Brain Storm Optimisation (BSO). Robots cooperate using local perception and local communication. Each robot iterates through the following operations: sense the surrounding environment, integrate sensor information in a map representing the environment known so far, detect the frontier of the map, share the information with other robots within communication range, decide the next locations to reach through cooperating with other robots, and move to the selected locations. Through simulation, the authors compare their algorithm against the Nearest Frontier Approach (NFA) for exploration and mapping which is based on selecting the shortest path to the reachable frontiers. For each of the algorithms, two variants are used: one with Euclidean distance as the method of finding the shortest path, and one using the A* algorithm. This results in 4 algorithms being compared against each other. The comparison is done for 3 different environments: home, warehouse, parking lot. For each scenario, Li et al.compare the four algorithms on the following metrics: total time of successful exploration, total

distance of successful exploration, the probability of success, unexplored grids of failed exploration. The comparison results in demonstrating that the authors proposed BSO algorithm outperforms NFA.

Other works such as [55] [56] use BSO in a similar manner for exploration. Similar works that use PSO are [57] [58] [59].

Swarm optimisation algorithms, however, are used mostly for exploration with the purpose of detecting and localising targets. Random-walk and frontier-based exploration are used more commonly for the sake of full area coverage and mapping.

"Random-walk" is a canonical form of uncoordinated exploration algorithms. Huang et al. [60] propose a decentralised uncoordinated algorithm for a swarm of robots to localize chemical leakages or radiation in a factory. The authors define three main stages. Where the robots explore the area via random walk, only changing their direction once an obstacle is detected. If a target is detected the robot stops for a short period of time to investigate the contents of the detected target, and record the relative position of this target.

Kegeleirs et al. [61] compare five flavors of random-walk: Brownian motion, correlated random walk, Lévy walk, Lévy taxis, and ballistic motion. The authors implement all five on a swarm of 10 wheeled mobile robots, let them map out two types of lab environments, and quantify the quality of the maps the swarm generates. They conclude that ballistic motion yields the best maps for the same mapping time as other approaches, mainly because the swarm covers the environment faster. In ballistic motion, a robot moves in a straight line until it detects an obstacle, then changes its direction at random.

Other examples based on Brownian motion are [62] [63]. Examples of exploration and mapping using Lévy walk include [64] [65] [66].

The limitation of random-walk algorithms is that they are uncoordinated and hence take longer to complete as the robots may traverse redundantly through the same area multiple times. Multi-robot coordination tackles this issue by having the exploration and mapping happen in a more systematic way as in frontier-based exploration, such as the algorithms proposed in [67] [68] [69].

Ramaithitima et al. [70] is a very good example of a centralised coordinated frontier-based approach. All robots start at the same starting point inside the yet unexplored area. The central controller (a computer) is located at that starting point. Each robot is equipped with sensors that allow it to distinguish between nearby robots and obstacles. Robots can wirelessly communicate with one another using a short-range radio. The robots form a wireless mesh rooted in the central controller. This means that the central controller receives location and robot/obstacle detection information from each robot, and controls the movement of all robots.

The navigation and mapping algorithm in [70] operates in discrete steps. In

each, the controller instructs some robots to move, the robots move and report information back to the controller. What happens at each step at the controller is as follows. Based on the information received from the robots, the controller builds the partial map discovered so far by the swarm. This is represented internally as a Rips complex , through which the central server identifies the robots that are next to unexplored cells (the "frontier subcomplex") and the robots that are next to obstacles (the "obstacle subcomplex"). Based on a breadth-first search, the central controller identifies the frontier robot to "push away" so as to expand the frontier towards unexplored areas. After that robot has moved, the central controller coordinates with the robots behind it to fill in the void left by the frontier robot moving.

The algorithm presented in [70] results in more systematic exploration, as opposed to random walk. The main downside of this algorithm is that it requires a large number of robots to yield a complete map. If there are not enough robots in the swarm, the frontier is not complete and the resulting map contains unexplored regions.

From this, we conclude that coordinated algorithms appear to be more efficient and better suited for time sensitive exploration and mapping missions. However, coordination requires reliable communications. In the next section we dive deeper into communication assumptions in coordinated exploration and mapping algorithms.

2.5 Communication Assumptions in Coordinated Exploration and Mapping Algorithms

Robotic swarm exploration is often simplified by assuming that communication between robots is always possible between any two locations without packet loss [13]. Here, we survey the related work that *does* consider lossy connectivity.

Manfredi et al. [71] propose an algorithm tolerant to packet loss. The network of robots is composed of one leader and several followers. The goal of the algorithm is to set the position, velocity, and control parameters of the followers in a manner that enables them to follow the leader. Depending on the rate of packet loss, control inputs are corrected to reduce the error. While this algorithm is packet-loss tolerant, it purely depends on maintaining a close distance with other robots, as it assumes the existence of a joint path from the leader to every follower across each uniformly bounded interval.

Benavides et al. [23] also propose an exploration strategy for multi-robot systems. Their approach consists in avoiding disconnection between the robots by having the robots be aware of the connectivity. Given the position of robots and

obstacles, robots estimate the connectivity degree of a specific location. Robots can only confirm the absence of connectivity or deliver an optimistic estimation of connectivity. This is equivalent to either having a 100% Packet Delivery Ratio (PDR) if the robots can connect, or 0% if not. Data losses in between are not taken into account here.

Banfi et al. [72] propose the concept of "recurrent connectivity", where planning is centralised and robots connect back to the central orchestrator only when a new piece of information is available. This eliminates the need for a full communication graph. Packet loss is not directly considered here: robots are assumed to be able to communicate in line-of-sight conditions.

Few research has considered the effect of packet delivery ratio on the overall performance of exploration and mapping. Zhivkov et al. [73] examine the impact of degrading communication quality in a swarm with the aim of quantifying the effects and assessing the risks associated with poor communication quality in robotic swarms. To do so, they conduct a series of experiments with multiple message transmission success rates. They do this using simulation and experimentation. Their simulation results show that the exploration time increases as the packet loss percentage increases. However, the increase in exploration completion time from when no packet loss is not experienced to that of when it is 75%, is only 10 s in simulation, while it is around 40 s in the physical experimentation. While this added time to mapping completion seems insignificant, the paper does not provide any details about the exploration or communication algorithms used, and focuses on random packet loss rates that do not align with real life packet losses. It is therefore difficult to infer from their evaluation the performance of the exploration algorithm, or the severity of the impact of packet losses, in time critical exploration scenarios.

Jensen et al. [13] discuss how communication impacts online multi-robot coverage algorithms' viability for real-world scenarios, and show how communication can affect the performance of various algorithms. However, they focus more on comparing different algorithms with different communication models, as opposed to comparing degrading communications and losses with the same exploration algorithm and communication model.

In conclusion, there is a scarcity of work on multi-robot exploration and mapping that consider realistic communication assumptions, such as packet loss, and their impact on the performance of the algorithm.

2.6 Maintaining Reliable Communications in Exploration and Mapping Algorithms

Most research on how to maintain reliable communications during a multi-robot expedition relies on static control in one way or another. Such solutions are not flexible and reactive to changes in the environment in terms of communication, or require prior knowledge of the environment to some extent.

Common issues with reliable wireless communication in environments that lack pre-existing infrastructure, such as multipath fading, are worsened by domain-specific challenges, like interference, damaged or malfunctioning equipment. Path planning algorithms which respond to changes in communication performance in real time are a promising solution [74]. We survey literature for such algorithms and classify related work based on answering the following questions:

- Are robots assigned as relays prior to, or during, the expedition?
- Is any prior knowledge of the environment needed?
- Is the algorithm reactive and adaptable to the quality of the communications?
- Would this algorithm work for multi-robot expeditions that define as a goal the full coverage of the explorable space?

Nath et al. [75] propose a communication QoS (Quality of Service) aware A* algorithm to choose the path with best RSSI out of multiple possible paths. The A* function has an additional metric to the heuristic which is QoS. Reliable communications is maintained through avoiding paths with poor quality communications and prioritising those with reliable, more stable, communications. However, in exploration and mapping full coverage of the area is essential, hence, we can not afford to avoid certain paths. We require a solution that improves the quality of communications and increases coverage all throughout the explorable area.

Saboia et al. [76] propose another communication-aware algorithm that adapts to changes in the dynamic network by using radio propagation models to predict link quality. A connectivity map of the signal quality over the explored area is maintained throughout the expedition. Droppable radios are used as relays, where exploration robots drop these radios in the positions with the lowest Signal To Noise Ratio (SNR). This limits speed and flexibility as the relays cannot move themselves, requiring other robots to interrupt their exploration to drop the relay radio at the allocated position.

Kim et al. [27] propose a relay positioning algorithm for multi-agent systems in indoor environments. The robots explore the area of interest prior to the mission to build a communication map using Gaussian Process Regression based link quality prediction. A heuristic optimization based on Particle Swarm Optimization (PSO) is used to search for the optimal relay positions. Once the optimal relay positions are determined by the optimisation process, the relay agents are dispatched to those positions. While this algorithm may find optimal positions for placing relays, it requires an entire separate mission for placing the relays, as opposed to placing relays throughout the main expedition itself. The number of relay robots appears to be selected prior to the mission.

Gao et al. [77] propose a relay control algorithm for end-to-end communication for mobile robots with WiFi routers. They model WiFi propagation using Gaussian Process (GP) to optimally position the relays. However, here also, the relay placement algorithm is an entire mission of itself, where it is not clear how the number of relays is allocated beforehand. They also depend on prior knowledge of the environment to assist in building their WiFi Propagation model which lacks flexibility and adaptability.

Arnold et al. [78] use relays to maintain connectivity among a swarm of Autonomous Aerial Vehicles (UAVs). Distance is used as a placement metric. A certain number of UAVs are assigned with the type "Relay" before the mission starts. The role of these Relay UAVs is to maintain a distance equal to half of the maximum range of the WiFi module (approximately 400 meters) from the closest member of the swarm. The number of relays as well as which UAVs have the role of being relays, is assigned prior to the mission.

Varadharajan et al. [79], [80] propose an algorithm that creates a chain of relay robots from a base station to a robot that has lost communication, to "heal" the broken communication in that area. First a root robot is selected then worker robots are selected to carry out the task. Once root and workers are selected, the worker extends the communication chain starting from the root. Then when a worker has determined it is a certain distance away from the root it chooses, a free robot to be a networker robot to act as a relay to the root. When that relay is a certain distance away from the root, it chooses another free robot to be an extra relay to maintain a chain of connectivity. This algorithm provides the ability to place the relays dynamically during the mission, which is more advantageous than running a separate mission for relay placement (it requires less time and resources). The algorithm proposed also assigns various roles to the robots during the mission, where robots go from being idle (free) to becoming relay robots, as opposed to being pre-assigned as relays prior to the mission. For the reasons mentioned above, we believe that this algorithm is a good benchmark to compare against. We will refer to this algorithm as the DBRA (Distance Based Relay

	When are relays assigned	Prior knowledge required	Communication awareness	Suitable for exploration
Nath et al. [75]	No relays placed	✓	✓	✗
Saboia et al. [76]	During exploration	✓	✓	✓
Kim et al. [27]	Prior to exploration	✓	✓	✓
Gao et al. [77]	Prior to exploration	✓	✓	✓
Arnold et al. [78]	Prior to exploration	✗	✗	✓
Varadharajan et al. [79]	During exploration	✗	✗	✓

Table 2.1: A comparison between state-of-the-art algorithms for maintaining reliable communications in multi-robot applications.

Algorithm) algorithm throughout the remainder of this thesis to make it easier to reference.

Table 2.1 summarises the comparison between the different state-of-the-art algorithms for maintaining reliable communications in multi-robot expeditions.

From this, we conclude that most existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots. This lacks flexibility and adaptability to the environment. In addition most works do not consider time to completion in their relay placement algorithms.

2.7 Summary and Contributions

This chapter discusses related work on robots in Industry 4.0 and describes the multiple elements, considered in research, for designing Multi robot systems. As well as a survey of related work on multi-robot exploration and mapping algorithms in terms of exploration strategies, communication assumptions, and methods for maintaining reliable communications throughout the missions, with a focus on relay placement. We conclude that frontier-based coordinated algorithms are more efficient and better suited for time sensitive exploration and mapping missions. We also highlight how there is a lack of work on exploration and mapping algorithms that consider realistic communication assumptions and their impact on the performance of the algorithm. And we end by demonstrating how most existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots lacking flexibility and adaptability to the environment.

This thesis is built upon the related work in this section, bringing the following key contribution:

1. We develop a simulation platform which we use to quantify and compare the performance of Ramaithitima's [70] algorithm (called "Ramaithitima" in the remainder of the thesis) and two variants of random walk to test our

intuition on frontier based coordinated algorithms being better suited for time sensitive applications.

2. We demonstrate that, while efficient with a large number of robots, Ramaithitima does not always result in full maps when using a sparse robot swarm. We therefore design Atlas v1, a centralised exploration and mapping algorithm specifically designed for sparse swarms.

3. Similar to [73], we evaluate the effect of packet loss on the performance of exploration and mapping, however we use an RSSI-based propagation model for modelling the packet loss as opposed to a random model.

4. We develop Atlas v2, which uses an event-based communication protocol where the robots in a swarm communicate with a central orchestrator once triggered by specific events and that is robust to degrading network conditions with 100% completion ratio with PDRs of 0.1 and above.

5. We design CARA, a dynamic relay algorithm that places relays during multi-robot expeditions without prior knowledge of the environment.

6. We compare CARA to a state-of-the-art distance-based relay placement algorithm, demonstrating that connectivity-aware relay placement uses less relays, which in return reduces the time to completion of the multi-robot mission.

Chapter 3

Methodology

Key Takeaways: This chapter introduces in detail both versions of the open-source simulation platform we developed to test exploration and mapping algorithms for multi-robot systems. We explain why we chose simulation as a way to implement and test our algorithms. Followed by a detailed explanation of the first simulator, it's limitations in terms of lack of communication features and how that led to the improved second version. We then provide an explanation of the advantages and new features of the updated version which includes communication and networking features.

3.1 Introduction

Simulation appears as a good method to extract and compare the performance of different exploration and mapping algorithms. It allows for perfect repeatability (the exact same scenario is presented to the different algorithms), resulting in fair comparison. It also allows for repeating experiments easily, and constructing a large enough dataset to present statistically relevant results.

There are several simulation platforms commonly used for (swarm) robotics, or multi robot systems with minimalistic robots in general, Argos [81] and Stage [82] being arguably the most commonly used. These are however general-purpose robotic simulators which embed models for the motors, the battery life, the sensor accuracy, etc. Besides being complex to use, the main danger is for the results on exploration and mapping to be impacted by other considerations. In addition to that these simulators have high computational requirements. We therefore develop a minimalistic simulator and tailor it to our specific research needs relating to multi-robot exploration and mapping, allowing us to add features

as needed throughout the research process. The simulator is open-source to enable other researchers in the community to easily customise and modify to fit their requirements and needs. The first version of the simulator, explained in detail in section 3.2, is discreet, synchronous and does not include any features related to communications. We use this version to compare various exploration and mapping algorithms purely based on exploration strategy. The second version of the simulator, explained in detail in section 3.3, is continuous, asynchronous, and includes communication and networking features. We use this version of the simulator to evaluate and compare the networking capabilities of exploration and mapping algorithms.

Both versions of the simulator are written in Python 3. They are composed of two main elements: the simulator which generates log files, and the Jupyter Notebook-based analysis script which extracts performance indicators from these log files and generates the graphs presented in this thesis. To ensure reproducibility, we follow rigorous software development best practices. In particular, all the source code used in this thesis is part of a release, and bundled together with the instructions to reproduce the log files, and re-generate the graphs. All source code is released under an open-source license*.

3.2 Atlas Simulator Version 1.0

This version of the simulator was designed to test exploration and mapping algorithms for robots with minimalistic capabilities. We use this simulator for conducting the experiments in chapter 4.

3.2.1 Modelling

We represent a 2D area as a discrete number of square cells. A cell is an atomic quantum of space: a single cell can either hold a single robot, be entirely filled by an obstacle, or be entirely empty. As shown in Fig. 3.1, a robot can move to any of the 8 cells in its 1-neighborhood. We call that movement a "step". A robot is not constrained in the direction it moves to. That is, it can move North, then immediate South.

The simulator cuts time into discrete "ticks". At each tick, each robot can move by one step. Multiple (possible all) robots can move during the same tick. The navigation algorithm decides, at each tick, the movement of each of the robots.

We assume each robot is equipped with the necessary sensors to detect the presence of an obstacle or another robot in its 1-neighborhood. These sensors

* As an online addition to this thesis, all the source code used in this thesis is published under a BSD open-source license at `https://github.com/openwsn-berkeley/Atlas`

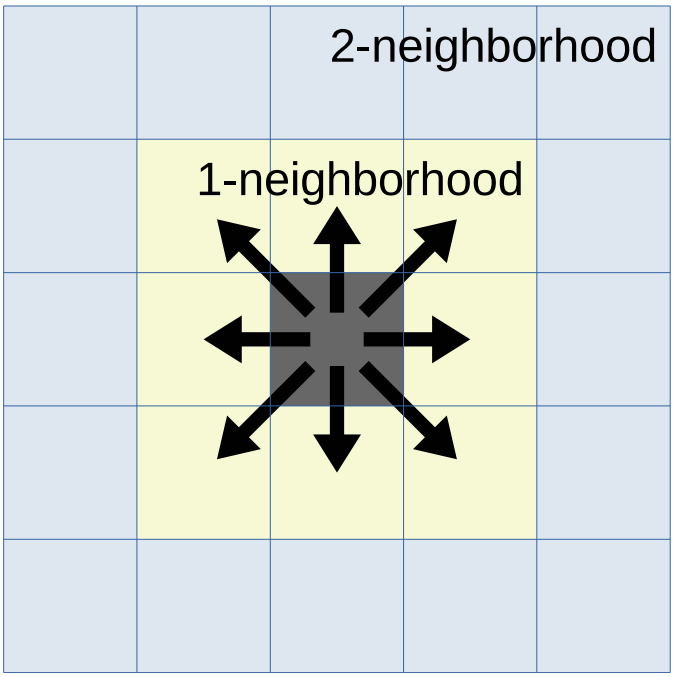


Figure 3.1: We call a robot's "1-neighborhood" the eight cells directly surrounding it. At each tick, the robot can move to any of the cells in its 1-neighborhood. We call "2-neighborhood" the 16 cells directly surrounding the 1-neighborhood.

allow the robot to distinguish between obstacles and robots. The 1-neighborhood of a robot therefore represents the robot's sensing range.

We further assume robots can communicate together, and can communicate back to the starting point where a central controller is located, for centralised protocols.

The movement of all robots here is synchronous. All robots move and stop at the same time in terms of steps per ticks.

3.2.2 User Interface

A simple character-based User Interface (UI) can be activated to see the progress of robots across the area. The user interface is deactivated by default, as it slows down the simulation.

Fig. 3.2 shows a screenshot of the user interface. A dot ("`.`") represents an unexplored cell. A hash ("`#`") represents a discovered obstacle cell. An empty space ("` `") represents a discovered open cell. Character "`S`" represents the start position. Numbers represent the robots.

The UI also shows useful information to the user, such as how many cells have been explored so far, what exploration algorithm is currently running, how many

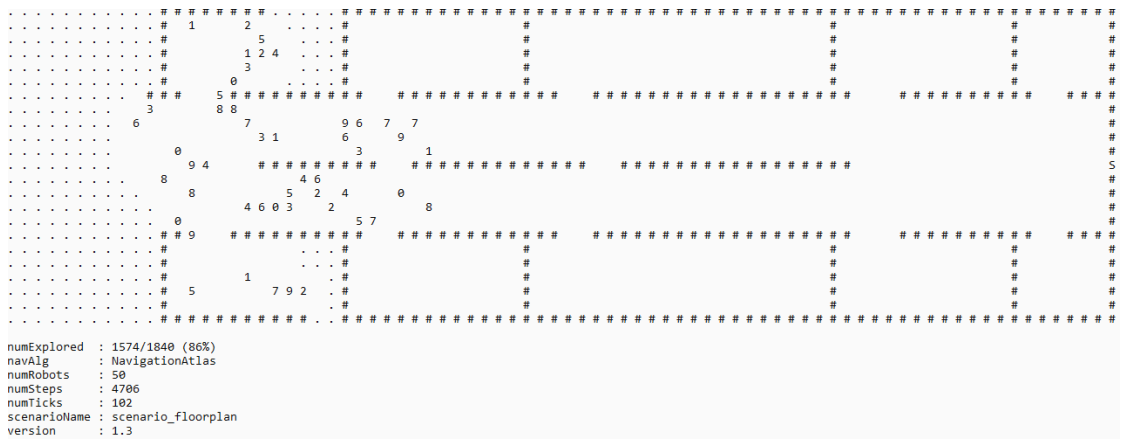


Figure 3.2: The user interface of the initial version of the atlas simulator.

robots are exploring and how many "ticks" and "steps" have passed.

3.2.3 Configurations

The simulator requires certain configuration settings to be given before running an algorithm:

- A scenario: this represents the blueprint of the area the robots are to explore.
- The number of robots that are to conduct the mission.
- The initial position that the robots start exploring from.
- The exploration algorithm to be used.

