A multi-robot communication framework for the analysis and mitigation of network perturbations

Zhivkov, Tsvetan

Awarding institution:
King's College London

KING'S COLLEGE LONDON

DOCTORAL THESIS

A Multi-Robot Communication Framework for the Analysis and Mitigation of Network Perturbations

---

Author:

Tsvetan ZHIVKOV

First Supervisor:

Prof. Elizabeth SKLAR

Second Supervisor:

Dr. Simon MILES

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Natural & Mathematical Sciences
Department of Engineering

July 31, 2020

Declaration of Authorship

I, Tsvetan ZHIVKOV, declare that this thesis titled, “A Multi-Robot Communication Framework for the Analysis and Mitigation of Network Perturbations” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 31-07-2020

Abstract

As multi-robot systems (MRS) become more affordable, they transition from being used in controlled research laboratories into practical environments. For example, in a physical environment it is critical to maintain network connectivity in cooperative multi-robot teams (MRT). However, communication quality is not covered in as much detail as other research topics in MRS. In this field of research the majority of works investigate aspects of communication quality and network connectivity, but present exclusive solutions for very particular problems, and very few show physical (real world) results.

The MRComm (Multi-Robot Communication) framework presented here contributes towards a better understanding of the effects of communication quality on MRTs. MRComm is created to be used for both simulated and physical robot experiments whilst allowing different communication parameters (i.e. network types and network perturbations) to be applied to the robot team. The results from said experiments are used to analyse MRT performance.

The network types implemented in MRComm are: Wireless Local Area Network (WLAN) and Ad-Hoc (AH) network. This is the first time that ROS (Robot Operating System) has been used to connect multiple robots in a self-made AH network to analyse the communication in cooperative task execution.

The MRComm framework can subject a MRT to one, two or no network parameters. Four network perturbations are designed and integrated into MRComm. These are: Simulated Packet-Loss (SPL), which drops a static amount of shared mission messages; simulated Signal Loss Threshold (SLT), a threshold limit established using signal strength and distance; Simulated Signal strength Degradation (SSD), modelled using two separate Support Vector Regression (SVR) models, one for direct line-of-sight and the other for obstructed line-of-sight, which are combined together to model the parameter; and Effective Signal strength Degradation (ESD), obtained directly from the robots in the network. Moreover, MRComm is integrated with a unique Leader-Follower (LF) behaviour, which is a novel approach to mitigate most network connection issues and support uninterrupted communication.

Acknowledgements

I would like to thank Elizabeth for all her guidance throughout my long journey in my academic research and beyond, and for all her support, working hard alongside me those long days when paper deadlines approached. She has given me many opportunities beyond this PhD and thanks to this I believe I have gained many valuable skills and achieved many feats I would otherwise never have imagined or attempted. I consider her more than just a supervisor, rather like family. I would like to thank Simon for his guidance, support and his trust in my resolve and abilities. I am greatful to all my close colleagues and friends at King's College London, for their support and making me feel welcome. Especially Eric, my predecessor, colleague and most importantly my close friend. His invaluable advice and patience helped me understand and persevere through many difficult times not only academically, but personally as well. And of course Alexandra, the most understanding and patient person in my life. She continuously helped me during every single paper deadline and throughout my thesis writing, by endlessly proof-reading and telling me where to fix those god forsaken commas. My life (and my writing) have changed for the better around her. Last but not least, I would like to thank my family. My mother, Svetlana, and father, Ivaylo, for being excellent role models, providing me with never-ending opportunities and motivating me to work hard and learn new things. My grandparents, particularly my grandmothers, whom I did not wish to listen to when younger, but whos advise and love I now appreciate even more.