As shown in Fig 3.3 and summarised in the bullet points above, there are certain configurations the user needs to specify in order to run an exploration and mapping algorithm. There are built in algorithms including random walk, ballistic motion, Ramaithitima (a frontier based exploration and mapping algorithm explained in Chapter 2), and our Atlas Algorithm v1. The user must specify which of these algorithms they want to run. Since the simulator is open-source, the user may add their own algorithms if hey wish. The user must also specify the number of robots they want their simulated fleet to be comprised of. There is no limit on the number of robots from the simulator side, the only limitation is the minimum number of robots the particular algorithm chosen needs to complete the exploration. This is discussed in further detail in Chapter 4. The remaining configurations are the scenario and initial position. In Fig 3.4 we see how a desired blueprint of a scenario is given as an input to the simulator. It is given as a string with A hash (“#”) for

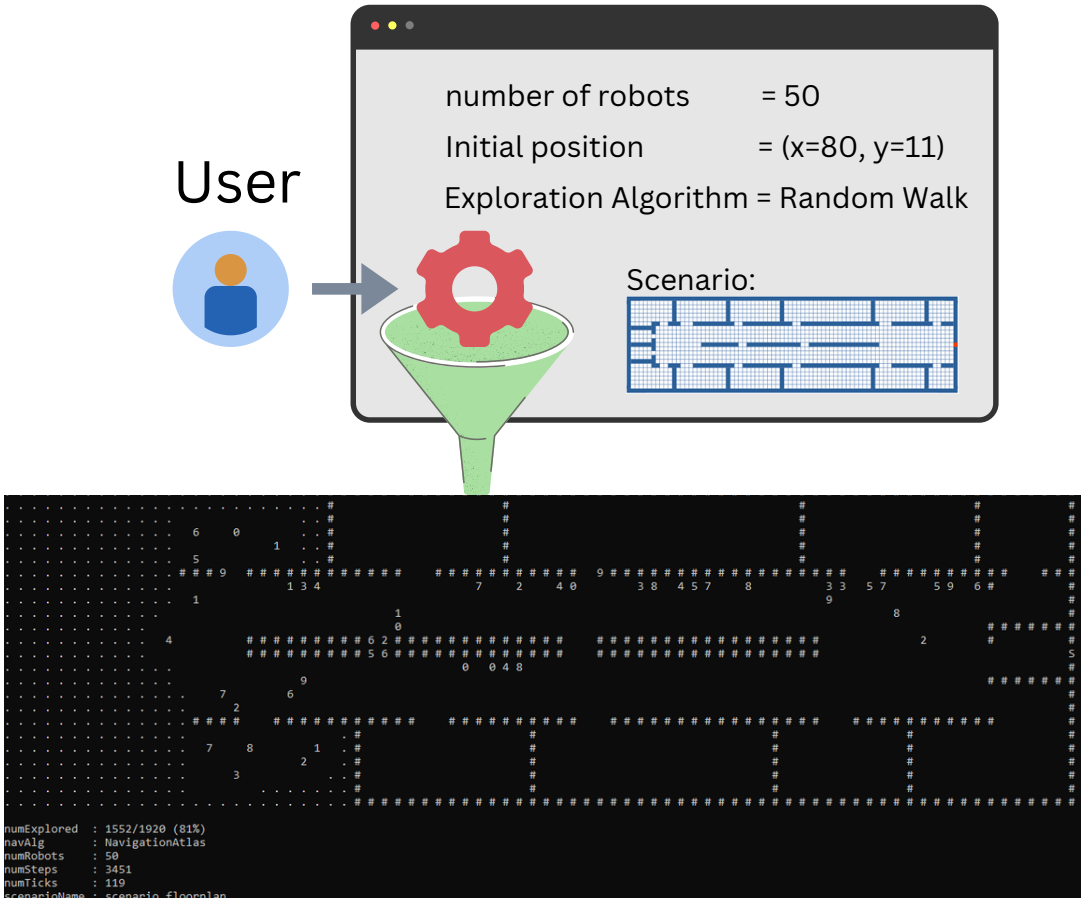


Figure 3.3: A visual explanation of the configurations needed to simulate an exploration and mapping algorithm using the first version of the Atlas simulator

every obstacle and an empty space (“ ”) for any cell in the scenario that is not occupied by an obstacle. An (“`s`”) is used to mark the initial positions of all the robots, where they are to start exploring from.

3.2.4 Limitations

The biggest limitation of this version of the simulator is that it assumes an ideal network interconnecting the robots, yet real wireless networks suffer from limited range, limited capacity, and packet loss. Similarly, it assumes perfect localization and ideal sensors. In a real system, robots don't always know exactly where they are, and are equipped with sensors which might wrongly detect an obstacle. Despite these simplifications, we believe the simulator, as a tool, represents the behavior (location, movement) of a robot well.

The limitations in terms of networking and communications led us to develop an improved version of the simulator that takes such considerations into account. We explain these improvements in the next section.

3.3 Atlas Simulator Version 2.0

This version of the Atlas simulator was built upon the previous version. We improved it for the purpose of investigating the effect of packet loss on event-based communication models. We use this simulator to conduct experiments for chapter 5 and 6. The most significant added features are:

1. Continuous time and space as opposed to breaking downtime into ticks and movements into steps

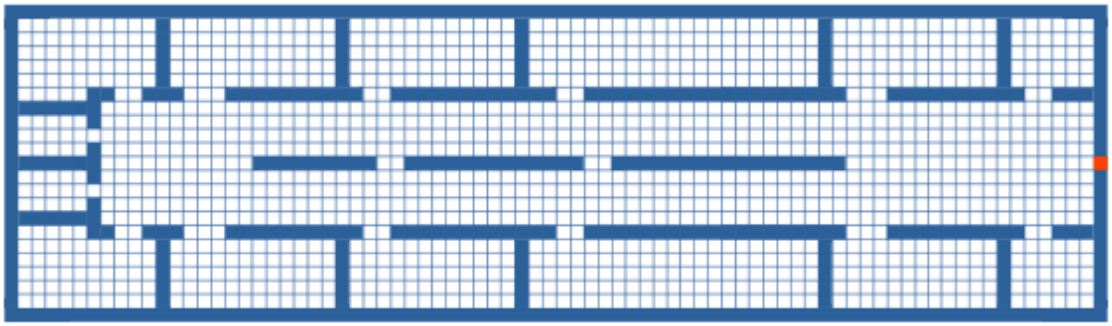
2. Asynchronous as opposed to synchronous: robots update movements when an event occurs regardless of whether an event has happened to the rest of the robots. This allows robots to continue with their movement while the robot concerned gets its own updated instructions

3. Improved User Interface

3.3.1 Modelling

We represent a 2D space in a continuous manner, meaning that the robots can move freely over any distance and in any direction at any angle between 0 and 360 degrees. As for time, the simulator is discrete-event: time updates as scheduled events are processed, leading to a continuous time representation, as opposed to breaking time down into ticks or steps.

Blueprint of a Scenario



Scenario as a Simulator Input

```
scenario_floorplan = '''
################################################################################
#          #           #           #                   #           #      #
#          #           #           #                   #           #      #
#          #           #           #                   #           #      #
#          #           #           #                   #           #      #
#          #           #           #                   #           #      #
#     ##  ######  ############  ###########  ##################  ##########  ###
#######                                                          #     #
#     #                                                                  #
#     #                                                                  #
#                                                                  #######
#               #########  #############  #################           #     #
#######         #########  #############  #################                 S
#     #                                                                  #
#                                                                  #######
#                                                                        #
#######                                                                 #
#     #  #########  ###########  ##########  ################  ###########     #
#          #           #           #                   #           #          #
#          #           #           #                   #           #          #
#          #           #           #                   #           #          #
#          #           #           #                   #           #          #
#          #           #           #                   #           #          #
################################################################################
'''
```

Figure 3.4: An example of a desired blueprint scenario vs how it would be given as input to the simulator. The red square (top figure)/ "s" (bottom figure) represents the initial position the robots start exploring from.

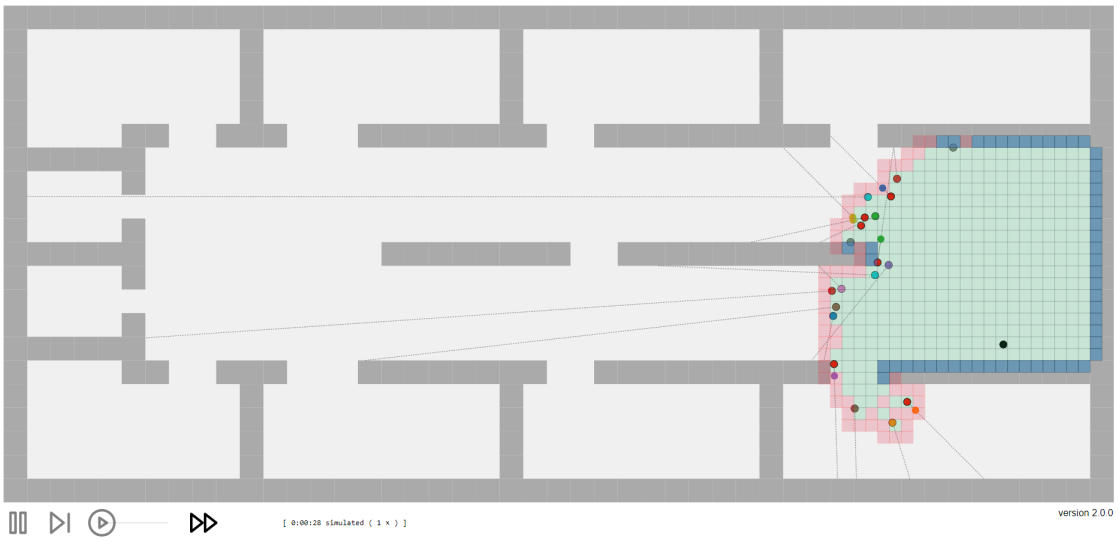


Figure 3.5: The user interface of Atlas simulator version 2.0

We assume each robot has a "bump" sensor that gets triggered whenever it hits an obstacle, as well as transceivers to enable communication between robots and the orchestrator.

3.3.2 User Interface

For this version of the simulator, shown in Fig 3.5, the UI displays the complete scenario/floorplan to help see the percentage of the floorplan explored and mapped as the robots progress throughout the mission. However, The robots themselves do not have this knowledge. It is just for visual representation to aid the user in following the progress of the mission. The robots are represented by coloured circles. The map built is represented by an overlay grid of square cells. Obstacle cells are represented as blue squares. Open cells (obstacle free space) are represented by green cells. For frontier-based algorithms, red cells represent the frontier cells that separate explored and unexplored space. the UI also shows the trajectory of movement fr each robot as a grey line that starts from the robot. The simulation and be speed up through the control buttons at the bottom of the UI. To the right of the speed control panel, we can see the simulation time as the simulation progresses.

3.3.3 Configurations

The simulator requires certain configuration settings to be given before running an algorithm. All configurations must be specified in a TOML configuration file.

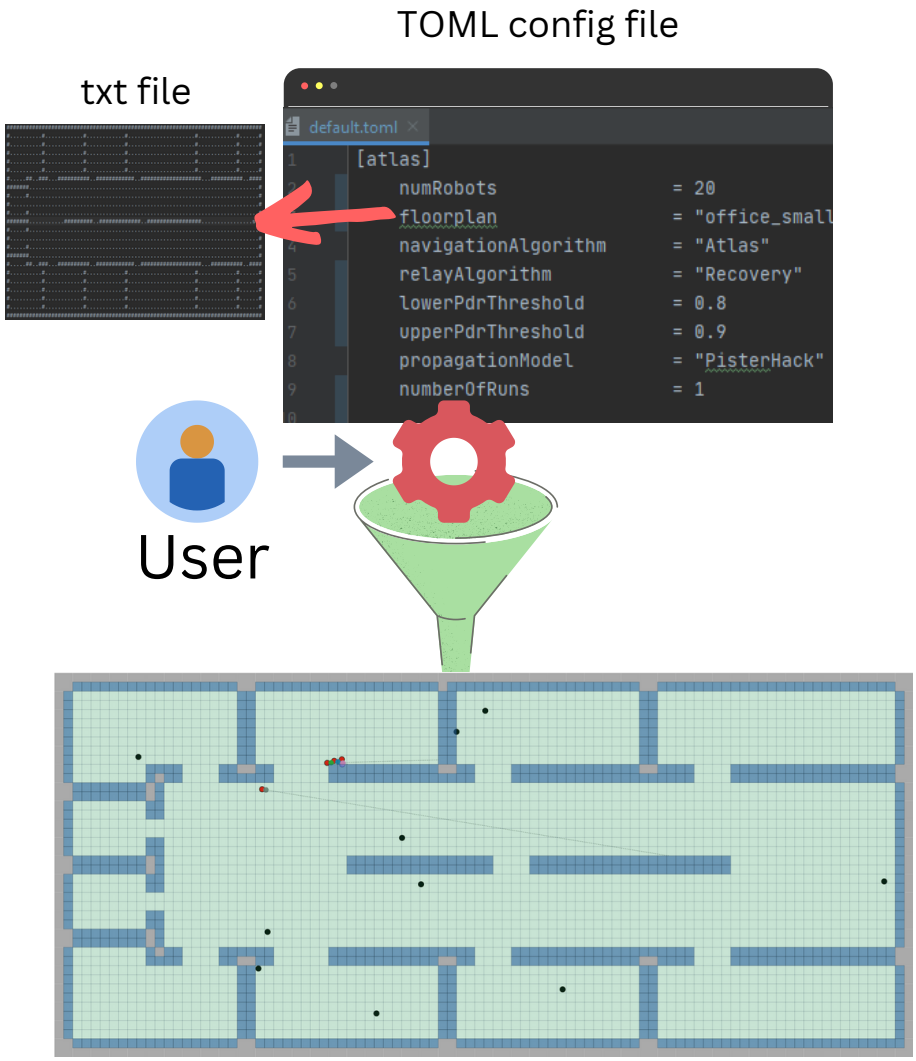


Figure 3.6: A visual explanation of the configurations needed to simulate an exploration and mapping algorithm using the second version of the Atlas simulator

There is a default configuration file used in case the user does not want to specify any settings. The first configuration is a Floorplan, this is the same concept as a scenario from the previous version, however, here we specifically use the term floorplan as it must have a closed border of obstacles (as in a building floorplan) to be valid, whereas with scenarios in the previous version of the simulator, a certain sized boundary was assumed for all cases. Here the floorplan is given as a text file that contains a combination of hashes ("`#`") representing obstacles, an ("`s`") representing the starting point/initial position, and dots (". ") representing empty space. As mentioned, the floorplan must be encapsulated within a boundary of obstacles to be valid. As with the previous version, the number of robots in the fleet, the initial position, and the exploration algorithm to be used can be specified prior to running the simulation.

In this version of the simulator, there are extra configuration related to networking. These include the relay placement algorithm used to maintain connectivity between all robots and the central orchestrator. As well as Packet Delivery Ratio (PDR) thresholds specific to certain algorithms. This is explained in detail in Chapter 6.

3.3.4 Propagation Model

In simulated networks, a propagation model is the relationship between physical distance and Received Signal Strength Indicator (RSSI) or the Packet Delivery Ratio (PDR). The PDR represents the number of packets received out of the total sent. PDR is often heavily influenced by RSSI. We use PDR to represent communication losses and disruptions in our simulator. We refer to the PDR between any two nodes as the link stability between those nodes.

To compute the link stability directly between any two devices, we use the Pister-Hack (experimental randomness) model [84], which is used to obtain the RSSI between the robots and the orchestrator as shown in Equation (3.1). Where P_{tx} is the transmit power in dBm, G_{tx} and G_{rx} are the transmit and receive gains in dB, c is the speed of light in m/s, and D is the distance between the transmitter and the receiver in meters. This RSSI value is then translated to a PDR value based on the work done by Municio et al. [85]: We subtract a uniform variance of [0, −40 dB] from the Friis model equation output and convert the RSSI to link stability values using a conversion table based on real-world deployments. This is shown in Fig. 3.7

$$RSSI(\text{dBm}) = P_{tx} + G_{tx} + G_{rx} + 20log_{10}\frac{c}{4 \cdot \pi \cdot D \cdot 2.4(\text{GHz})} + rand[0, -40](\text{dBm}) \tag{3.1}$$

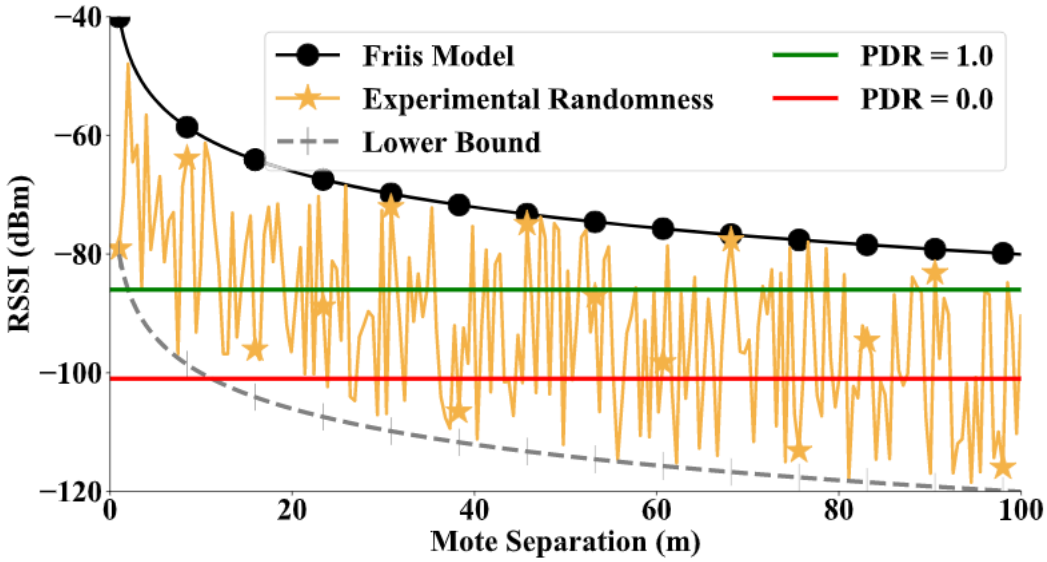


Figure 3.7: This figure was taken from the work of Selden et al. [83]. It shows RSSI values they generated using the Pister-Hack (experimental randomness) model.

3.4 Summary

This chapter presents two versions of the open-source Atlas simulator we developed to test exploration and mapping algorithms for this thesis. The initial version is discrete, synchronous and assumes ideal communications. This version was used to conduct the experiments in chapter 4. The second version of the Atlas simulator is continuous, asynchronous and has networking features such as simulating packet loss through the Pister-Hack (experimental randomness) propagation model. This version of the simulator was used to conduct the experiments in chapters 5 and 6.

Chapter 4

Exploration and Mapping with a Sparse Swarm of Networked IoT Robots

Parts of this chapter were published as part of the following article: *Exploration and Mapping with a Sparse Swarm of Networked IoT Robots*. Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifa, Brian Kilberg, Kris Pister, Thomas Watteyne. **16th International Conference on Distributed Computing in Sensor Systems (DCOSS)** pp. 338-342, May **2020**.

Key Takeaways: Exploration and mapping is a fundamental capability of a swarm of robots: robots enter an unknown area, explore it, and collectively build a map of it. This capability is important regardless of whether the robots are crawling, flying, or swimming. Existing exploration and mapping algorithms tend to either be inefficient, or rely on having a dense swarm of robots. This chapter introduces Atlas, an exploration and mapping algorithm for sparse swarms of robots, which completes a full exploration even in the extreme case of a single robot. We develop an open-source simulator and show that Atlas outperforms the state-of-the-art in terms of exploration speed and completeness of the resulting map.

4.1 Introduction

The goal of this chapter is to provide a "hands-on survey" of the literature of exploration and mapping in terms of exploration strategy. We use version 1 of

the Atlas simulator (explained in Chapter 3) to implement what we believe to be the most relevant proposals, and compare their performance. We discover that existing efficient proposals only generate complete maps when the swarm is very dense (e.g. hundreds of robots deployed on a medium-sized office floor). We therefore design Atlas, a systematic exploration and mapping algorithm specifically designed for sparse swarms, which creates complete maps even in the extreme case of a single robot.

Comparing exploration and mapping algorithms necessarily means extracting some key performance indicators from each. Yan et al. [86] analyzes the performance metrics and lists the following as the most relevant: exploration time, exploration cost, exploration efficiency, map completeness, and map quality. We use these metrics for comparison.

The contributions of this chapter are twofold:

- We design Atlas, an exploration and mapping algorithm for sparse swarms.
- We extract the performance of Atlas, as well as three state-of-the-art algorithms, and present performance results on three representative scenarios.

The remainder of this chapter is organized as follows. Section 4.2 explains the algorithms we compare in this chapter. Section 4.3 details the set-up we use to compare the various algorithms. Section 4.4 shows by simulation that efficient algorithms only work for dense networks. Section 4.5 introduces Atlas, an algorithm specific to sparse swarms. Section 4.6 describes the simulation results. Finally, Section 4.7 summarizes the chapter and discusses avenues for future work.

4.2 The Exploration Algorithms

The algorithms we compare in this chapter are pure random walk, ballistic motion, the Ramaithitima algorithm, proposed by Ramaithitima et al. [70], and the algorithm we developed as an improvement of Ramaithitima, Atlas.

In pure random walk, as shown in 5, each robot moves to a randomly chosen open cell in its 1-neighborhood at each step.

In ballistic random walk, as shown in 10, each robot keeps moving in the same direction until it hits an obstacle or another robot. It then picks another direction at random.

Ramaithitima, shown in 12, is a frontier based exploration algorithm. All robots start at the same starting point inside the yet unexplored area. The central controller (a computer) is located at that starting point. Each robot is equipped with sensors that allow it to sense *(1)* the bearing angle between itself

Algorithm 1: Pure Random Walk for each robot in fleet

```
1  while unexplored cells exist do
2  |  populate map with explored cells; for Every time step do
3  |  |  in random direction;
4  |  end
5  end
```

Algorithm 2: Ballistic Motion for each robot in fleet

```
1  while unexplored cells exist do
2  |  populate map with explored cells; for Every time step do
3  |  |  if bumped into obstacle then
4  |  |  |  move to random free cell in 1-hop neighbourhood;
5  |  |  end
6  |  |  else
7  |  |  |  keep in same direction as last movement;
8  |  |  end
9  |  end
10 end
```

Algorithm 3: Ramaithitima exploration and mapping

```
1  while frontierRobots exist do
2  |  frontierRobots = all robots with unexplored cells in their 2-hop
   |   neighbourhood; for Every time step do
3  |  |  for robot in robots do
4  |  |  |  if robot in frontierRobots then
5  |  |  |  |  move to random unexplored cell in 2-hop neighbourhood;
6  |  |  |  end
7  |  |  |  else
8  |  |  |  |  identify closest frontierRobot to robot; move robot to
   |  |  |  |   random free cell in 1-hop neighbourhood of closest
   |  |  |  |   frontierRobot;
9  |  |  |  end
10 |  |  end
11 |  end
12 end
```

and any robot within sensing range, and *(2)* the bearing angle between itself and any obstacle within sensing range (its sensors distinguish between nearby robots and obstacles). Robots can wirelessly communicate with one another using a short-range radio. The robots form a wireless mesh rooted in the central controller. This means that the central controller receives location and robot/obstacle detection information from each robot, and controls the movement of all robots.

The navigation and mapping algorithm in [70] operates in discrete steps. In each, the controller instructs some robots to move, the robots move and report information back to the controller. What happens at each step at the controller is as follows. Based on the information received from the robots, the controller builds the partial map discovered so far by the swarm. This is represented internally as a Rips complex in which a 0-simplex corresponds to every deployed robot in the environment, a 1-simplex exists between two 0-simplices if the corresponding robots are in each other's sensing range, and a 2-simplex exists for every three robots that can all see each other. Using the constructed Rips complex, the central server identifies the robots that are next to unexplored cells (the "frontier subcomplex") and the robots that are next to obstacles (the "obstacle subcomplex"). Based on a breadth-first search, the central controller identifies the frontier robot to "push away" so as to expand the frontier towards unexplored areas. After that robot has moved, the central controller coordinates with the robots behind it to fill in the void left by the frontier robot moving.

4.3 Simulation

In this section we explain the simulation configurations we used to compare the exploration algorithms fairly.

4.3.1 Scenarios

We define three scenarios to run simulations on. The term "scenario" encompasses both the location of the obstacles in the area being explored, and the location of the starting position. Fig. 4.1 shows the three scenarios, which we call "empty", "canonical" and "floorplan".

We made all three exploration areas the same size (80×21 cells) to be able to directly compare the impact of the position of obstacles on the performance of the exploration and mapping algorithms. In all scenarios, all robots start from the same position.

A scenario goes as follows. All robots are initially at the starting position which serves as a "door" into the exploration area. The goal of the algorithm is to map

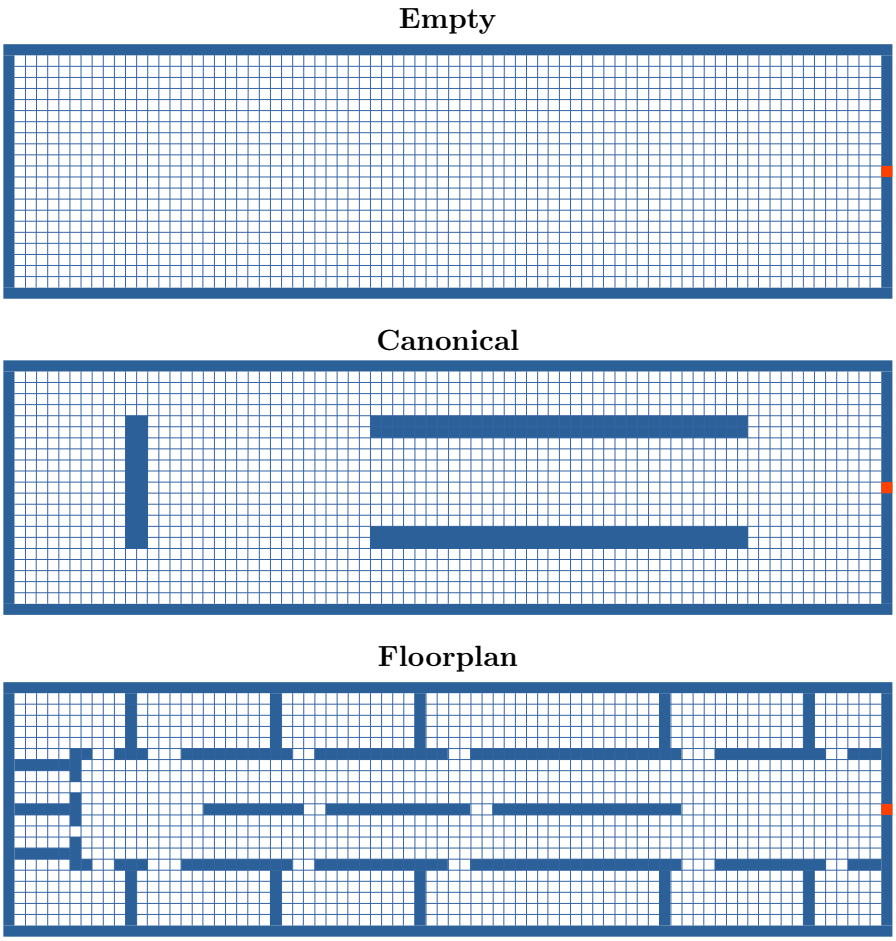


Figure 4.1: The three simulated scenarios. All scenario areas are the same size (80×21 cells). The starting position is depicted as a red cell on the right.

out that space, i.e. find which of the 630 cells are obstacles, and which are not. At the start of a simulation run, the exploration and mapping algorithm knows nothing about the area. As the robots move around in the area, they discover the position of the obstacles by moving next to them, giving the algorithm a more and more complete map. The simulation run ends when either the map completes, or when the navigation algorithm does not trigger any further robot movements. We call "completion ratio" the portion of simulation runs that result in a complete map.

We want a collection of scenarios which trigger diverse behaviors of the navigation algorithm. The "empty" scenario is the simplest one: an empty room. We use it as a reference. The "canonical" scenario is the one used extensively by Ramaithitima et al. [70]. Given that we implement Ramaithitima and compare it against other algorithms, we wanted that comparison to be done in the same conditions as in [70]. Finally, the "floorplan" scenario represents a more complete end-to-end use case, in which a swarm of robots is tasked to map out a floor of an office building. This scenario is modeled after a real office floor at Inria.

4.3.2 Running the Simulation

We end up implementing 4 algorithms, and have 3 scenarios. To be able to compare the impact of the number of robots, we run simulations for a number of robots ranging from 10 to 100, in steps of 10. We call a simulation cycle the resulting $4 \times 3 \times 10 = 120$ simulation runs. We repeat that cycle 145 times. The full simulation time is approx. 24 h, which we split across multiple computers to speed up the simulation campaign.

Because of the random nature of some algorithms, in each of these cycles, the simulation does not execute in the same way. We end up collecting logs for 13033 simulation runs. All results are presented with a 95% confidence interval.

4.4 Limits of Ramaithitima in Sparse Swarms

This section details preliminary simulation results for Ramaithitima. It shows that Ramaithitima does not guarantee full exploration in sparse swarms (a small number of robots), therefore justifying the creation of the Atlas algorithm. Atlas is presented in Section 4.5 and its performance are examined in Section 4.6.

The Ramaithitima algorithm is presented in [70], and summarized in Section 4.2. From an implementation point of view, we implement it as a central controller. At each step of the simulation, that central controller starts by identifying the frontier robots. These are the robots which have at least one unexplored cell in their 2-neighborhood. From that set, it identifies the closest

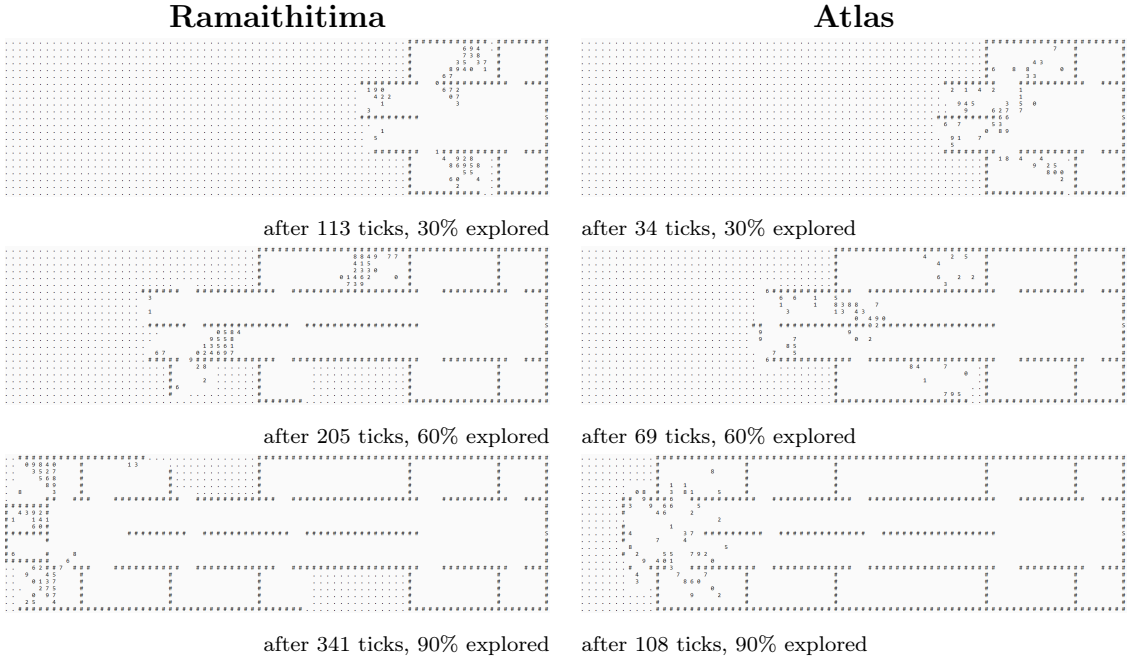


Figure 4.2: Progress of the Ramaithitima and Atlas exploration and mapping algorithms. Using 50 robots in the floorplan scenario.

robot to the start position, and moves it to a cell further from the start position. Rather than use Euclidian distance, the controller uses the Dijkstra algorithm [87] to compute the distance between two cells in the area, i.e. the number of steps a robot would have to take to go from one cell to the other if it took the shortest path. It repeats this process and moves as many frontier robots as possible. It then moves the non-frontier robots so they fill in the voids left by the frontier robots moving. This results in the swarm moving as a pack.

This approach works well when there is a large number of robots, as simulated in [70]. With less robots, the problem is that the frontier robots aren't always side-by-side, so by moving each away from the starting point, it is possible to "forget" to explore an area. This is what is shown in Fig. 4.2: the robots progress from right to left, but pass by 2 rooms that are left unexplored.

To quantify this problem, we plot in Fig. 4.3 the completion ratio of Ramaithitima for a number of robots between 10 and 100. The more robots, the higher the completion ratio, which is expected. Fig. 4.3 also shows that, the more cluttered the area, the lower the probability of creating a complete map. Yet, even with 100 robots, the completion ratio stays below 80%. Worse, regardless of the number or robots, there are always cases, even if rare, in which Ramaithitima does not result in complete exploration.

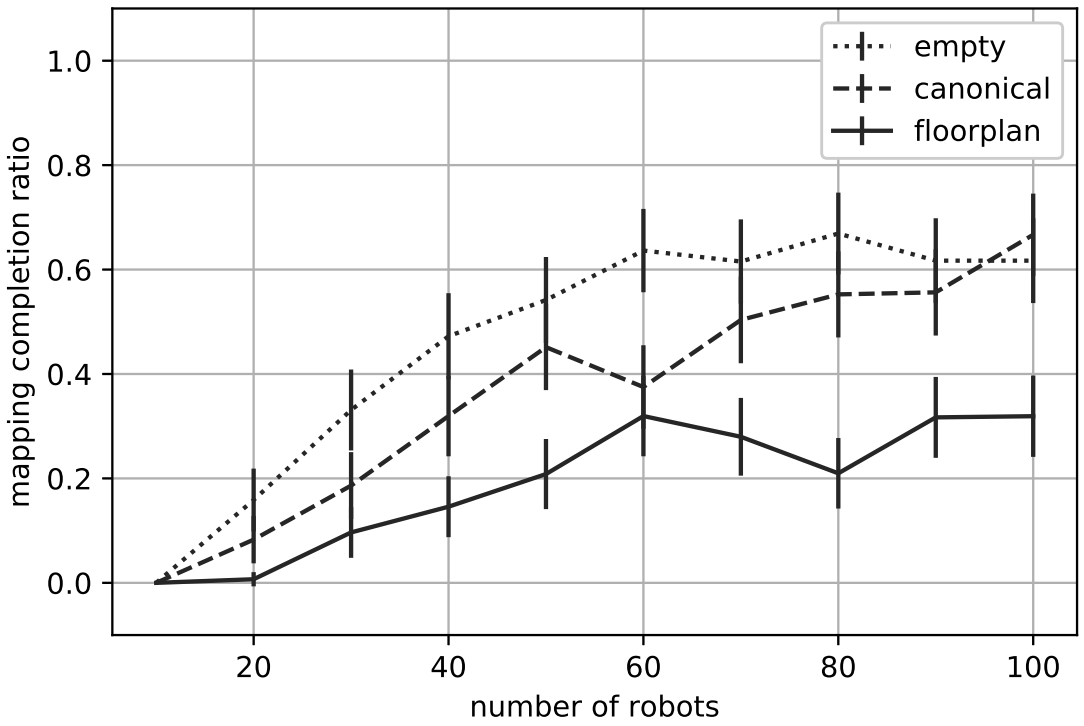


Figure 4.3: Completion ratio of Ramaithitima: the portion of runs where the robots map out the entire area.

All other algorithms evaluated in this paper (including Atlas, our proposal) have a completion ratio of 100% in all cases.

4.5 Atlas

Atlas can be seen as an improvement of Ramaithitima to ensure mapping completion even in the extreme case of having only a single robot. It uses *systematic frontier-based exploration*. Robots are controlled by a central controller which maintains a partial map throughout the exploration and sends robots to explore yet unexplored zones within the area. The main difference with Ramaithitima is that, instead of focusing on the frontier *robots*, it focuses on the frontier *cells*. 5 explains Atlas in algorithmic form.

Algorithm 4: Atlas exploration and mapping

```
1 while frontierCells exist do
2 |   populate map with explored cells; frontierCells = all unexplored 1-hop
  |     neighbour cells of each free explored cell; closestFrontiersToStart =
  |     closest frontier cells to initial position; for robot in robots do
3 |   |   move robot to closest frontier cell to it out of
  |   |     closestFrontiersToStart;
4 |   end
5 end
```

The central controller of Atlas does the following at each step. It starts by identifying the frontier cells, i.e. open cells which have an unexplored cell in their 1-neighborhood. From that set, it keeps only the cells which have the closest distance to the starting point. As Ramaithitima, Atlas uses topological distance (i.e. number steps along the shortest path), *not* Euclidian distance. Once it has the set of frontier cells and the set of robots, it identifies the robot that is closest to any frontier cell, and moves it toward the frontier cell.

The overall behavior is that the frontier expands away from the starting position, and the robots are controlled to "push" the frontier further from the starting point. In the extreme case of a single robot, that robot makes circular movements around the starting point, one step further from it at each revolution. In scenarios where there are many obstacles, the swarm can be cut into subgroups as it navigates around obstacles. The full behavior of Atlas is implemented in 177 lines of code in the simulator.

4.6 Simulation Results

This section presents simulation results and compares the performance of four navigation and mapping algorithms: Ramaithitima, Atlas, and two random walk variants: pure random walk and ballistic random walk.

4.6.1 Heatmaps

Fig. 5.5 allows us to qualitatively understand the behavior of the algorithms by plotting a heatmap of the number of times robots have visited each cell. With random walk, robots tend to hover around the start position, making it a very long process to explore the entire area. With ballistic, robots quickly move about the area, but because the robots are not coordinated, they tend to bounce around the same features over and over. In the Ramaithitima case, we can clearly see that some areas are visited very often, others not; it is the latter that causes Ramaithitima to sometimes "forget" to explore an area. The robot swarm in Atlas progresses from right to left; a small number splits off to explore each of the rooms. Its systematic nature makes the heatmap more homogeneous and symmetrical.

4.6.2 Mapping Profiles

We call mapping profile the plot that shows the number of explored cells as a function of time. It is a good representation to see the overall behavior of the algorithm. Fig. 4.5 shows the mapping profiles of the algorithms for the floorplan scenario.

We clearly see that Random Walk explores rapidly at the very beginning (<100 ticks), then takes a very long time to discover the last unexplored cells. Ballistic, although also uncoordinated, explores the area very fast if there are no obstacles. Its performance significantly degrades in a cluttered area, such as in the floorplan case.

Ramaithitima and Atlas are coordinated, with a central controller which ensures the exploration is done in a systematic way. Both exhibit a mostly constant exploration rate (a straight line in Fig. 4.5). In the floorplan scenario, the non-linearities of the Ramaithitima profile are because of the different rooms being explored, creating "bursts" of explored cells. We also see that the 95% confidence interval widens for Ramaithitima at the end of the exploration, as some explorations complete, others not. We can clearly see the systematic nature of Atlas, which shows a linear mapping profile throughout the exploration. This is because Atlas is designed so robots always move toward non-explored areas.

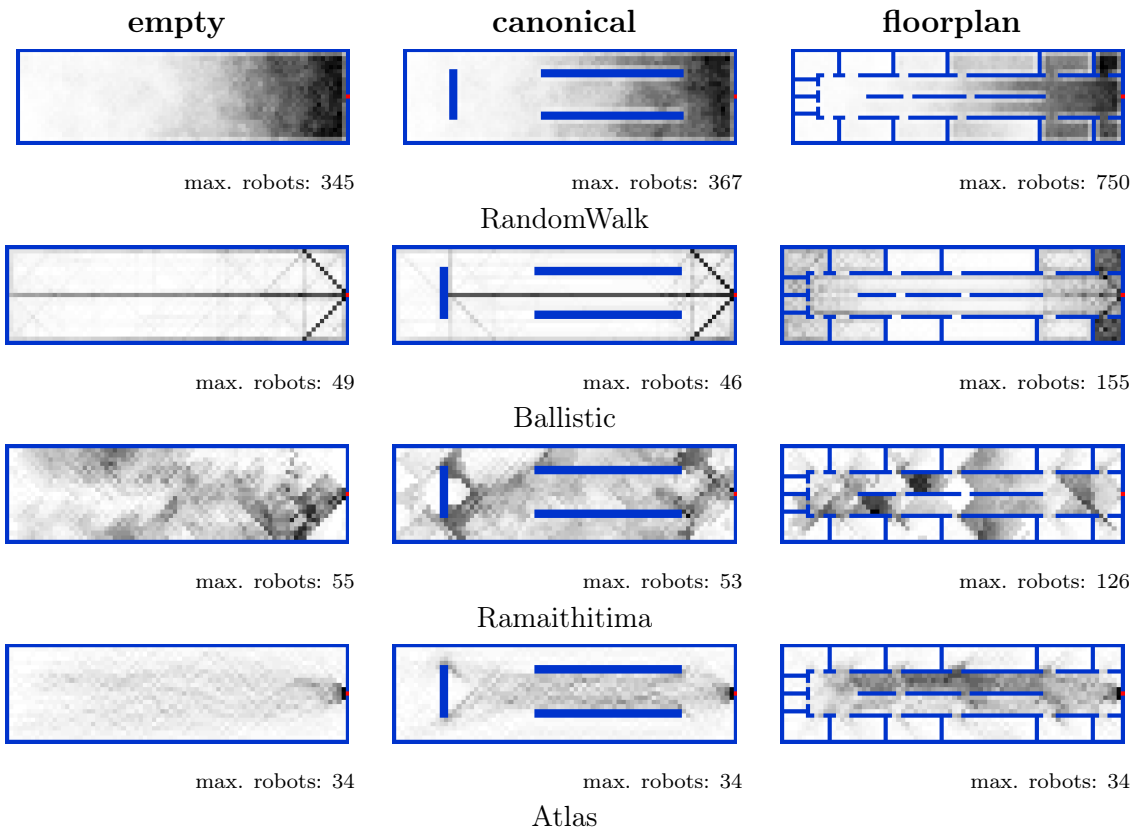


Figure 4.4: Heat maps of how often robots have been present on each cell, at the end of a simulation run. Results presented for a 100-robot swarm. The opacity of each cell is mapped to a different scale for each heat map; the number of robots that have passed by the darkest cell is indicated under each heat map.