Contents

List of Figures

List of Tables

List of Acronyms

AMD	**A**dvanced **M**icro **D**evices
AH	**A**d **H**oc
AMCL	**A**daptive **M**onte-**C**arlo **L**ocalisation
ANN	**A**rtificial **N**eural **N**etworks
AON	**A**ll-**O**r-**N**othing
AP	**A**ccess **P**oint
BSS	**B**asic **S**ervice **S**et
CI	**C**onfidence **I**nterval
CPU	**C**entral **P**rocessing **U**nit
CT	**C**onstrained **T**asks
DA	**D**ynamic **A**ssignment
DWA	**D**ynamic **W**indow **A**pproach
ESD	**E**ffective **S**ignal **D**egradation
ESS	**E**xtended **S**ervice **S**et
EV	**E**xternal Task Assignment **V**iew
FSM	**F**inite **S**tate **M**achine
FSPL	**F**ree-**S**pace **P**acket **L**oss
GPU	**G**raphics **P**rocessing **U**nit
GUI	**G**raphical **U**ser **I**nterface
HH	**H**uman-**H**uman
HR	**H**uman-**R**obot

HRI	**H**uman-**R**obot **I**nteraction
IA	**I**nstantaneous **A**ssignment
IBSS	**I**ndependent **B**asic **S**ervice **S**et
ICRA	**I**nternational **C**onference on **R**obotics and **A**utomation
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
IT	**I**ndependent **T**asks
IV	**I**nternal Task Assignment **V**iew
LF	**L**eader **F**ollower
MANET	**M**obile **A**d-Hoc **Net**work
MAS	**M**ulti-**A**gent **S**ystem
MDP	**M**arkov **D**ecision **P**rocess
MR	**M**ulti-**R**obot
MRCP	**M**ulti-**R**obot **C**ommunication **P**roblem
MRComm	**M**ulti-**R**obot **Comm**unication
MRS	**M**ulti-**R**obot **S**ystem
MRT	**M**ulti-**R**obot **T**eam
MRTeAm	**M**ulti-**R**obot **T**ask **A**llocation and Tea**m**work
MT	**M**ulti-**T**ask
NA	**N**ot **A**ssigned
NB	**N**o **B**ehaviour
NetEm	**Net**work **Em**ulator
OS	**O**perating **S**ystem
OSI*	**O**pen **S**ystems **I**nterconnection
OSI	**O**rdered **S**ingle-**I**tem
PSI	**P**arallel **S**ingle-**I**tem
P2P	**P**eer-to-**P**eer
Q-Q	**Q**uantile-**Q**uantile
RAM	**R**andom **A**ccess **M**emory

RAmI	**R**obots-Assisted **Am**bient **I**ntelligence
RBF	**R**adial **B**asis **F**unction
RCAMP	**R**esilient **C**ommunication-**A**ware **M**otion **P**lanner
ROS	**R**obot **O**perating **S**ystem
RPS	**R**endezvous **P**oint **S**trategy
RR	**R**ound-**R**obin
RSSI	**R**eceived **S**ignal **S**trength **I**ndicator
SA	**S**tatic **A**ssignment
SEA	**S**weep **E**xploration **A**ssignment
SLT	**S**imulated **L**oss **T**hreshold
SNR	**S**ignal-to-**N**oise **R**atio
SPL	**S**imulated **P**acket **L**oss
SR	**S**ingle-**R**obot
SRS	**S**erried **R**anks **S**trategy
SRTP	**S**ingularity-**R**obust **T**ask-**P**riority
SSD	**S**imulated **S**ignal **D**egradation
SSI	**S**equential **S**ingle-**I**tem
ST	**S**ingle **T**ask
SUCRE	**S**upporting **U**sers, **C**ontrollers and **R**esponders in **E**mergencies
SVR	**S**upport **V**ector **R**egression
SW	**S**hapiro-**W**ilk
TA	**T**ime-Constrained **A**ssignment
TAROS	**T**owards **A**utonomous **R**obotic **S**ystems
TCP	**T**ransmission **C**ontrol **P**rotocol
TCPROS	**T**ransmission **C**ontrol **P**rotocol **R**obot **O**perating **S**ystem
UDP	**U**ser **D**atagram **P**rotocol
UDPROS	**U**ser **D**atagram **P**rotocol **R**obot **O**perating **S**ystem
USAR	**U**rban **S**earch and **R**escue **O**perations
USARSim	**U**nified **S**ystem for **A**utomation and **R**obot **Sim**ulation