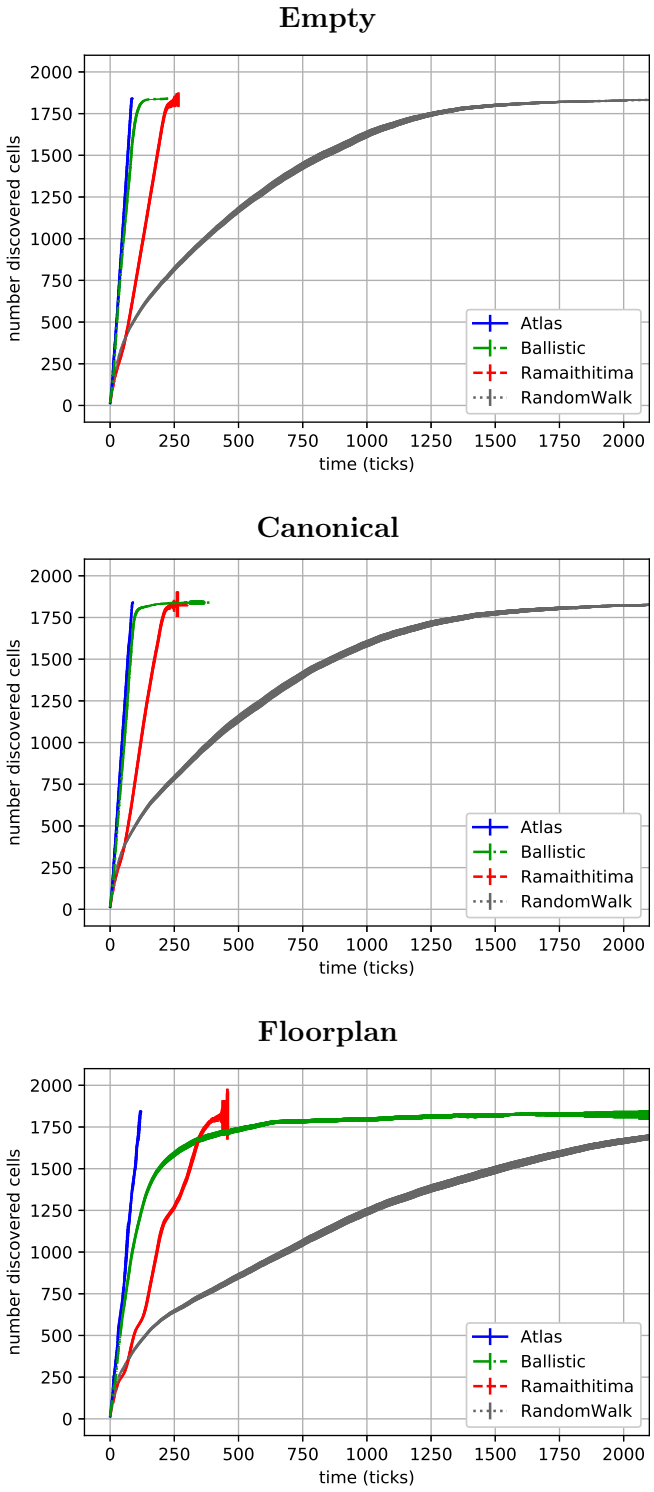


Figure 4.5: Mapping profiles of the algorithms: the number of cells discovered over time.

4.6.3 Mapping Speed

We call "mapping speed" the number of ticks from the moment the first robot enters the area until the moment the area is fully mapped. Fig. 4.6 plots the mapping speed as a function of the number of robots, for all algorithms for the floorplan scenario. For fair comparison, the speed (and associated 95% confidence interval) are presented only for cases where the mapping completes in all runs. No results appear for Ramaithitima as it often does not complete.

In all cases, we see that the mapping is faster with more robots, which is expected. We see that a coordinated algorithm such as Atlas is significantly faster than uncoordinated algorithms (note the log scale on the y-axis). Interestingly, we see that Ballistic performs very poorly when there are many obstacles and few robots, and they tend to enter a repetitive pattern preventing robots from quickly exploring the full area.

4.7 Summary

In this chapter we provide a hands on survey of different exploration and mapping algorithms. We show that existing algorithms tend to be either inefficient, or rely on dense swarms of robots. We develop Atlas, an algorithm that also produces complete maps with sparse swarms. We show by simulation that Atlas outperforms the state-of-the-art in terms of mapping accuracy and mapping speed.

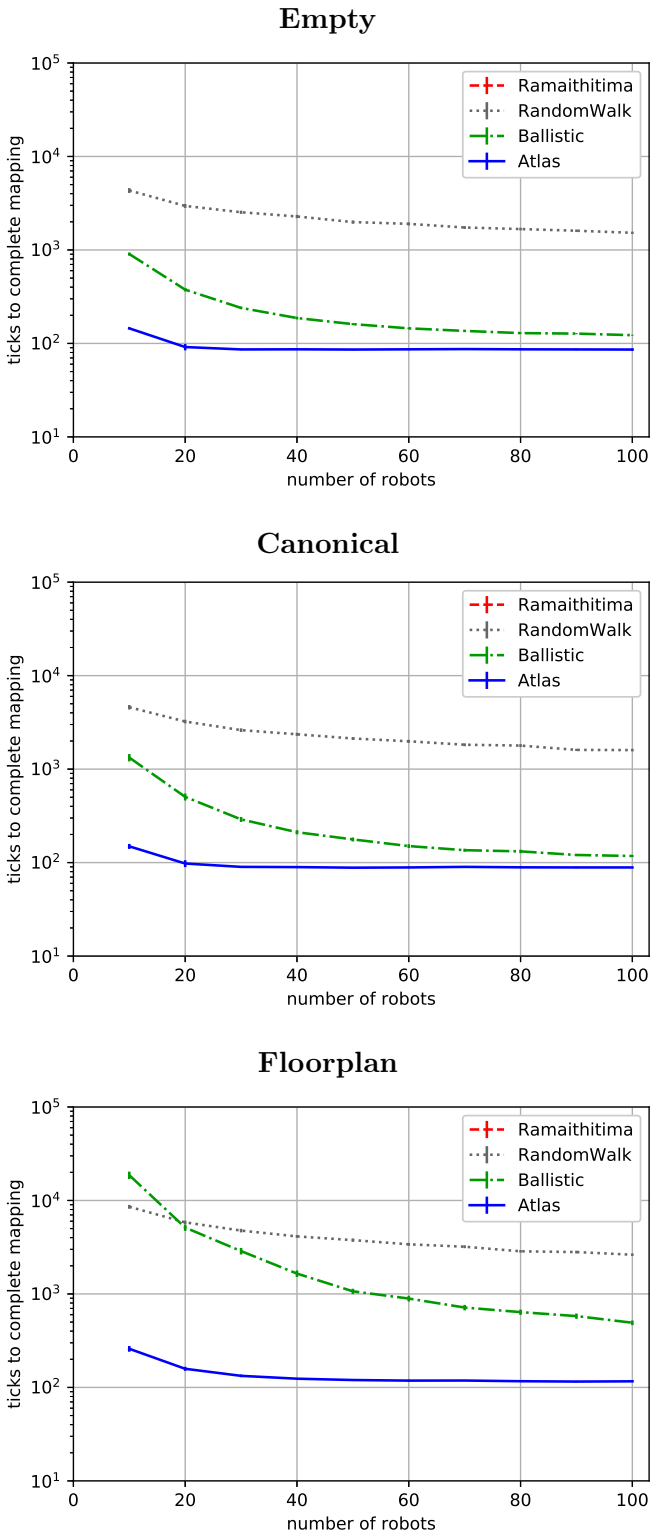


Figure 4.6: Mapping speed: time until the area is fully explored and mapped.

Chapter 5

Coordinating a Swarm of Micro-Robots Under Lossy Communication

Parts of this chapter were published as part of the following article: *Coordinating a Swarm of Micro-Robots Under Lossy Communication.* Razanne Abu-Aisheh, Francesco Bronzino, Lou Salaün, Myriana Rifai, Thomas Watteyne. **Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems**, pp. 635-641, November **2021**.

Key Takeaways: We envision swarms of mm-scale micro-robots to be able to carry out critical missions such as exploration and mapping for hazard detection and search and rescue. These missions share the need to reach full coverage of the explorable space and build a complete map of the environment. To minimize completion time, robots in the swarm must be able to exchange information about the environment with each other. However, communication between swarm members is often assumed to be perfect, an assumption that does not reflect real-world conditions, where impairments can affect the Packet Delivery Ratio (PDR) of the wireless links. This chapter studies how communication impairments can have a drastic impact on the performance of a robotic swarm. We present Atlas 2.0, an exploration algorithm that natively takes packet loss into account. We simulate the effect of various PDRs on robotic swarm exploration and mapping in three different scenarios. Our results show that the time it takes to complete the mapping mission increases significantly as the PDR decreases: on average, halving the PDR triples the time it takes to complete mapping. We emphasise the importance of considering methods to compensate for the delay caused by lossy communication when designing and implementing algorithms for multi robot coordination.

5.1 Introduction

This chapter aims at demonstrating the importance of considering communication disturbances and losses when designing multi robot cooperation algorithms for critical exploration based missions. We improve the Atlas algorithm (Atlas 2.0) by adding an event-based communication protocol where the robots in a fleet communicate with a central orchestrator once triggered by specific events and that is robust to degrading network conditions with 100% completion ratio with PDRs of 0.1 and above. Similar to [73], we evaluate the effect of packet loss on the performance of exploration and mapping, however we use an RSSI-based propagation model for modelling the packet loss as opposed to a random model.

The remainder of this chapter is organized as follows. Section 5.2 presents our system model and the challenge we address in this chapter. Section 5.3 describes the communication model and its implementation in the simulator. Section 5.4 details the modifications made to the Atlas algorithm. Section 5.5.1 showcases the impact of packet loss on the performance of Atlas. Finally, Section 5.6 concludes this chapter.

The contributions of this chapter are twofold:

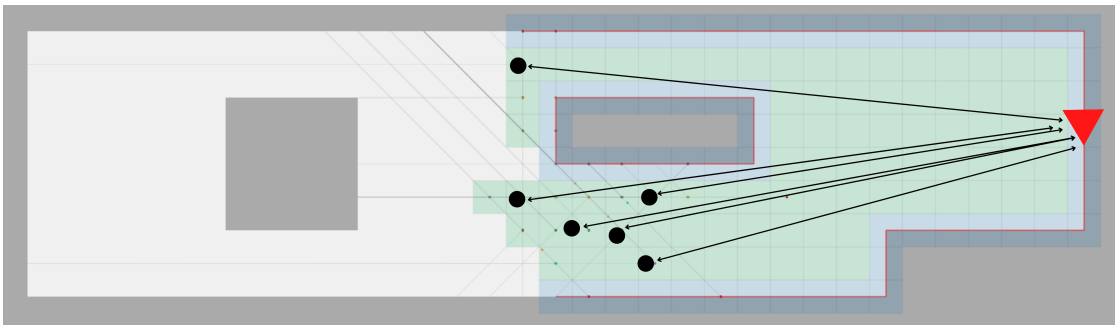


Figure 5.1: The orchestrator and robotic swarm in an environment, showing a partially built map of a previously unknown environment.

- We design a modified version of the Atlas algorithm to include packet loss tolerant exploration and mapping that guarantees a 100% completion ratio.
- We emphasize the need for focusing on communication limitations when designing exploration algorithms by signifying the effect of packet loss on mission time-to-completion.

5.2 System Model and Challenge

Our system model consists of the following elements:

1. **Robots.** We assume each micro-robot is small enough that it can be modeled as a dot with (x,y) coordinates. We also assume that each robot can move at a speed of up to 1 m/s, has sensing capabilities limited to a bump sensor that is triggered upon contact with an obstacle, and the ability to wirelessly communicate. We went for basic robots with minimal capabilities in order to reduce size and cost significantly; making obtaining and maintaining large swarms of robots more feasible.

2. **Orchestrator.** This central entity is responsible for coordinating the exploration by the robots. The centralized nature of the orchestrator enables better exploration strategies based on its global view.

3. **Environment and communication.** The environment is initially unknown to the system. All robots start the exploration from the location of the orchestrator. The robots only report back to the controller when either of two possible events happen: a) a robot's bump sensor is triggered, or b) a robots assigned moving duration timer runs out. In this manner, the swarm behaviour is asynchronous as the orchestrator updates the movement

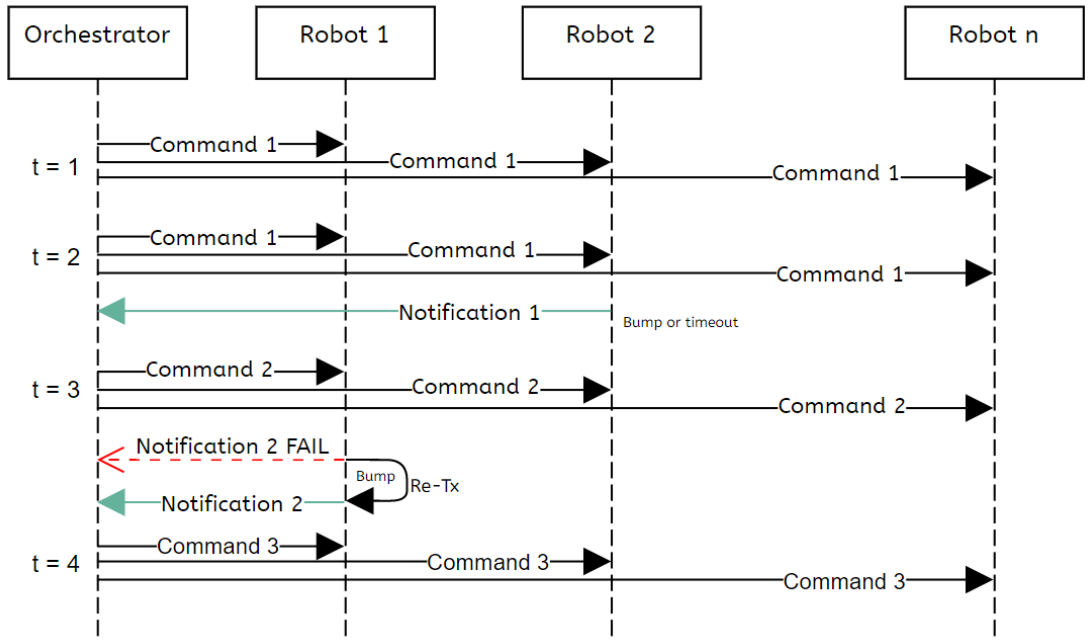


Figure 5.2: Communication between the orchestrator and the swarm.

plan for that particular robot only when it hears back from it. We assume communication limitation by modelling packet loss into the environment in order to represent more realistic losses in a typical environment.

4. **Exploration and mapping.** We use the Atlas algorithm as a starting point, which we modify to be tolerant to packet loss. We refer to this version as "Atlas 2.0". The mapping is represented by dots with (x,y) coordinates on a continuous map, where each dot represents the location at which a robot's bump sensor was triggered. These dots connect into lines once they are a certain distance apart and create an outline of all the obstacles in an environment, see Section 5.4.

We focus on developing and validating a mapping algorithm that reliably completes all of the time, even with packet loss. We evaluate the impact of packet loss on the time it takes to complete the mapping task.

5.3 Communication Protocol

We base our communication protocol on the assumption that it will be used with the IEEE 802.15.4 standard. We see IEEE 802.15.4 as a suitable standard for multi-robot systems due to various factors such as hardware availability,

license-free operation, low power usage, support of mesh networking and low cost.[88]

We design a communication protocol that takes packet losses into account to guarantee mapping completion with any PDR above zero. In the protocol, communication occurs between the orchestrator and each robot in a fleet (and vice-versa) through a star topology, meaning that the robots can not communicate between each other, they can only communicate with the orchestrator. The communication occurs through two types of packets: commands (from the orchestrator to robots) and notifications (from robots to the orchestrator). The protocol is based on an event-based communication model with recurrent connectivity requirements. That is, robots only communicate back to the orchestrator when they have new relevant data or have reached the assigned target position they were directed to go to. Otherwise, no connectivity is needed between the robots and the orchestrator. Time is cut into 1 s cycles, with two steps in each cycle.

Step 1. Commands. The orchestrator transmits a packet. This command packet contains the heading, speed, and movement duration for each robot in the swarm (Fig. 5.3). Headings refer to the direction a robot should take, between 0 and 360 degrees. Movement duration is the amount of time the robot should move for. Note that broadcasting was chosen as a means for communication to avoid the need for complex routing tables when addressing specific robots in large swarms.

Step 2. Notifications. When a command is broadcast, all robots that do receive the command packet check if their instructions of movement have been updated. If not, they ignore the command and continue moving according to their previous instructions. Otherwise, they extract their next heading, movement duration and speed, and start moving. The robots keep moving in the given direction, without the need for connectivity, until an event happens, at which point they stop moving. When an event occurs, a robot transmits a notification packet to the orchestrator. The notification packet contains four pieces of information (Fig. 5.3): the robot ID, a first timestamp with the time at which the robot started moving, a second timestamp with the time at which the robot stopped moving and sent the notification, and whether or not the robot had bumped upon stopping. The logic behind using two timestamps will be explained further on in this section.

As shown in Figs. 5.2 and 5.4, the protocol tolerates packet loss by incorporating the following logic: In terms of commands, the headings for each robot are only updated when the orchestrator receives a notification from the robot. These commands are transmitted periodically every second. Robots that are still moving don't need to "listen" as they haven't bumped or reached their target, and are safe to move in the same direction. The only new information

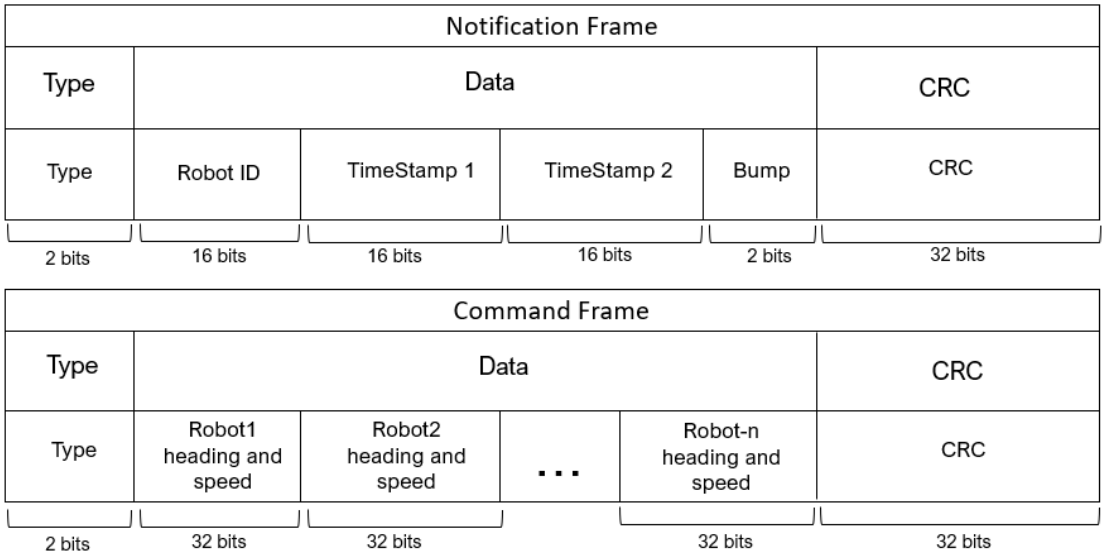


Figure 5.3: Packet frames for commands and notifications

in a command will be relevant to the robots that notified the orchestrator in request for a new heading. As for notifications, when a robot sends a notification, it waits to get a command back from the orchestrator pointing it in a certain direction. If either command or notification packets are lost and the robot doesn't hear back from the orchestrator with a new command, it keeps re-transmitting the notification every second, until it receives a new command. Hence the need for two timestamps. Without packet loss (PDR = 1), one timestamp would suffice as the robot would report the current time as the event time and the orchestrator would receive it on time. Given that the orchestrator already knows the speed and headings of all robots, as well as when and where they last stopped, when it hears back from a robot it takes the stop time and back traces the location of the robot. However, when packets are lost and notifications are re-transmitted, there is a gap in time between the last time the orchestrator recorded an event, when the robot actually started moving after stopping, and when the new event occurred. With two timestamps, the orchestrator knows exactly when the robot started moving from one timestamp, and exactly when it stopped from the other, and can therefore accurately calculate the new position at which the robot stopped. Further details on the mapping are explained in Section 5.4.