WANET	**W**ireless **A**d-Hoc **Net**work
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WWAN	**W**ireless **W**ide **A**rea **N**etwork

Chapter 1

Introduction

In recent years, there have been substantial advances in robotics leading to their use in practical and industrial settings, such as robots working in warehouses [1] and urban search and rescue operations (USAR) [2, 3]. This is in no small part due to the advancement in technology, which has helped to reduce the size and cost of such technology, and increased interest in the consumer and business markets in using robotics. However, the other integral advancement in technology that drives robotics research, specifically the multi-robot research domain, has been the improvement of communication technology.

Communication technology advances have allowed Cloud Robotics to be used practically for the first time [4]. Moreover, many companies have heavily invested in multi-robot systems, such as Amazon Robotics (i.e. Kiva robot [5]), Starship technologies [6], Small Robot Company and SAGA Thorvald robotics, the latter two of which aim to deliver agricultural services. All these companies have in common the focus of multi-robot research, and use some form of multi-robot system (MRS) architecture. In speed-critical jobs such as warehouse operations, redundancy and safety is required for consistent communication and control of the warehouse robots. There is a higher volume of robots for redundancy in case of failure, which does occur in applications such as warehouse operations, where the robots are in constant use throughout the day. However, this puts a larger focus on the design of the facility (warehouse) as a whole. This implies that the cost of the robots is generally lower in comparison to the communication and control infrastructure and continuous maintenance put in place at the facility. Communication is the key to success for MRS and multi-robot coordination, however this becomes highly complex and extremely expensive when many different robots and tasks are introduced to the system. Even with all the robot and communication redundancy introduced, it remains a difficult and unsolved problem.

MRS communication issues are even more challenging in USAR and agricultural robotics, where the environment is dynamic and there is a certain limit to how much control or regulation is possible compared to warehouse facilities. This makes the problem even more complex and expensive than in the case for warehouse robotics. It may not be immediately apparent that the delivery robot by Starship, compared to Amazon's Kiva warehouse robot, is not only more complex but far more expensive to build and maintain. Moreover, this is further demonstrated by the agricultural robots designed by SAGA Thorvald([7]), in Figure 1.1. Network connectivity continues to be a significant issue in experimental and industrial multi-robot teams, despite the continuous improvement of robotics technology, communication network quality and coverage.

Multi-robot communication and connectivity is a prerequisite for successful multi-robot coordination and operation. It is important for a MRS to have a well designed communication system, where the communication medium and the network type are chosen to suit the environment, the common communication perturbations are known, and fault tolerant communication methods exist to prevent perturbations. While there are several kinds of communication, as outlined in Chapter 2, this thesis is concerned with explicit communication (i.e. Wi-Fi), it's related break-down of communication infrastructure (i.e. ad-hoc networks), and common network perturbations.

FIGURE 1.1: SAGA Thorvald robots, image from [8].

1.1 Motivation

There exist instances in which degrading or complete collapse of communication infrastructure occur. There is a lot of research on improving telecommunication networks for message propagation using mobile systems during disaster situations. Generally, these works focus on re-establishing fragmented or non-existent communication networks using ad-hoc communication. Most research on this topic focuses on human-human communication, but serves to inform work on human-robot and robot-robot (multi-robot) communication as well. Robots will frequently be deployed in task domains where they are not co-located with humans, such as search-and-rescue, humanitarian de-mining or nuclear plant surveillance. It is these types of non-proximal relationships (i.e. multi-robot communication) that are of concern in this work. Even though it is indicated in Chapter 2 that multi-robot communication is a significant issue, most of the research does not focus on linking MRS with communication network quality and ad-hoc networking in a physical environment. Moreover, research focusing on multi-robot communication use a plethora of unique research robot platforms that are designed with specific communication capabilities or sensors, whereas I am interested in what a low-cost research platform with no specialised communication sensors can achieve. Poor communication in MRS is a critical factor to successful operation. An unreliable network connection can mean that messages get dropped and robots lose their ability to receive commands, transmit sensor data and generally interact with other robot team members. In real-world settings, network devices drop connections and bandwidth degrades when signal strength declines, even when using high-speed networks.