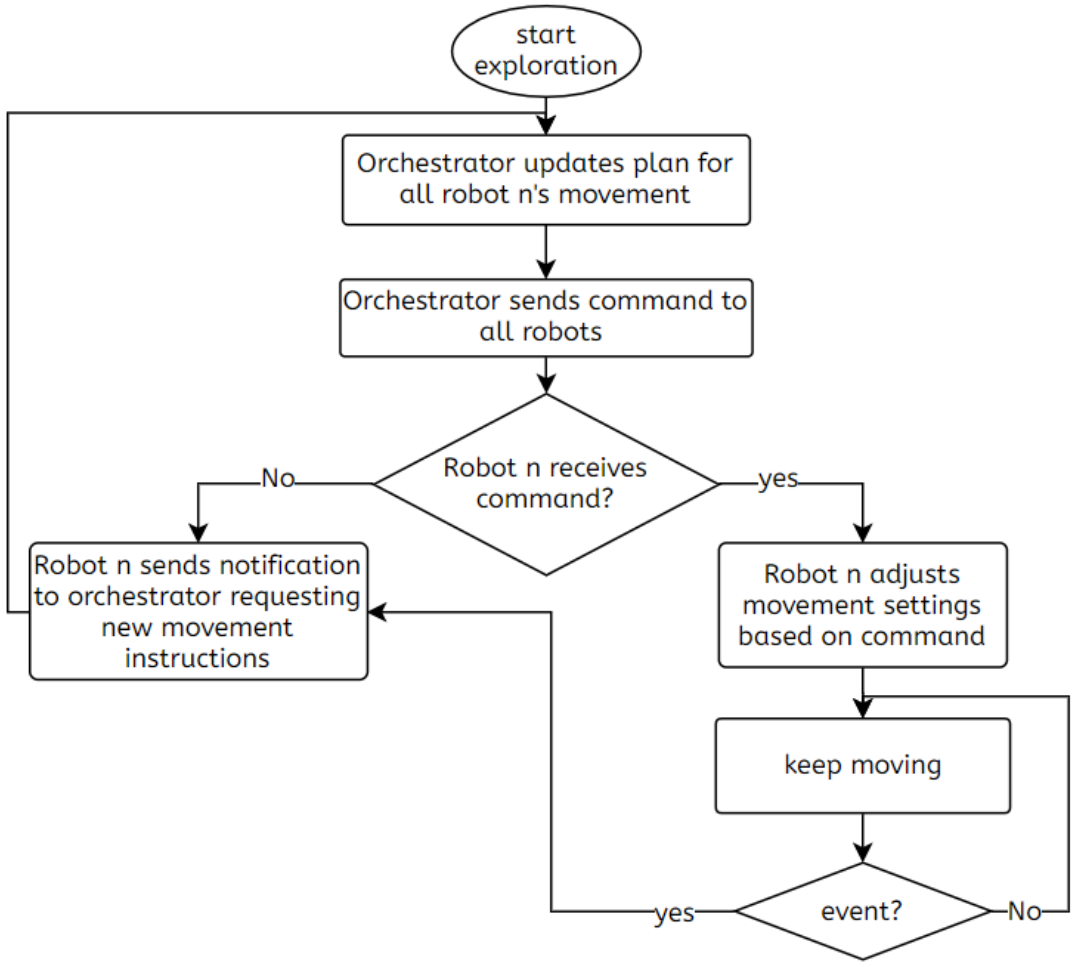


Figure 5.4: A flow chart representing the communication between each robot and the orchestrator

5.4 Exploration and Mapping

5.4.1 Exploration

Atlas is an exploration algorithm, designed for sparse robot swarms. (explained in detail in chapter 4) Because it is centralized, it relies on robust communication between each robot and the "orchestrator". We choose Atlas as a starting point as any packet loss causes the exploration and mapping expedition to fail. It uses frontier-based systematic exploration: robots are controlled by a central orchestrator which maintains a partial map throughout the exploration and sends robots to explore the yet unexplored zones within the area. However, in that version of Atlas, ideal lossless communication is assumed. We modify the previous version of Atlas to make it tolerant to lossy communications; we call that Atlas 2.0.

Atlas is synchronous: all robots start moving at the same time and stop moving at the same time. Atlas 2.0 is asynchronous: each robot receives a new command with movement instructions every time it has an event occur. This is done to reduce the overall time it takes to complete mapping and to reduce the impact of packet loss. If one robot is stuck and hasn't received a new packet, it does not affect the rest of the swarm.

The orchestrator maintains an artificial overlay grid that it builds on the go during the exploration which can be expanded infinitely. Each grid cell belongs to one of the following categories at every point in time:

- *Open Cells (OC)*: containing no obstacles
- *Obstacle Cells (ObC)*: containing obstacles
- *Unexplored Cells (UC)*: cells that have been built during the exploration and navigation process but have not yet been explored.

A robot stops and sends a notification to the orchestrator upon the occurrence of either of two evens: *(1)* it bumped into an obstacle and its bump sensor got triggered, *(2)* it reached the target unexplored cell the orchestrator assigned it, indicated by its movement duration timer running out. Once the orchestrator receives a notification from a robot, it sends new movement instructions to that robot.

If we apply Algorithm 5 starting from the first cell and taking that as the *current cell* for robot n, as an example, the next movement instructions would be set as follows. Given that the robot is already inside that cell, the starting cell is an open cell. Since there are no explored cells yet, the starting cell is considered a frontier cell. Note that, if we have multiple frontier cell options, the one closest to the starting cell will be chosen.

Algorithm 5: Setting new movement instructions in Atlas 2.0

```
1  OCs = empty;
2  ObCs = empty;
3  frontier_cells = empty;
4  Back track all cells traversed by robot;
5  OCs ← traversed_cells ;
6  if Robot bumped then
7  |  ObCs ← current_cell;
8  |  Add "dot" to map at robot position;
9  end
10 if current_cell ∈ OCs & connected to UC then
11 |  frontier_cells ← current_cell;
12 else
13 |  Find closest frontiers to robots;
14 |  Find closest frontier to start point;
15 |  frontier_cells ← selected_frontier;
16 end
17 Choose random UC connected to selected_frontier;
18 Set chosen cell as target;
19 Choose shortest path to target via A* algorithm;
20 Use vectoring to set next heading, speed,movement duration;
21 Update command;
```

All overlay cells directly connected to this frontier – i.e. are within its direct surrounding neighbours without having to pass any other cells to reach it – are valid targets. A random cell out of these is selected as the target for this robot. The A* algorithm [89] is used to find the shortest path to that target. Vectoring is used to set the new movement instruction for the robot. Vectoring is a navigation service provided to aircraft by air traffic control: the controller decides on a particular airfield traffic pattern for the aircraft to fly, the aircraft follows this pattern when the controller instructs the pilot to fly specific headings at appropriate times. In Atlas 2.0, the orchestrator replaces the controller and the robot replaces the plane. The movement pattern is the path generated by A*. A robot moves at the speed and in the heading instructed by the orchestrator until its allocated movement duration runs out, unless it bumps into an obstacle.

The overall behavior is that the frontier expands "away" from the starting position: the robots are controlled to "push" the frontier further from the starting point. In scenarios where there are many obstacles, the swarm can be cut into subgroups as it navigates around obstacles.

5.4.2 Mapping

The map represents an outline of the walls and obstacles in a bounded unknown environment, with the assumption that obstacles can be broken down into square shaped basic elements. These basic elements are referred to as minimum obstacle features. The orchestrator initiates exploration by sending a command containing the headings h, speeds s and movement duration (after which it should stop to change its heading and redirect itself towards the target) for all the robots in the swarm. The orchestrator stores the initial positions of the robots, as well as the headings and speeds sent for each robot per command. Once a robot bumps into an obstacle, it stops moving and reports the time Tm it started moving and the time Tb it bumped back to the orchestrator. It also does this when the movement duration times out and a heading update is due. In this case, however, the packet indicates that no bump occurred, in order to avoid adding data to the map. The orchestrator calculates the position of the robot it just received a notification from, based on (5.1) and (5.2). The orchestrator then updates the next command to include a new heading, speed and movement duration for that robot. It also updates the last known position of this robot.

$$newx = (Tb - Tm) \times \cos h \times s \tag{5.1}$$

$$newy = (Tb - Tm) \times \sin h \times s \tag{5.2}$$

Every bump is stored as a "dot" on the map, representing the (x,y) coordinates of the robot at which its bump sensor was triggered. Any two dots are connected into a line if the distance between them is less than the size of the minimum obstacle size. This is because the two dots are on an obstacle and an obstacle can not be smaller than that size. Any common points on two lines lead to the two lines being connected at that point in the same method. Mapping completion is detected once a line "loop is closed": it has no disconnected edges. The map builder constantly checks all edges to see if they can be connected to one another.

5.5 Experimental Results

We use version 2 of the Atlas simulator (discussed in chapter 3) to evaluate Atlas 2.0. We assume each robot has a "bump" sensor that get triggered whenever it hits an obstacle. The robots are networked by the communication protocol described in Section 5.3. We call PDR the portion of packets sent by a robot that are received by another; PDR$<$1 means there is packet loss.

5.5.1 Results

Our aim is to emphasize the importance of considering more realistic communication while designing swarm robotic cooperation algorithms, by demonstrating the impact communication can have on mission time to completion.

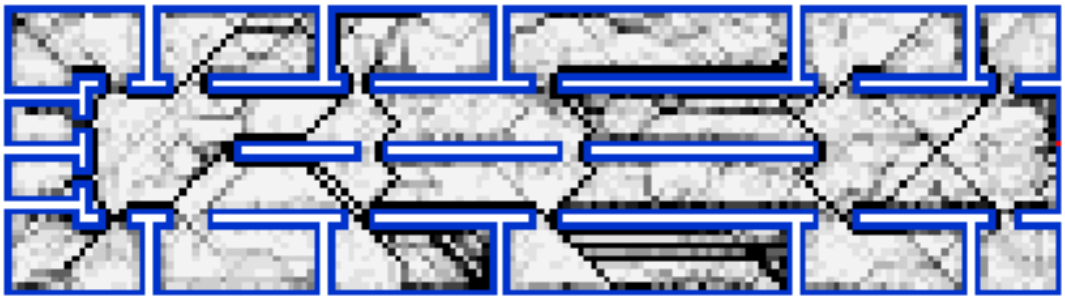
In order to adequately demonstrate the impact of PDR, we first run the simulations with various flat PDR rates across any point in the environment from 0.1 to 1 in steps of 0.1. We run the simulations with a swarm of 50 robots. We then run the simulation with the Pister-hack model which generates different PDRs based on distances between robots and the orchestrator. All results are presented with a 95% confidence interval.

Fig. 5.5 demonstrates the behaviour of the swarm during exploration with various static PDRs. We can see that the behaviour does not vary with packet loss. The paths that seem to be taken more than others are very similar all the way from ideal communication with a PDR of 1 (no packet loss) all the way to a PDR of 0.1 where most packets are lost. This also shows how the modified Atlas algorithm is robust to packet loss, as the overall behaviour and exploration strategy is not affected by packet loss.

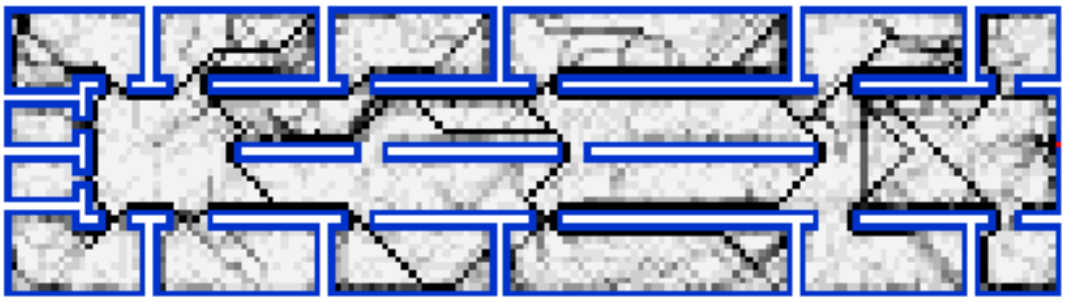
However, as clearly seen in Fig. 5.6, what is significantly impacted by PDR is the exploration time. We can see that as the PDR goes down, the exploration rate goes down with it and the time it takes to complete the mapping increases significantly. In the the floorplan use case we tested, mapping completed in 6.5 min with no packet loss, 19 min with 50% packet loss, 1.85 hours with 90% packet loss. In critical missions such as search and rescue, hazard detection or chemical leakage, this delay in completing the mission could cost lives.

Fig. 5.7 compares the mapping profile with 100% PDR, 10% PDR, and when using the Pister-hack model. With Pister-Hack, PDR gets lower as the distance between the transmitter and the receiver increases. We can therefore see how initially the mapping profile of Pister-hack resembles that of the case with no packet loss. As time passes, the robots get further and further away from the orchestrator (located at the starting point), and hence, the rate of cells explored per second decreases and the mapping gets slower as communication gets lost.

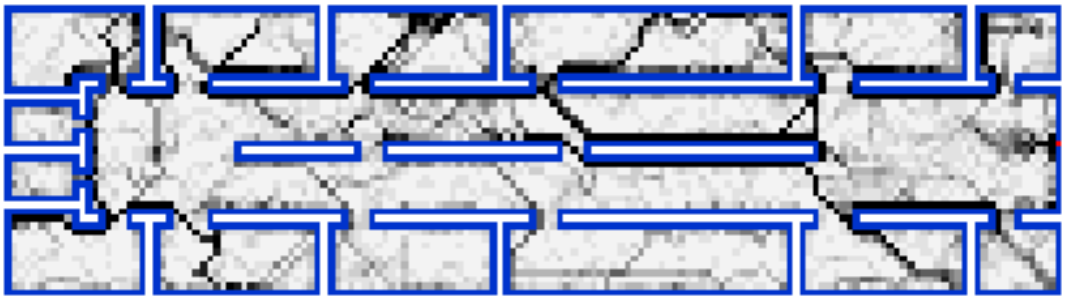
We conclude that the communication quality, the packet delivery ratio, and the communication protocol drastically impact the performance of the swarm. We also deduce that Atlas 2.0 is robust to packet loss; where the overall behaviour and exploration strategy are not affected by packet loss, neither is the accuracy of the map built.



PDR = 0.1



PDR = 0.5



PDR = 1

Figure 5.5: Heat maps of how often robots have been present on each cell, at the end of a simulation run. Results presented for a 50-robot swarm. The darkest cells represent cells that have been passed by 10 or more times. The floorplan used for the simulations had a size of (80×21 cells). The starting position is depicted as a red cell on the right.

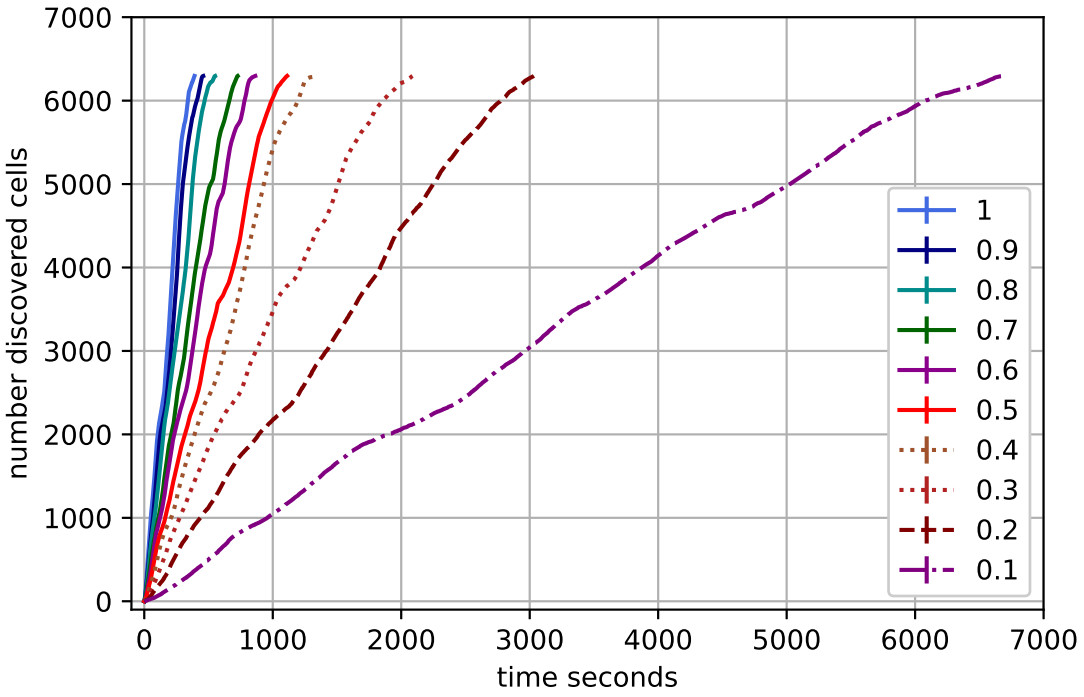


Figure 5.6: Mapping profiles for different PDRs: the number of cells discovered over time

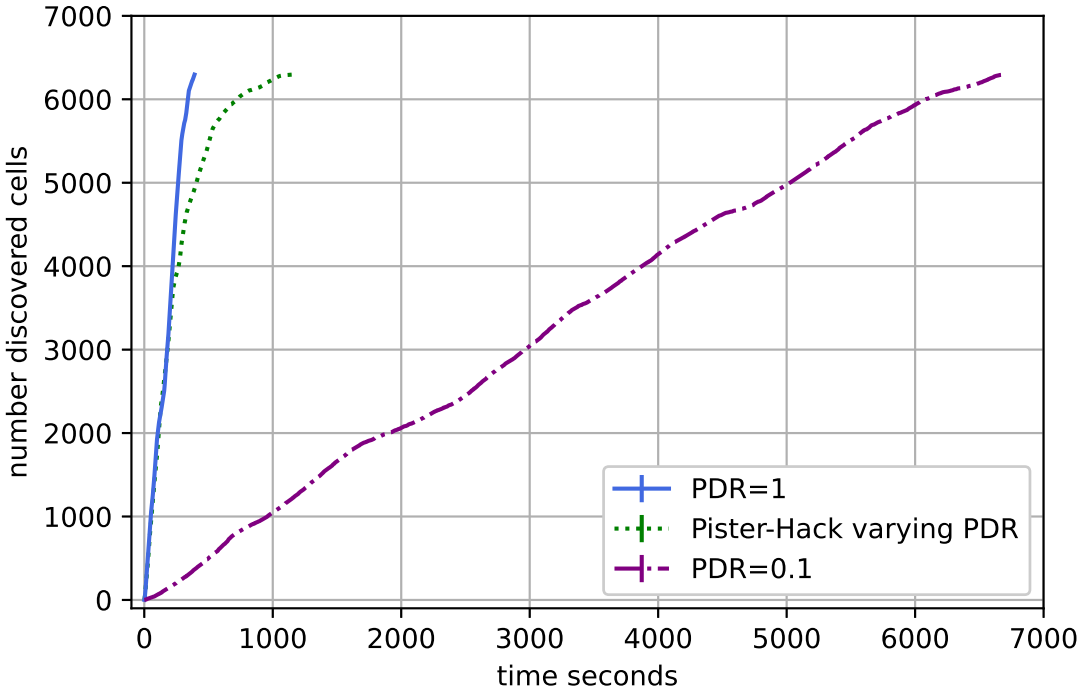


Figure 5.7: Mapping profile with Pister-hack model.

5.6 Summary

This chapter introduces Atlas 2.0; an extension of the Atlas algorithm, which we augment to include packet loss tolerance. The result is an exploration and mapping solution that guarantees a mapping completion ratio of 100% even with lossy communication. We infer that Atlas 2.0 is robust to packet loss; where the overall behaviour and exploration strategy are not affected by packet loss, neither is the accuracy of the map built. We demonstrate the need for focusing on communication limitations when designing exploration algorithms by signifying the effect of packet loss on mapping time to completion. We run various simulation scenarios on a discrete-event, continuous time and space open-source simulator that integrates lossy communication models. We show how, the higher the packet loss, the longer the mapping takes to complete. We therefore stress the importance of considering methods to compensate for the delay caused by lossy communication when designing and implementing algorithms for multi robot exploration.

Chapter 6

CARA: Connectivity-Aware Relay Algorithm for Multi-Robot Expeditions

Parts of this chapter were published as part of the following article: *CARA: Connectivity-Aware Relay Algorithm for Multi-Robot Expeditions*, Razanne Abu-Aisheh, Francesco Bronzino, Lou Salaün,Thomas Watteyne, **MPDI Sensors**, Special Issue on Robotic Systems for Remote and Hazardous Environments **2022**.

Key Takeaways: Exploration of unknown environments is an essential application of multi-robot systems, especially in critical missions such as hazard detection and search and rescue. These missions share the need to reach full coverage of the explorable space in the shortest time possible. To minimize completion time, robots in the fleet must be able to exchange information about the environment reliably with one another. One of the main ways to expand coverage is placing relays. Existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots. These approaches lack flexibility and adaptability to the environment. This chapter introduces the "Connectivity Aware Relay Algorithm" (CARA), a dynamic context-aware relay placement algorithm that does not require any prior knowledge of the environment. We compare CARA against a state-of-the-art distance based relay placement algorithm. Results demonstrate that CARA outperforms the state-of-the-art algorithm in terms of time to completion by a factor of 10 as it places, on average, half the number of relays.