This research seeks to find an answer to the following overarching question: *How can a multi-robot team ensure continuous communication throughout the duration of a mission?*

1.2 Research Questions

The research in the multi-robot communication domain is centred around creating communication infrastructure, fault tolerance procedures in the case of communication failure or methodologies directed at improving communication; very little research investigates quantifying and assessing the effects of degrading communication quality in this domain. Furthermore, most research on multi-robot communication, reviewed in Chapter 2, either present experiments that are performed in a simulated environment or simply do not analyse the actual network performance and its impact on the robots. Combining this knowledge with the motivation from Section 1.1, a series of research questions are extrapolated:

1. What are the main network parameters that should be analysed in

a multi-robot team to determine communication quality? Moreover, what multi-robot team experiments can be designed that accurately evaluate how degrading communication (network) quality effects performance? Finally, which performance metrics are important to represent communication success or failure?

2. What is the impact of introducing behaviours (i.e. formations or strategies) that react to the changes of a communications network? Can such a behaviour implemented on a multi-robot team, which is severely affected by network perturbations, allow for the continuous and successful communication of messages throughout the duration of a mission? When communication-aware behaviours are utilised, does this always signify that overall multi-robot performance is improved?

3. In an environment with no network infrastructure, is it possible for a standard low-cost physical multi-robot team to communicate using a self-maintained dynamic ad-hoc network? What is the difference between a network with infrastructure compared to one without (i.e. robot team organisation, messaging design and communication issues)?

4. Are simulated network perturbations accurate enough to represent the effects that degrading communication has on team performance, compared to physical (actual) network perturbations? Moreover, what is the comparative performance between simulated and physical robot teams?

1.3 Contributions

To answer the research questions four main contributions are presented in this thesis:

1. The development of an experimental communication based framework in Chapter 3. I developed the Multi-Robot Communication (*MRComm*) testbed framework to specifically conduct experiments investigating the common communication issues that arise in multi-robot teams. Furthermore, *MRComm* is based on the existing *MRTeAm* framework, developed by Schneider [9], which focuses on market-based task allocation research. In *MRComm* the communication network and other network parameters that affect messaging quality are defined. It is a multi-robot system that employs software agents to undertake task navigation and execute different roles and recovery behaviours, while sharing status information (coordination) between team members. Moreover, it is used for experimentation on both simulated and physical robots. Finally, I integrate *MRComm* with the *ROS* (Robot Operating System [10], further described in Section 3.2) based *FKIE* software package (multimaster-fkie [11], further described in Section 3.2.1), which provides the framework with the ability for multi-node inter-robot communication. *MRTeAm* lacked the ability and robustness of unobstructed and fully distributed message passing between robots. The *FKIE* software package assists *MRComm* toward a MRS which is unobstructed and truly distributed in capability.

2. I present baseline experiment results to assess communication network quality in both simulated and physical experiments and expand upon the standard set of collected performance metrics, which are presented in Chapter 4. Moreover, I validate the hypothesis that poor-quality communication will hinder the proper operation of a multi-robot team by showing that communication quality is a critical factor to mission success. Chapter 4 results are based on benchmark experiments and are split into two parts, preliminary results by *MRTeAm* and *MRComm*. The preliminary results conducted on *MRTeAm* are based

on a partially centralised system and do not yet use some of the network parameters or functionality implemented in *MRComm*. The results assist in the design of *MRComm* by identifying aspects, features, functionalities and parameters of the system that I want to be able to control experimentally, such as establishing the network type, introducing network perturbations, truly distributed agents and the analysis of shared messages, and important performance metrics required to evaluate the effects of degrading communication quality and assist in the final design of the communication network. The preliminary results conducted on *MRComm* use similar experiment configuration to *MRTeAm* experiments, to analyse the difference in performance metrics and more importantly to evaluate the communication specific performance metrics.