6.1 Introduction

While not all multi-robot exploration applications require a central base station or "orchestrator", many do [16]. Having situational awareness at a central orchestrator is often required for the effective supervision of critical missions [17]. When exploring communication-restricted environments, it is essential to maintain reliable connectivity for as far as the robots spread out. We therefore want to expand the exploration range as far as possible, while maintaining connectivity between all robots in the fleet and the orchestrator. Defining multiple roles (including communication relays) has shown to be a worthy strategy to address this problem [23]. The majority of research on relay placement in such cases tends to fall into two categories. First, RSSI-based communication-aware placement based on running a full mission prior to the exploration to find the optimal position for the relays to be placed. Second, maintaining a distance (specified prior to the mission) between relays and exploration robots. While these methods do improve the quality of the communication, they add to the time it takes to complete the mission. Running a separate exploration mission to find the optimal relay position based on building communication models of the area, followed by a mapping mission, adds significant delay between the moment the fleet of robots is placed in the unknown environment and the moment the map is fully built. Maintaining a

certain distance between all robots and all relays requires a high number of relays, which may not all necessarily be needed. This reduces the number of exploration robots, leading to a longer time to completion, given that the higher the number of exploration robots, the faster the time to completion. The main issue at hand is reducing time to completion and these methods do not contribute to that. The goal of this chapter is to find a solution for the following research question: How can we place relays *(i)* to maintain communications as reliable as possible and *(ii)* dynamically throughout the exploration mission without prior knowledge of the environment, in a way that minimises delay to the exploration and mapping time to completion.

The contributions of the paper are twofold:

- We design CARA, a dynamic relay algorithm that places relays during multi-robot expeditions without prior knowledge of the environment.
- We compare CARA to a state-of-the-art distance-based relay placement algorithm, demonstrating that connectivity-aware relay placement uses less relays, which in return reduces the time to completion of the multi-robot mission.

The remainder of this chapter is organized as follows. Section 6.2 explains the DBRA algorithm, explained in chapter 3, in further detail. Section 6.3 introduces CARA, our proposed context-aware relay placement algorithm. Section 6.4 details the simulation environment and setup used to evaluate CARA. Section 6.5 describes the simulation results. Finally, Section 6.6 summarizes the chapter.

6.2 A Focus on DBRA, the Distance-Based Relay Algorithm

To demonstrate the benefits of the CARA algorithm, we compare against DBRA [79], a state-of-the-art algorithm that uses distance based relay placement. Out of all the distance based work, this algorithm provides the clearest break down as to when, where, and which relays are placed, which makes it the best candidate to replicate and compare against. However, this algorithm was not designed specifically for exploration and mapping. Hence, we made some adjustments to it to make it fit our exploration and mapping target application.

The objective of DBRA is to construct a tree of relays from a central reference location (i.e., the root of the tree) to the robots that are performing a mission (as illustrated in Fig. 6.1), that have lost connectivity, in order to "heal" broken connections. The distance to be maintained between relays and robots is to be pre-set by a human operator. First, a root robot is selected then worker robots

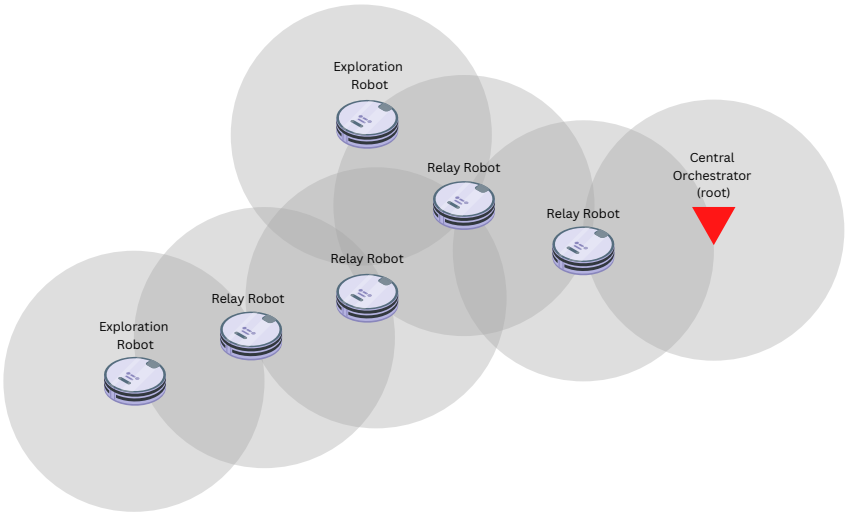


Figure 6.1: Illustration of relay tree chain in DBRA algorithm.

are selected to carry out the task. Next, the worker extends the communication chain starting from the root. When a worker has determined it is beyond the set distance away from the root, it chooses a free robot to act as a relay to the root. When that relay is a certain distance away from the root, it chooses another free robot to be an extra relay to maintain a chain of connectivity, and so on.

To adjust the DBRA algorithm to work for exploration and mapping algorithms by making the following adjustments:

- We set the root of the tree chain to be the central orchestrator.
- There are multiple worker robots rather than just one. We consider each exploration robot to be a task robot that needs to maintain connectivity with the orchestrator.
- We have no "free" robots: each robot in the fleet is an exploration robot until it is assigned as a relay when required.
- The original algorithm did not consider obstacles. We adjust the algorithm to take obstacles into account when building the relay chain, to avoid sending a relay to an unreachable position.

Note that these changes do not affect the overal behavior of the relay placement algorithm, they allow for a fair "apples-to-apples" comparison with CARA.

The general behaviour of DBRA goes as follows. A distance range is set prior to the mission based on the communications range. This distance can be seen as the radius of a disk around each relaying device, including the orchestrator. All robots start exploring from the location of the orchestrator. Once any robot goes beyond the disk around the orchestrator, a random robot is selected to become a relay and is placed at the boundary of the disk around the orchestrator. The communication area now becomes the union of the disk around the orchestrator and the disk around the relay robot. Once a robot exits that communication area, another relay is placed in the same manner. We end up with a connectivity tree starting from the orchestrator at the root, ending with exploration robots at the edges, with relay robots in between.

A limitation of this algorithm is that is does not consider obstacles when choosing the relay position. To address this, we add a mechanism to adjust the relay position if the computed position ends up on an obstacle. A Breadth First Search is ran starting from the chosen position, until the closest explored cell with no obstacles is found. That cell is selected as the relay position and the relay moves there instead, hence, avoiding obstacles.

6.3 CARA: Connectivity-Aware Relay Algorithm

The aim of CARA is to develop a relay placement algorithm that can be used along with any multi-robot exploration algorithm, to place relays dynamically throughout the mission, as part of the exploration expedition. The main goal is to minimise the time it takes to complete a multi-robot expedition successfully through the following design choices: *(i)* when to place a relay? *(ii)* which robot to select as a relay? *(iii)* where to place that relay? These three choices are essential as they affect both the quality of the communication and the speed of completion.

Two elements that impact time to completion the most in centralised exploration missions, are *(i)* the number of robots exploring; *(ii)* the quality of communication between the central orchestrator and all the exploration robots. In terms of relay placement, this would translate into the following main requirements: *(i)* minimising the number of relay robots that switch from exploring the area, becoming relays that are stationed in specific positions; *(ii)* maintaining high quality reliable communications measured by a certain metric such as RSSI or Packet Delivery Ratio (PDR).

Our intuition is that the key to minimising the number of relays is reactivity to the quality of communications in the environment, hence, only placing relays when needed. By avoiding placing redundant relays, we maximise the number of exploration robots, thereby reducing the time to completion. This is referred to as "communication awareness": when a multi-robot system has the ability to adapt

to changes in the quality of communication throughout the mission.

CARA is a dynamic communication-aware algorithm that does not require any prior knowledge of the environment. It is designed to maintain reliable communication during multi-robot exploration and mapping expeditions to minimise the time to completion. The algorithm dynamically stabilises the communications by keeping the PDR between the orchestrator and robots above a certain threshold throughout the mission. In order to do so, the following is required:

- *lowerthreshold*: as soon as the PDR of one or more robots goes below this threshold a new relay must be placed.
- *upperthreshold*: when the position of the new relay is being chosen, the PDR between the relay and the orchestrator at that position must be above or equal to this threshold.
- $CommunicationsHistoryMap$: a map that contains the estimated PDR history of every robot in every position it has traversed. This map is updated dynamically throughout the mission

The General behaviour of CARA is as follows. All robots in the fleet start as exploration robots with the role of exploring and mapping the unknown explorable space. They all start exploring from the position of the orchestrtor, expanding further and further away from it with time. Every 500 ms, all robots transmit a heartbeat packet allowing the orchestrator to estimate the Packet Delivery Ratio (PDR). The orchestrator keeps track of the history of timestamps when a heartbeat packet was received for each of the robots. Periodically, for each robot, the timestamps are used to compute the estimated PDR over a pre-configured sliding window period (in seconds), using (6.1). For example, if the sliding window period is 10 seconds, and there are 5 timestamps stored between $t = x$ and $t = x + slidingWindowPeriod$, that means 5 packets were received within the last sliding window period, out of an expected 20.

$$estimatedPDR = \frac{numberOfPacketsReceived}{slidingWindowPeriod \cdot 2} \tag{6.1}$$

The resulting estimated PDR for each robot is stored in the orchestrator's communications history map along with the position of the robot at the time of estimation. For example: [Robot_id : 5, position : (10.5, 2), estimated_PDR: 0.7]. The orchestrator checks periodically for the estimated PDR values of all the robots. Once the PDR goes below the lower limit threshold for any robot, that robot is set to become a relay robot rather than an exploration robot. Next, the orchestrator chooses a position to send that relay to. It searches the

communications history map for the PDR history of the robot and looks for the closest position where it had a PDR above the upper limit threshold.

Algorithm 6: CARA algorithm

```
1  for i ← 0 to numberOfRobots do
2  |  if currentEstimatedPDR[robot[i]] < lowerThreshold then
3  |  |  set robot[i] as relay;
4  |  |  break;
5  |  end
6  end
7  if relay then
8  |  for i ← 0 to length of reversedPdrHistory do
9  |  |  if reversedPdrHistory[i][pdr] >= upperThreshold then
10 |  |  |  relayPosition ← reversedPdrHistory[i][position];
11 |  |  end
12 |  end
13 end
14 handover to navigation algorithm to move relay to it's assigned position;
```

As shown in algorithm 6, the CARA algorithm is broken down into four main steps:

1. Is a relay check due? If the amount of time that has passed since the last check is equal to the sliding window period for checking the PDR, then yes. Otherwise, do not check if relays are needed yet.

2. Is a new relay needed? If the estimated PDR is below the lower threshold then the answer is yes, otherwise no.

3. Which robot should stop exploring to become a relay? The robot with the PDR below the threshold is the one that should become a relay.

4. Where should the new relay be sent to be stationed at? This would be the closest position to where that robot currently is, that had an estimated PDR above or equal to the upper threshold when it last traversed there.

Based on this, if the PDR never goes below the upper threshold no relays will be placed. If it happens once, one relay will be placed, and so on. We do not specify the number or relays prior to the mission nor do we set which robots are to become relays. Hence, the number of relays placed depends entirely on the quality of communications in the environment, the dynamic context aware reactivity to the environment.

6.4 Simulating CARA

We evaluate the CARA relay placement algorithm and compare it to the DBRA algorithm through simulation. The simulation of both algorithms requires: a centralised exploration and mapping algorithm that requires communication between a central orchestrator and all the robots in a fleet, and a communication model to dictate how the packets are transmitted and received and how we model the quality of communications in an environment.

6.4.1 Setup

We use the Atlas 2.0 exploration and mapping algorithm from chapter 5 to test how the CARA algorithm affected the performance of the mission verses the DBRA algorithm. Atlas 2.0 is a centralized algorithm, hence, it relies on robust communication between each robot and the "orchestrator". Each robot in the fleet receives a new command with movement instructions every time a robot stops upon fulfilling its last given task or bumping into an obstacle. The orchestrator maintains an artificial overlay grid that it builds on the go during the exploration which can be expanded infinitely. Each grid cell belongs to one of the following categories at every point in time:

- *Explored Open Cells*: containing no obstacles
- *Explored Obstacle Cells*: containing obstacles
- *Frontier Cells*: the cells surrounding the explored cells that should be explored next to expand the map.

The algorithm is frontier based, meaning that it moves each robot to a random frontier cell out of the ones that are closest to it. The A* algorithm [89] is used to find the shortest path to that frontier cell. Vectoring is used to set the new movement instruction for the robot. Vectoring is a navigation service provided to aircraft by air traffic control: the controller decides on a particular airfield traffic pattern for the aircraft to fly, the aircraft follows this pattern when the controller instructs the pilot to fly specific headings at appropriate times. In Atlas 2.0, the orchestrator replaces the controller and the robot replaces the plane. The movement pattern is the path generated by A*. A robot moves at the speed and in the heading instructed by the orchestrator until its allocated movement duration runs out, unless it bumps into an obstacle. The overall behavior is that the frontier expands "away" from the starting position: the robots are controlled to "push" the frontier further from the starting point.

In the relay algorithms we compare, the chosen relay robots will be instructed to move to a "relay position", assigned by the algorithms, rather than to a frontier

cell. The same navigation method is followed to send the relay from it's current position to it's allocated relay position. The robots navigate and explore in the same manner with both CARA and DBRA relay placement algorithms. The only differences are: (i) when robots are assigned to switch from exploring to becoming relays. (ii) which robots are assigned as relays. (iii) where the relays are positioned.

To trigger diverse behaviour in the algorithms, we test them with different floorplans. The "Empty" floorplan is the simplest one: an empty room. We use it as a reference, to evaluate the impact of obstacles on the overall performance. The "Office" floorplan represents a more complete end-to-end use case, in which a fleet of robots is tasked to map out a floor of an office building. This could be for a search and rescue mission, to search for victims after a fire or natural disaster, for example. Finally, the "Factory" floorplan, which is based on a chemical process plant blueprint from [90]. This would be for use-cases such as hazard detection in a chemical plant, for example.

We also run simulations with various numbers of robots (15, 25 and 50 robots) to evaluate the impact of the relay to exploration robot ratio on the time to completion.

6.4.2 Communication Model

The most crucial element that the relay algorithms depend on is how the different devices in the system communicate with one another. In the previous chapter we used a star topology for the communication protocol. However, this limits the range of the exploration, as neither the exploration and mapping algorithm, nor the communications protocol had a method to handle robots being completely out of range. In this chapter we solve this problem by considering a mesh topology to extend the range. In this section we explain the communication model we used for the simulations in more detail.

As explained in chapter 3, to compute the link stability directly between any two devices, we use the Pister-Hack (experimental randomness) model.

Concurrent Transmissions (CT) refers to tightly synchronized simultaneous transmissions. Multiple nodes in a network transmit the data they want to share with one another simultaneously (within 500 ns). Any nodes that overhear the concurrent/synchronous transmissions receive one of them with a high probability due to the following effects: (i) Capture effect: A receiving radio can capture one of the many colliding packets under specific conditions related to the technology used. (ii) Non-destructive interference: If the colliding packets are tightly synchronized and have the same contents, the resulting signal may be distorted, yet it is highly probable that they will not be destructive to each other. Hence, the receiver can recover the contents with a high probability.

CT embraces the broadcast nature of the wireless medium and synchronizes

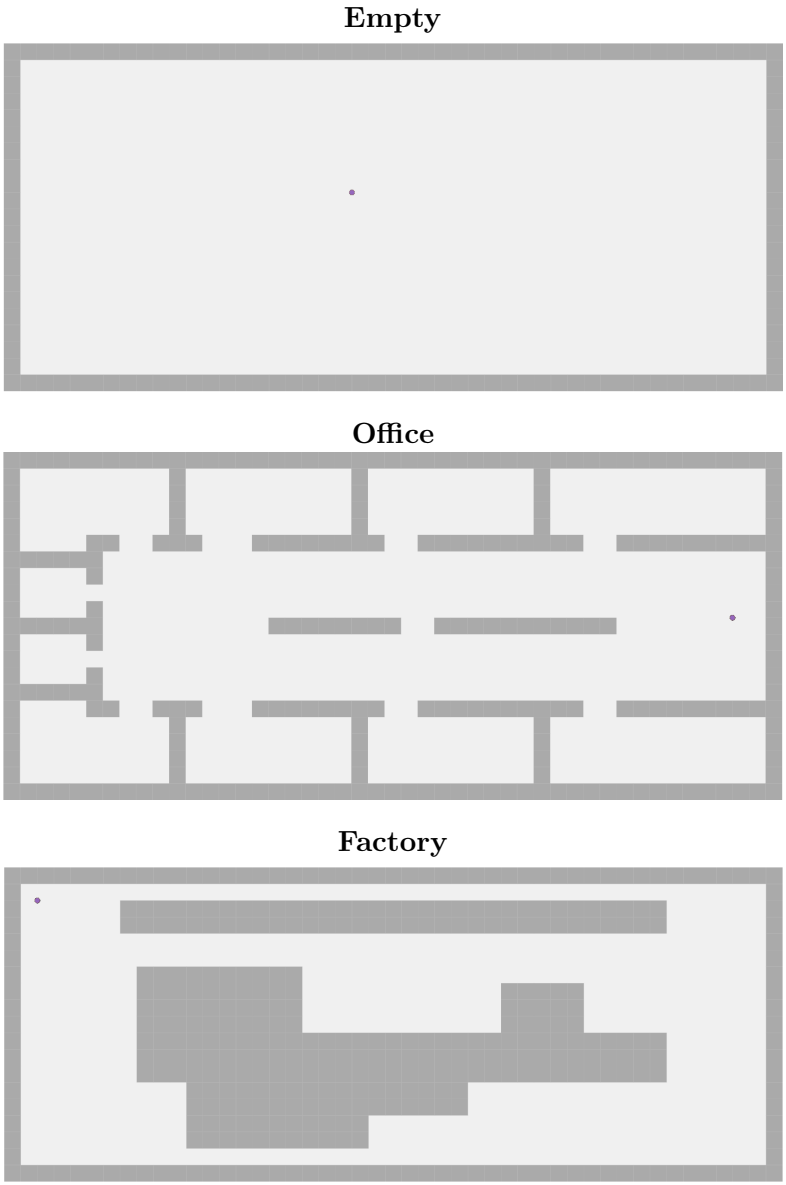


Figure 6.2: The three simulated floorplans. All floorplan areas are the same size (47×21 $m2$)

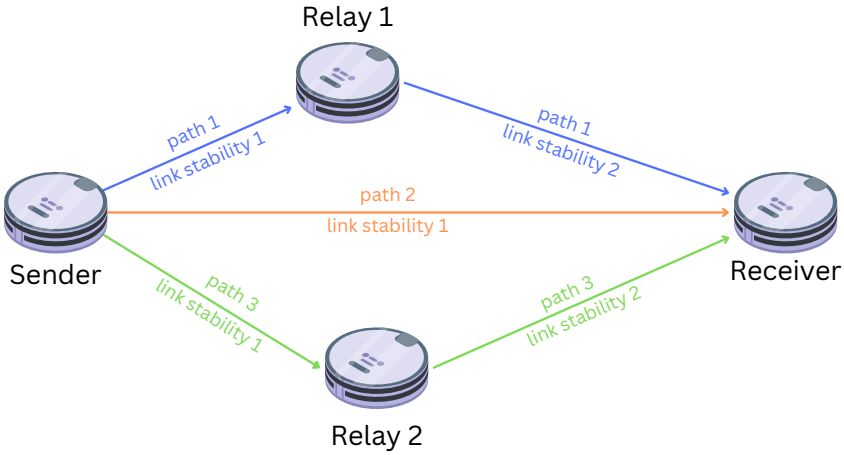


Figure 6.3: Illistrution of Concurrent Transmissions

transmissions to enhance the probability of packet reception. CT also benefits from sender diversity where the concurrent senders have independent links to the receiver. More importantly, this is a simple yet efficient flooding primitive that avoids the implementation and operation overhead of routing and link-based communications. It also achieves enormous performance gains in terms of the end-to-end reliability, latency and energy consumption even under harsh interference conditions [91].

Another advantage of CT that makes it suitable for our application is that it uses stateless relaying; a node can broadcast the packet to all nodes. This eliminates the need for complex routing tables that increase processing and memory requirements, communication overhead and complexity.[92]

In our model, only the relays re-transmit the packet they receive simultaneously. If we have no relays the PDR is equal to the link stability computed based on the RSSI computed using equation 3.1. If we have, say, 2 relays, and one of the exploration robots sends a packet to the orchestrator. As soon as the relays receive that packet they re-transmit it once. Relay 1 receives packet from Robot x and transmits that exact same packet and Relay 2 receives the packet from Robot x and transmits the exact same packet. Assuming the packet reached both relays and they both transmitted the packet simultaneously within 500 ns of one another, there are now three independent paths the orchestrator can recieve that packet from Robot x through. As shown in Fig. 6.3These paths are:

1. Robot x → Relay 1 → Orchestrator
2. Robot x → Relay 2 → Orchestrator
3. Robot x → Orchestrator

To compute the total PDR from Robot x to the Orchestrator, first we need to compute the probability of failure for each path, aka, the probability of the packet not being received at all through that particular path. We apply equation 6.2 to all three paths. Then we compute the total probability of success using equation 6.3. The probability of success is the probability of the packet reaching from the sender to the receiver.

$$pathProbabilityOfFailure = 1 - \prod_{1}^{Nlinks} (linkStability) \tag{6.2}$$

$$probabilityOfSuccess = 1 - \prod_{1}^{Npaths} (pathProbabilityOfFailure) \tag{6.3}$$

6.5 Simulation Results

We used vesrion 2 of the Atlas open source simulator to simulate 1800 exploration and mapping runs in total, including the three floorplans as well as three different fleet sizes. We run the CARA simulations with an upper threshold of 0.9 and a lower threshold of 0.8. The distance specified for the DBRA disk range was 7 m. The work of Brun et al. [93] demonstrates how, due to multi-path fading, PDR can not be directly mapped to distance at all times. Therefore, the distance of 7 m was chosen based on trial and error to determine the optimal distance where robots did not lose connectivity completely while minimising the number of relays placed. In this section we evaluate and compare CARA against the DBRA algorithm in terms of time to completion, number of relays placed and PDR evolution as relays are placed. All results are presented with a 95% confidence interval.