3. I present the simulated robot experiments conducted using the *MRComm* framework with the implementation of a unique behaviour and messaging protocol in Chapter 5, which allow the multi-robot team to maintain continuous communication throughout the duration of a mission. The design of the network perturbation introduced in Chapter 5 is inspected along with its affect on the performance metrics.

4. The physical robot experiments displayed in Chapter 6, which use the full suite of network parameters introduced to *MRComm*, are a proof-of-concept and validation that a low-cost physical multi-robot team with no modifications can communicate and self-maintain an ad-hoc network. Furthermore, the experiments allow for the assessment of communication quality in a physical robot team during a mission scenario. The results in Chapter 6 are obtained using a truly distributed MRS. Chapter 6 demonstrates communication-aware multi-robot behaviour that attempts to guarantee or maximise communication in the physical environment using real network perturbations. Finally,

the chapter provides a comparison between simulated signal strength degradation and actual signal strength degradation, which are network signal strength perturbations introduced to *MRComm*. The main contribution here is that once an ad-hoc network is created (deployable in experimental task-based or rescue scenarios) it is self-maintained by a multi-robot team, which enables the robots to continue communicating even in adverse network conditions in the physical environment.

1.4 Thesis Statement

The thesis affirms that, as communication degrades, multi-robot team coordination missions cannot be successfully completed in complex and dynamic environments, such as search-and-rescue, humanitarian de-mining or nuclear plant surveillance etc. Therefore, communication-awareness is vital for multi-robot team performance in such situations. The work presented here is to actualise the use of low-cost physical multi-robot teams in practical environments. I present experiments that are conducted on both simulated and physical robots with no modification (i.e. function "as intended"), and present comparable results.

1.5 Thesis Outline

The thesis starts by outlining the fundamental concepts and preliminaries of networked communication and MRS in Chapter 2. Furthermore, related work in multi-robot communication is reviewed and the broader motivation is briefly detailed. Chapter 3 describes the *MRComm* framework developed for investigating multi-robot communication. It also explains the general experiment design used in later chapters. Chapter 4 presents two sets of preliminary experiment results, which demonstrate MRT performance when subjected to degrading communication quality, conducted initially using *MRTeAm*. The purpose of the initial preliminary results is for assisting in the design of *MRComm*. The second set of preliminary experiments are conducted using *MRComm* and are partially compared to *MRTeAm*. Chapter 5 uses only the *MRComm* framework and many of the new functionalities it brings to present simulated robot experiment results. In Chapter 5 two new network perturbations are introduced along with one new network type. The expanded performance metrics presented in this chapter include specific communication metrics, which analyse the effect that network types, perturbations and robot behaviours have on MRTs. Chapter 6 presents similar experiment results, but conducted on physical robots and using physical communication for the network type. Moreover, in this chapter I introduce an actual network perturbation and compare its results with the corresponding simulated network perturbation. Finally, Chapter 7 presents an overview of all the work in the chapters, I briefly discuss the results of experiments in Chapters 4, 5 and 6, I consider future work and conclude the thesis.

1.5.1 Publications

A portion of the developed framework, results and experiments were published in publications 1, 2, 3 and 4. In Chapter 4, the preliminary results, conducted with the *MRTeAm* framework, were presented in publication 1. In Chapter 5, a portion of the results were presented in publication 2. Publication 3 presented part of the *MRComm* framework from Chapter 3 and part of the results from Chapter 5. In publication 4, a portion of the results from Chapter 6 are presented.

1. Tsvetan Zhivkov, Eric Schneider, Elizabeth I. Sklar, "Measuring the Effects of Communication Quality on Multi-robot Team Performance", *In: Towards Autonomous Robotic Systems (TAROS)*, 2017, pp408-420.