6.5.1 Impact of Relays on Time to Completion

Fig. 6.4 shows that DBRA takes almost 10 times longer to complete the exploration with 15 robots. The higher the number of robots in the fleet, the smaller the difference in time to completion between both algorithms. This is due to the ratio between the relay robots and the exploration robots. With 15 robots, CARA places, on average, eight relays by the end of the mission, whereas DBRA places

14 consistently in every case (as there is no variance, the "box" appears as a line in the plot). CARA places approximately 50% of the fleet as relays while DBRA place 90%, which is a drastic difference, leading to a significant difference in the time to completion. Consequently, DRBA will only have 10% of the fleet exploring towards the end of the exploration compared to CARA with 50%. With 50 robots, CARA places 16% of the fleet as relays, and DRBA places 40%, leaving DRBA with 60% of the fleet to explore even after all relays are placed. Similar behaviours can be observed for all three floorplans.

Fig. 6.4 also demonstrates how CARA is more reactive in terms of the number of relays placed. With 15 robots, DBRA almost always ends up placing 14 relays by the end of the mission, whereas, with CARA, it is somewhere between 5 and 12. This shows that CARA does not always place the same number of relays. The number of relays placed varies based on the PDR, which is the only difference between all the simulation runs with the same floorplan and number of robots. Similar behaviour is observed with different fleet sizes and floorplans.

6.5.2 Evolution of PDR as Relays are Placed

Fig. 6.5 shows the evolution of the average PDR of all the robots with time. We see that with CARA the PDR does indeed remain above the upper threshold of PDR=0.8 at all times and for all configurations. However, DBRA maintains a PDR of almost 1 at all times. This means that DBRA will loose less packets compared to CARA. Thus, with CARA, a packet may take longer to reach the receiver, which, in turn, adds delay to the overall expedition. However, given that CARA completes the exploration 10 times faster than DBRA, we conclude that the impact of the number of relays on the time to completion is far more significant than the 10% difference in the average PDR as long as communications is assured all throughout the mission assuring no loss of data.

Fig. 6.6 show the evolution of the minimum PDR with time. We see how the PDR keeps deteriorating until the first relay is placed for both algorithms. From there we see how the minimum relay spikes back up. Here lies the difference in both algorithms. With CARA we can map the change in PDR to the relay placement. We see as the minimum PDR dips below the lower threshold, a new relay is placed, then the PDR starts to go back up. On the other hand, with DBRA, after around 100 seconds, the PDR seems to be stable at 1 with 8 relays in place. Yet more relays keep getting placed, despite not being needed, unnecessarily adding to the overall time to completion.

In general we see how adding communication awareness optimises the number of relays placed. The challenge mostly lies in balancing between reliable communications, represented in our work in terms of PDR, and between the time to completion. CARA is a step towards this balance, as it maintains a PDR above

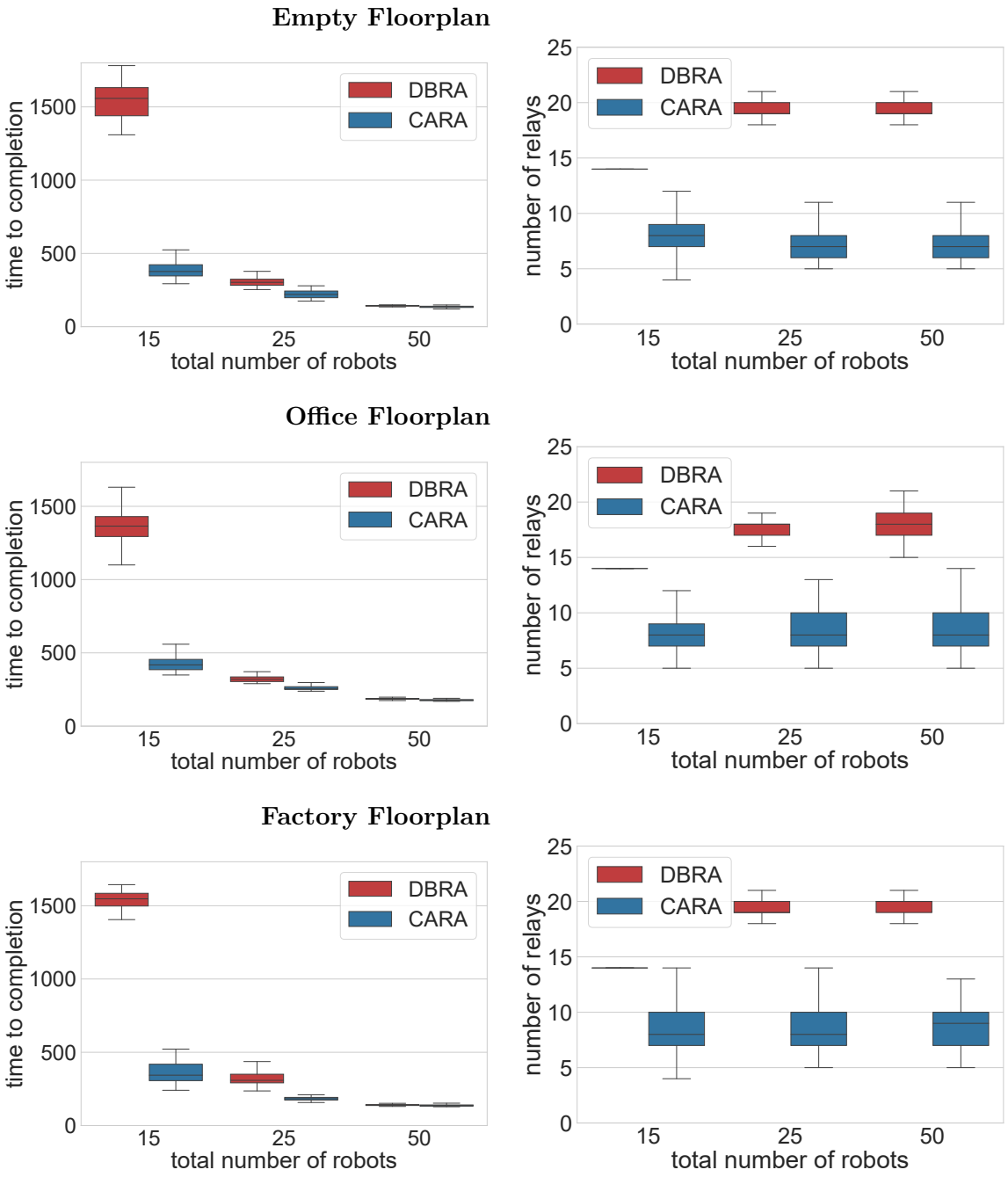


Figure 6.4: The time to completion and the number of relays placed by the end of the mission for all floorplans with three different fleet sizes.

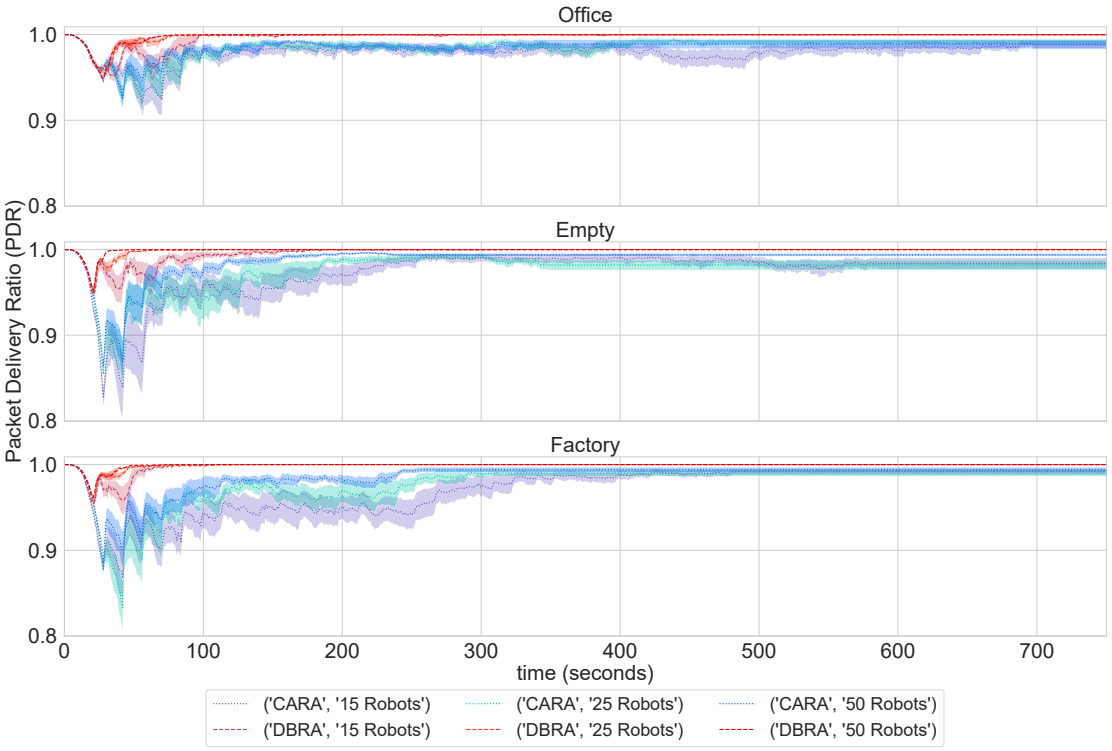


Figure 6.5: Evolution of average PDR with time for all floorplans. After 750 seconds the data remains stable for the rest of the experiment.

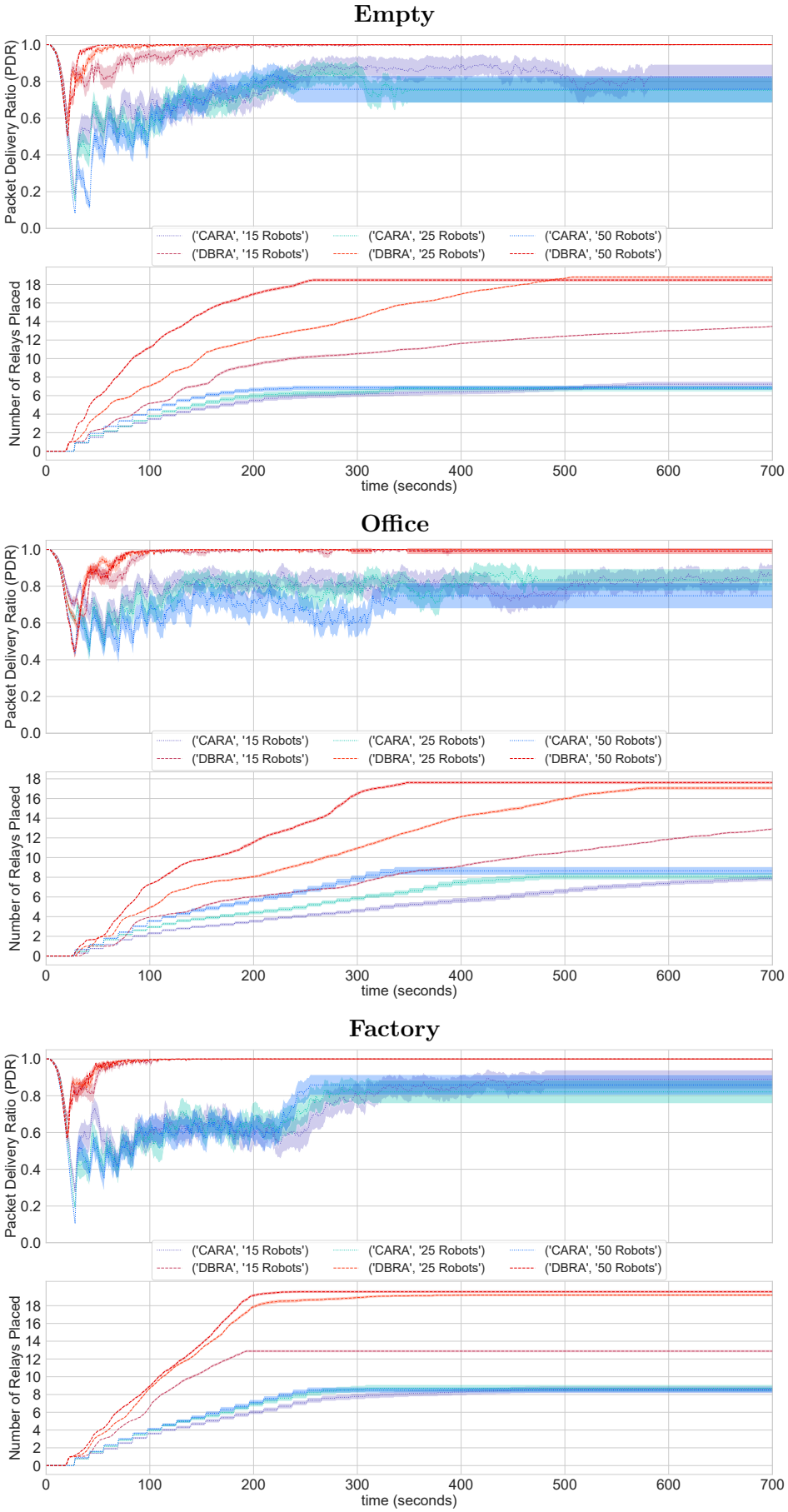


Figure 6.6: Evolution of minimum PDR as relays are placed for all floorplans. After 700 seconds the data remains stable for the rest of the experiment.

0.8 on average at all times with different floorplans and fleet sizes. It also completes the exploration 10 times faster than DBRA despite the fact that DBRA maintains a 10% higher average PDR throughout the mission.

6.6 Conclusions

Exploration of unknown environments is an essential application of multi-robot systems, especially in critical missions such as hazard detection and search and rescue. These missions share the need to reach full coverage of the explorable space in the shortest time possible. To minimize completion time, robots in the fleet must be able to exchange information about the environment reliably with each other. One of the main ways to expand coverage is placing relays. However, existing relay placement algorithms tend to either require prior knowledge of the environment, or rely on maintaining specific distances between the relays and the rest of the robots. This which lacks flexibility and adaptability to the environment. The research question we focus on is how can we place relays *(i)* to maintain communications as reliable as possible and *(ii)* dynamically throughout the exploration mission without prior knowledge of the environment, in a way that minimises delay to the exploration and mapping time to completion.

This chapter introduces CARA (Connectivity Aware Relay Algorithm), a dynamic context-aware relay placement algorithm that does not require any prior knowledge of the environment. We evaluate CARA against a state-of-the-art distance based algorithm, to compare connectivity performance and time to completion. Results show that CARA outperforms DBRA in terms of time to completion by a factor of 10 as it places, on average, half the number of relays that DBRA does by the end of a mission.

Chapter 7

Conclusions and Future Work

This chapter concludes this manuscript by summarizing the work and listing its main contributions (Section 7.1), and discussing the avenues for future research that this work opens (Section 7.2).

7.1 Conclusions

This thesis contributes to the growing field of research on multi-robot coordination in unknown environments. Specifically, it focuses on algorithms that reduce the time to completion and maintain high performance while taking into consideration communication inaccuracies and losses. This is to improve the quality of automation in Industry 4.0 scenarios. We focus on exploration and mapping use cases for critical missions and use that as a guiding application for our design choices and evaluation.

In Chapter 1, we start by introducing the role of autonomous systems in the context of Industry 4.0 scenarios. We then discuss exploration and mapping as an application of such systems in Industry 4.0 and highlight critical missions as a guiding application. We then highlight the importance of communications in multi robot coordination, specifically for exploration and mapping, as well as the impact and importance of considering lossy communications in such algorithms. Following that, we define context-aware systems, and explain the opportunities that utilising context-aware communications brings to exploration and mapping algorithms, and how that can enhance the performance of the robots in such missions. We conclude this chapter by providing an outline of the organisation of this thesis.

In Chapter 2, we introduce state of the art related to the subject of this thesis. We start by introducing Industry 4.0 and the role of multi-robot systems in it, highlighting the importance of using multiple robots together rather than just one. We then explain the multiple elements, considered in research, for designing

Multi robot systems, and use those elements as a guide to classify related work on multi-robot exploration and mapping algorithms. First,in terms of exploration strategies, then in terms of communication assumptions made when designing exploration and mapping algorithms. We then highlight current trends used for maintaining reliable communications throughout multi-robot expeditions, with a focus on relay placement. Finally, we end the chapter by summarizing the related work, and listing the key contributions of this thesis.

Chapter 3 introduces in detail both versions of the open-source simulation platform we developed to test exploration and mapping algorithms for multi-robot systems. It starts by explaining why we chose simulation as a way to implement and test our algorithms. Then explains the details of the first version of the simulator. Followed by an explanation of the updated version of the simulator that includes networking and communication functionalities.

Chapter 4 Introduces Atlas, an exploration and mapping algorithm we designed for sparse swarms, and shows a comparison between Atlas and state-of-the-art algorithms. We start by explaining the algorithms we compare in this chapter. followed by the set-up we use to compare the various algorithms. We then show by simulation that efficient algorithms only work for dense networks. We then explain how this led us to design Atlas, an algorithm specific to sparse swarms. We describe in detail the simulation results.

Chapter 5 explains in detail the modifications made to Atlas to make it packet loss tolerant while guaranteeing a 100% completion ratio. We present our system model and the challenge we address in this chapter. Then describe the communication model and its implementation in the simulator. We then detail the modifications made to the Atlas algorithm. Finally, we showcase the impact of packet loss on the performance of Atlas and emphasise the importance of considering the impact of lossy communications when designing algorithms for multi-robot systems.

Chapter 6 introduces CARA, a dynamic relay algorithm we designed that places relays during multi-robot expeditions without prior knowledge of the environment. We compare CARA to a state-of-the-art distance-based relay placement algorithm, demonstrating that connectivity-aware relay placement uses less relays, which in return reduces the time to completion of the multi-robot mission. We start by explaining the distance based relay algorithm algorithm we compare against. Followed by the details of CARA, our proposed context-aware relay placement algorithm. We then detail the simulation environment and setup used to evaluate CARA. Finally, we show and explain the simulation results showing that CARA outperformed state of the art by a factor of 10 in terms of time to completion.

In summary, the main contributions of this thesis are as follows:

1. We develop a simulation platform which we use to quantify and compare the

performance of Ramaithitima's [70] algorithm and two variants of random walk to test our intuition on frontier based coordinated algorithms being better suited for time sensitive applications.

2. We demonstrate that, while efficient with a large number of robots, Ramaithitima does not always result in full maps when using a sparse robot swarm. We therefore design Atlas, a centralised exploration and mapping algorithm specifically designed for sparse swarms.

3. We evaluate the effect of packet loss on the performance of exploration and mapping, however we use an RSSI-based propagation model for modelling the packet loss as opposed to a random model.

4. We develop an event-based communication protocol where the robots in a fleet communicate with a central orchestrator once triggered by specific events and that is robust to degrading network conditions with 100% completion ration with PDRs of 0.1 and above.

5. We design CARA, a dynamic relay algorithm that places relays during multi-robot expeditions without prior knowledge of the environment.

6. We compare CARA to a state-of-the-art distance-based relay placement algorithm, demonstrating that connectivity-aware relay placement uses less relays, which in return reduces the time to completion of the multi-robot mission.

7.2 Future Work

This work has opened up several avenues for future work. Section 7.2.1 discusses improvements that could be made to the algorithms, including considering further communication limitations, adding more accurate localisation considerations and error handling, looking into collision avoidance and dynamic obstacle detection and using hybrid architectures and heterogeneous robots in a multi robot system to enhance performance. Section 7.2.2 mentions the features that could be added to our simulator to ease the transition from simulations to real world experiments and talks about how we aim to test all our algorithms on a testbed of physical robots.

7.2.1 Algorithmic Improvements

Various further improvements can be made to the algorithms proposed in this thesis. This section explains the areas of improvement that can be explored from

an algorithmic point of view.

Further Communication Considerations: One of our main focuses in this thesis was to take communication inaccuracies into account when designing exploration and mapping algorithms for multi robot systems. While we looked into maintaining reliable communications through using concurrent transmissions with relays, further inaccuracies can be considered. For example we could look into channel capacity and how to handle situations where collision happens. We could also evaluate the scaliability of our current communications model and evaluate how it works in the real world and not just simulation. For now we do not consider how increasing the number of robots would affect communication conditions.

We could also look into moving redundant relays if they are no longer needed. For example if we place 3 relays at the beginning, 2 of which are in separate rooms, connecting these rooms to the orchestrator. If the robots are at the point of exploration where those two rooms are fully mapped, no robots will need to go back there. Hence, maintaining connectivity between the orchestrator and those rooms is no longer required. In such cases, the relays could go back to being exploration robots, or could be placed elsewhere, where/if needed.

Adding Localisation Inaccuracies: In our work, we have assumed ideal localisation throughout the mission. We use dead reckoning and ignore factors such as sensor drift for example. In future work we would like to consider ways to handle errors and inaccuracies in localising the robots. For example, how can the orchestrator verify that the location it has computed is correct. What does it do if it detects that a robot is not where it thinks it is?

This would require a more advanced method of localising the robots than dead reckoning, and would require algorithmic changes to accommodate possible inaccuracies and make up for them. This could open room for work on context awareness in localisation and using the systems understanding of the environment it is in to correct localisation errors.

Collision Avoidance and Detecting Dynamic Obstacles: We currently do not consider avoiding collisions between robots. In further work we would add further logic to the path planning to avoid robots crossing paths. Or could give the robots the capability to detect if they are going to collide, and to either adjust their movement themselves or stop and wait for the orchestrator to give them an updated instruction.

We consider static obstacles in our current work. However, in real world scenarios, there will likely be various dynamic obstacles, that move around, such as other robots or humans. Other examples are chairs, doors, etc.

The challenge in such cases would be to detect false positives, and false negatives. For example if a robot detects a chair as an obstacle, then someone moves the chair and that space is no longer occupied by an obstacle. The map in

this case could be updated if another robot manages to move through this space without detecting an obstacle.

Hybrid Architectures for MRS Control: Our algorithms are all centralised. While this allows for systematic and efficient exploration, its still leaves the risk of the orchestrator being a single point of failure. Possible solutions to explore in the future are hybrid architectures, where parts of the logic are centralised, such as building the map together, and others are decentralised, such as the exploration, obstacle avoidance or localisation for example. This would make for a more robust solution.

Using Fleets of Heterogeneous Robots: We designed our algorithms assuming all robots in the fleet have the same capabilities. However, having robots with different capabilities allows for designing algorithms that are more efficient and robust. For example using drones along side ground robots could help with target and obstacle detection as it could increase visibility.

Another solution could be merging both ideas together to have fleets of robots with different computing capabilities and levels of control, which could add heterogeneous capabilities to the fleet and various levels of hierarchical control.

7.2.2 Improvements to Simulator and Real World Experimentation

We aim to add new features to our simulator to allow testing all the enhancements mentioned in this section. Such as further communication limitations and inaccuracies, localisation inaccuracies, hybrid architectures, heterogeneous fleets or robots, collision avoidance, etc. These improvements will ease the transition from simulation to real world deployments and experimentation.

In future work we will be testing all the algorithms in this thesis on physical robots. This would allow further evaluation of the quality of our algorithms and would give a more realistic idea of the inaccuracies we need to account for when designing such algorithms.

Chapter 8

Publications Resulting from this Work

Journal Articles

1. *CARA: Connectivity-Aware Relay Algorithm for Multi-Robot Expeditions*, Razanne Abu-Aisheh, Francesco Bronzino, Lou Salaün,Thomas Watteyne, **MPDI Sensors**, Special Issue on Robotic Systems for Remote and Hazardous Environments, **2022**.

Conference Contributions

1. *Coordinating a Swarm of Micro-Robots Under Lossy Communication*, Razanne Abu-Aisheh, Francesco Bronzino, Lou Salaün, Myriana Rifai, Thomas Watteyne, ACM Conference on Embedded Networked Sensor Systems (**SenSys**), Workshop on Nanoscale Computing, Communication, and Applications (**NanoCoCoA**) Coimbra, Portugal, 15-17 November **2021**.

2. *Impact of Connectivity Degradation on Networked Robotic Swarm Cooperation*, Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifa, Thomas Watteyne, IEEE International Conference on Distributed Computing in Sensor Systems (**DCOSS**), 14-16 July **2021**.

3. *Atlas: Exploration and Mapping with a Sparse Swarm of Networked IoT Robots*, Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifa, Brian Kilberg, Kris Pister, Thomas Watteyne, IEEE International Conference on Distributed Computing in Sensor Systems (**DCOSS**), Workshop on Wireless Sensors and Drones in Internet of Things (**Wi-DroIT**), 15-17 **2020**.

Research Report

1. *Exploration and Mapping using a Sparse Robot Swarm: Simulation Results.* Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifa, Brian Kilberg, Kris Pister, Thomas Watteyne, Inria Research Report 9349, June **2020**.

Software Contributions

1. I am the lead developer of the open-source simulation platform "Atlas" for multi-robot exploration and mapping (`https://github.com/openwsn-berkeley/Atlas`)

Bibliography

[1] M. Ghobakhloo, "Industry 4.0, digitization, and opportunities for sustainability," *Journal of cleaner production*, vol. 252, p. 119 869, 2020.

[2] R. Galin and R. Meshcheryakov, "Automation and robotics in the context of industry 4.0: The shift to collaborative robots," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 537, 2019, p. 032 073.

[3] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[4] L. S. Dalenogare, G. B. Benitez, N. F. Ayala, and A. G. Frank, "The expected contribution of industry 4.0 technologies for industrial performance," *International Journal of production economics*, vol. 204, pp. 383–394, 2018.

[5] V. Yogesh and S. Kharad, "An overview on search and rescue robots during earthquake and natural calamities," *IJISET Int. J. Innov. Sci. Eng. Technol*, vol. 2, no. 5, 2015.

[6] C. S. Tang and L. P. Veelenturf, "The strategic role of logistics in the industry 4.0 era," *Transportation Research Part E: Logistics and Transportation Review*, vol. 129, pp. 1–11, 2019.

[7] A. V. Nazarova and M. Zhai, "The application of multi-agent robotic systems for earthquake rescue," in *Robotics: Industry 4.0 Issues & New Intelligent Control Paradigms*, Springer, 2020, pp. 133–146.

[8] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[9] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[10] F. Amigoni, N. Basilico, and A. Quattrini Li, "How much worth is coordination of mobile robots for exploration in search and rescue?" In *Robot Soccer World Cup*, Springer, 2012, pp. 106–117.

[11] O. I. Gladkova, V. V. Veltishev, and S. A. Egorov, "Development of an information control system for a remotely operated vehicle with hybrid propulsion system," in *Robotics: Industry 4.0 Issues & New Intelligent Control Paradigms*, Springer, 2020, pp. 205–217.

[12] A. Quattrini Li, R. Cipolleschi, M. Giusto, and F. Amigoni, "A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings," *Autonomous Robots*, vol. 40, no. 4, pp. 581–597, 2016.

[13] E. A. Jensen and M. Gini, "Distributed Autonomous Robotic Systems," in *Distributed Autonomous Robotic Systems*. Cham: Springer, 2019, ch. Effects of Communication Restriction on Online Multi-robot Exploration in Bounded Environments, pp. 469–483.

[14] Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Building A ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example," *Robotics*, vol. 6, no. 3, p. 21, 2017.

[15] B. Woosley, P. Dasgupta, J. G. Rogers, and J. Twigg, "Multi-Robot Information Driven Path Planning Under Communication Constraints," *Autonomous Robots*, vol. 44, no. 5, pp. 721–737, 2020.

[16] J. De Hoog, A. Jimenez-Gonzalez, S. Cameron, J. R. M. de Dios, and A. Ollero, "Using mobile relays in multi-robot exploration," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA'11), Melbourne, Australia*, 2011, pp. 7–9.

[17] J. Banfi, A. Q. Li, I. Rekleitis, F. Amigoni, and N. Basilico, "Strategies for Coordinated Multirobot Exploration with Recurrent Connectivity Constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 875–894, 2018.

[18] J. Banfi, "Recent Advances in Multirobot Exploration of Communication-Restricted Environments," *Intelligenza Artificiale*, vol. 13, no. 2, pp. 203–230, 2019.

[19] B. Woosley, C. Nieto-Granda, J. G. Rogers, N. Fung, and A. Schang, "Bid prediction for multi-robot exploration with disrupted communications," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, 2021, pp. 210–216.

[20] P. Rosenberger and D. Gerhard, "Context-awareness in industrial applications: Definition, classification and use case," *Procedia CIRP*, vol. 72, pp. 1172–1177, 2018.

[21] A. L. Wood, G. V. Merrett, S. R. Gunn, B. M. Al-Hashimi, N. R. Shadbolt, and W. Hall, "Adaptive sampling in context-aware systems: A machine learning approach," 2012.

[22] A. Salkham, R. Cunningham, A. Senart, and V. Cahill, "A taxonomy of collaborative context-aware systems," in *UMICS'06*, Citeseer, 2006.

[23] F. Benavides, C. Ponzoni Carvalho Chanel, P. Monzón, and E. Grampin, "An Auto-Adaptive Multi-Objective Strategy for Multi-Robot Exploration of Constrained-Communication Environments," *Applied Sciences*, vol. 9, no. 3, p. 573, 2019.

[24] I. Ermolov, "Industrial robotics review," in *Robotics: Industry 4.0 Issues & New Intelligent Control Paradigms*, Springer, 2020, pp. 195–204.

[25] K. Schwab, "The fourth industrial revolution. cologny-geneva: World economic forum," 2016.

[26] M. Rüßmann, M. Lorenz, P. Gerbert, *et al.*, "Industry 4.0: The future of productivity and growth in manufacturing industries," *Boston consulting group*, vol. 9, no. 1, pp. 54–89, 2015.

[27] J. Kim, P. Ladosz, and H. Oh, "Optimal communication relay positioning in mobile multi-node networks," *Robotics and Autonomous Systems*, vol. 129, p. 103 517, 2020.

[28] M. Zhou and X. Zhang, "Fast collision checking for dual-arm collaborative robots working in close proximity," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 243–249.

[29] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[30] N. Haghtalab, S. Mackenzie, A. Procaccia, O. Salzman, and S. Srinivasa, "The provable virtue of laziness in motion planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 106–113.

[31] I. Jawhar, N. Mohamed, J. Wu, and J. Al-Jaroodi, "Networking of multi-robot systems: Architectures and requirements," *Journal of Sensor and Actuator Networks*, vol. 7, no. 4, p. 52, 2018.

[32] M. Pavithra and T. Kavitha, "A review towards research in multi-robot coordination system," in *Computer Networks, Big Data and IoT*, Springer, 2022, pp. 543–549.

[33] E. Mathews, T. Graf, and K. S. Kulathunga, "Biologically inspired swarm robotic network ensuring coverage and connectivity," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2012, pp. 84–90.

[34] E. R. Hunt, G. Jenkinson, M. Wilsher, C. P. Dettmann, and S. Hauert, "Spider: A bioinspired swarm algorithm for adaptive risk-taking," in *ALIFE 2020: The 2020 Conference on Artificial Life*, MIT Press, 2020, pp. 44–51.

[35] F. Matoui, B. Boussaid, B. Metoui, and M. N. Abdelkrim, "Contribution to the path planning of a multi-robot system: Centralized architecture," *Intelligent Service Robotics*, vol. 13, no. 1, pp. 147–158, 2020.

[36] A. S. Sayed, H. H. Ammar, and R. Shalaby, "Centralized multi-agent mobile robots slam and navigation for covid-19 field hospitals," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, IEEE, 2020, pp. 444–449.

[37] M. Dorigo, V. Trianni, E. Şahin, *et al.*, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots*, vol. 17, no. 2, pp. 223–245, 2004.

[38] M. Haire, X. Xu, L. Alboul, J. Penders, and H. Zhang, "Ship hull inspection using a swarm of autonomous underwater robots: A search algorithm," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, 2019, pp. 114–115.

[39] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot Exploration of Communication-Restricted Environments: A Survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017.

[40] D. Vielfaure, S. Arseneault, P.-Y. Lajoie, and G. Beltrame, "Dora: Distributed online risk-aware explorer," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6919–6926.

[41] J. Hvězda, M. Kulich, and L. Přeučil, "Improved discrete rrt for coordinated multi-robot planning," *arXiv preprint arXiv:1901.07363*, 2019.

[42] W. Gosrich, S. Mayya, R. Li, *et al.*, "Coverage control in multi-robot systems via graph neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 8787–8793.

[43] Y. Yue, C. Zhao, Y. Wang, Y. Yang, and D. Wang, "Aerial-ground robots collaborative 3d mapping in gnss-denied environments," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 10 041–10 047.

[44] A. J. Smith and G. A. Hollinger, "Distributed inference-based multi-robot exploration," *Autonomous Robots*, vol. 42, no. 8, pp. 1651–1668, 2018.

[45] J. Liu, K. Chen, R. Liu, Y. Yang, Z. Wang, and J. Zhang, "Robust and accurate multi-agent slam with efficient communication for smart mobiles," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2782–2788.

[46] I. Bisio, C. Garibotto, A. Grattarola, F. Lavagetto, and A. Sciarrone, "Exploiting context-aware capabilities over the internet of things for industry 4.0 applications," *Ieee network*, vol. 32, no. 3, pp. 101–107, 2018.

[47] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," *arXiv preprint arXiv:2003.05055*, 2020.

[48] Y.-S. Chen and Y.-R. Chen, "Context-oriented data acquisition and integration platform for internet of things," in *2012 Conference on Technologies and Applications of Artificial Intelligence*, IEEE, 2012, pp. 103–108.

[49] S. Kang, J. Lee, H. Jang, *et al.*, "Seemon: Scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 267–280.

[50] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[51] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review," in *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088.

[52] B. Yang, Y. Ding, Y. Jin, and K. Hao, "Self-Organized Swarm Robot for Target Search and Trapping Inspired by Bacterial Chemotaxis," *Robotics and Autonomous Systems*, vol. 72, no. C, pp. 83–92, Oct. 2015.

[53] P. Ghassemi and S. Chowdhury, "Decentralized Informative Path Planning with Exploration-Exploitation Balance for Swarm," vol. abs/1905.09988, arXiv, 2019.

[54] G. Li, D. Zhang, and Y. Shi, "An Unknown Environment Exploration Strategy for Swarm Robotics Based on Brain Storm Optimization Algorithm," in *Congress on Evolutionary Computation (CEC)*, IEEE, 2019, pp. 1044–1051.

[55] J. Yang, D. Zhao, X. Xiang, and Y. Shi, "Robotic brain storm optimization: A multi-target collaborative searching paradigm for swarm robotics," in *International Conference on Swarm Intelligence*, Springer, 2021, pp. 155–167.

[56] H. Luo, S. Yang, M. Zhao, and S. Cheng, "Multi-robot indoor environment map building based on multi-stage optimization method," *Complex System Modeling and Simulation*, vol. 1, no. 2, p. 6, 2021.

[57] S. Shukla, N. Shukla, and V. K. Sachan, "Multi robot path planning parameter analysis based on particle swarm optimization (pso) in an intricate unknown environments," in *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, IEEE, vol. 1, 2019, pp. 1–10.

[58] B. Nakisa, M. N. Rastgoo, M. Z. Ahmad Nazri, M. Nordin, *et al.*, "Target searching in unknown environment of multi-robot system using a hybrid particle swarm optimization," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 13, pp. 4055–4065, 2018.

[59] V. Garg, A. Shukla, and R. Tiwari, "Aerpso—an adaptive exploration robotic pso based cooperative algorithm for multiple target searching," *Expert Systems with Applications*, vol. 209, p. 118 245, 2022.

[60] X. Huang, F. Arvin, C. West, S. Watson, and B. Lennox, "Exploration in Extreme Environments with Swarm Robotic System," in *International Conference on Mechatronics (ICM)*, Mar. 2019.

[61] M. Kegeleirs, D. Garzón Ramos, and M. Birattari, "Random Walk Exploration for Swarm Mapping," in *Towards Autonomous Robotic Systems (TAROS)*, vol. 11650, Springer, 2019, pp. 211–222.

[62] G. LU and H. SUN, "Two-dimensional mapping of swarm robot based on random walk," *Journal of Computer Applications*, vol. 41, no. 7, p. 2121, 2021.

[63] B. Pang, Y. Song, C. Zhang, H. Wang, and R. Yang, "A swarm robotic exploration strategy based on an improved random walk method," *Journal of Robotics*, vol. 2019, 2019.

[64] Y. Katada, "Swarm robots using a new lévy walk generator in targets exploration," in *Proceedings of the 4th International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM2021)*, 2021, pp. 114–125.

[65] Y. Khaluf, S. V. Havermaet, and P. Simoens, "Collective lévy walk for efficient exploration in unknown environments," in *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Springer, 2018, pp. 260–264.

[66] R. K. Ramachandran, Z. Kakish, and S. Berman, "Information correlated lévy walk exploration and distributed mapping using a swarm of robots," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1422–1441, 2020.

[67] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang, and Y. W. Yun, "An improved frontier-based approach for autonomous exploration," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2018, pp. 292–297.

[68] D. L. da Silva Lubanco, M. Pichler-Scheder, and T. Schlechter, "A novel frontier-based exploration algorithm for mobile robots," in *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, IEEE, 2020, pp. 1–5.

[69] N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative frontier-based exploration strategy for multi-robot system," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, IEEE, 2018, pp. 203–210.

[70] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar, "Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 746–753, 2016.

[71] S. Manfredi, E. Natalizio, C. Pascariello, and N. R. Zema, "A Packet Loss Tolerant Rendezvous Algorithm for Wireless Networked Robot Systems," *Asian Journal of Control*, vol. 19, no. 4, pp. 1413–1423, 2017.

[72] J. Banfi, A. Q. Li, N. Basilico, I. Rekleitis, and F. Amigoni, "Asynchronous Multirobot Exploration Under Recurrent Connectivity Constraints," in *International Conference on Robotics and Automation (ICRA)*, IEEE, Stockholm, Sweden: IEEE, 2016, pp. 5491–5498.

[73] T. Zhivkov, E. Schneider, and E. I. Sklar, "Measuring the Effects of Communication Quality on Multi-Robot Team Performance," in *Annual Conference Towards Autonomous Robotic Systems (TAROS)*, Guildford, United Kingdom: Springer, 2017, pp. 408–420.

[74] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, no. 2, pp. 189–200, 2021.

[75] A. Nath and R. Niyogi, "Communication qos-aware navigation of autonomous robots for effective coordination," in *International Conference on Advanced Information Networking and Applications*, Springer, 2020, pp. 1045–1056.

[76] M. Saboia, L. Clark, V. Thangavelu, *et al.*, "Achord: Communication-aware multi-robot coordination with intermittent connectivity," *arXiv preprint arXiv:2206.02245*, 2022.

[77] Y. Gao, H. Chen, Y. Li, C. Lyu, and Y. Liu, "Autonomous wi-fi relay placement with mobile robots," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2532–2542, 2017.

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

[79] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Swarm relays: Distributed self-healing ground-and-air connectivity chains," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5347–5354, 2020.

[80] V. S. Varadharajan, B. Adams, and G. Beltrame, "The unbroken telephone game: Keeping swarms connected," *Links*, vol. 1, no. 2, p. 3, 2019.

[81] C. Pinciroli, V. Trianni, R. O'Grady, *et al.*, "ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[82] R. Vaughan, "Massively Multi-robot Simulation in Stage," *Swarm Intelligence*, vol. 2, pp. 189–208, 2008.

[83] M. Selden, J. Zhou, F. Campos, N. Lambert, D. Drew, and K. S. Pister, "Botnet: A simulator for studying the effects of accurate communication models on multi-agent and swarm control," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, 2021, pp. 101–109.

[84] H.-P. Le, M. John, and K. Pister, "Energy-Aware Routing in Wireless Sensor Networks with Adaptive Energy-Slope Control," *EE290Q-2 Spring*, 2009.

[85] E. Municio, G. Daneels, M. Vučinić, *et al.*, "Simulating 6TiSCH Networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 3, e3494, 2019.

[86] Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Metrics for Performance Benchmarking of Multi-robot Exploration," in *International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2015.

[87] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[88] J. Gielis, A. Shankar, and A. Prorok, "A critical review of communications in multi-robot systems," *Current Robotics Reports*, pp. 1–13, 2022.

[89] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[90] K. Han, S. Cho, and E. S. Yoon, "Optimal layout of a chemical process plant to minimize the risk to humans," *Procedia Computer Science*, vol. 22, pp. 1146–1155, 2013.

[91] B. A. Nahas, A. Escobar-Molero, J. Klaue, S. Duquennoy, and O. Landsiedel, "Blueflood: Concurrent transmissions for multi-hop bluetooth 5–modeling and evaluation," *arXiv preprint arXiv:2002.12906*, 2020.

[92] T. Chang, T. Watteyne, X. Vilajosana, and P. H. Gomes, "Constructive interference in 802.15. 4: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 217–237, 2018.

[93] K. Brun-Laguna, P. Minet, T. Watteyne, and P. H. Gomes, "Moving beyond testbeds? lessons (we) learned about connectivity," *IEEE Pervasive Computing*, vol. 17, no. 4, pp. 15–27, 2018.