2. Tsvetan Zhivkov, Eric Schneider, Elizabeth I. Sklar, "Establishing Continuous Communication through Dynamic Team Behaviour Switching", *In: Enabling & Supporting RAS Technologies (2$_{nd}$ UK-RAS)*, 2019, pp83-86.

3. Tsvetan Zhivkov, Eric Schneider, Elizabeth I. Sklar, "MRComm: Multi-Robot Communication Testbed", *In: Towards Autonomous Robotic Systems (TAROS)*, 2019, pp346-357.

4. Tsvetan Zhivkov, Elizabeth I. Sklar, "Modelling variable communication signal strength for experiments with multi-robot teams", *In: Robots into the Real World (3$_{rd}$ UK-RAS)*, 2020, pp128-130.

Chapter 2

Background

2.1 Introduction

This chapter describes and reviews related work to the multi-robot communication problem (MRCP). In Section 2.2 the fundamental concepts of communication, which are required for multi-robot communication, are explained. This includes the specifications and communication networks used for experiments in this thesis. Section 2.3 describes multi-robot systems (MRS), and important concepts and definitions are introduced, which are used in further discussion in the thesis. I introduce behaviour-based control, which is an important concept for the MRCP. The MRCP research is expanded upon and related work is reviewed and discussed in Section 2.4. Section 2.5 briefly discusses the broader context of communication research involving human-human and human-robot communication. Furthermore, I review and draw motivation from communication related research. Finally, I conclude with Section 2.6.

2.2 Fundamentals of Networked Communication

There are two main categories of communication in multi-robot systems (MRS), namely *explicit* and *implicit* communication. Explicit communication is intentional and is usually directed toward a specific recipient (agent), whereas implicit communication is focused on the act of performing an action (i.e. gesturing, signalling, etc.) to convey information with no regard (direction) to who or what receives it. Related work is reviewed and discussed in Section 2.4. However, the two categories are very broad and the focus here is on underlying issues and possible improvements for explicit communication. There are multiple media that are used to provide explicit communication in MRS, for example Wi-Fi, Bluetooth, ZigBee, infrared, etc. Although the framework designed in this thesis could theoretically support the use of multiple different communication media, I examine the most common one used in MRS today, which is Wi-Fi. Furthermore, the communication issues investigated in later chapters are general to some of the other communication media as well.

2.2.1 Wi-Fi Communication

I briefly outline the basic principles of Wi-Fi, the transportation of data over the medium and in particular the setup used in the experiments conducted in this work. The key factors considered in this work for the Wi-Fi communication are signal quality, noise strength and channel bandwidth.

Wi-Fi uses radio waves and network devices are configured to work at a range of frequencies (channel bandwidth), further described by Otung [12]. There are two common bands for Wi-Fi using the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standard, which are the *"lower-band"*, denoted as 2.4 GHz, and the *"upper-band"*, denoted as 5 GHz. Moreover, every country has marginally different channel bandwidth specifications and allocations. However, in general the frequency values for the *"lower-band"* are in a range between 2.4 GHz and 2.5 GHz and for the *"upper-band"* these are between 5.1 GHz and 5.8 GHz.

The Wi-Fi signal's theoretical broadcast distance (reach) between a transmitter and receiver is found by analysing a common property of wave propagation, which is the attenuation experienced by the signal while travelling in a medium, such as air. The attenuation is defined as free-space *path loss* (i.e. $FSPL$), which is a function of the operating frequency and distance between the transmitting and receiving antennae in relation to the velocity of the wave propagation in the medium. The *path loss* is derived using the (simplified form) Friis transmission equation [13], shown in equation 2.1. The theoretical *path loss* is plotted for a Wi-Fi signal using the 2.4 GHz band in Figure 2.1 to show the expected *path loss* with distance. The result is conceptually useful to understand how distance impacts Wi-Fi signal. Although the theoretical FSPL in Figure 2.1 suggests that a signal is reasonably good even at 10 meters, this is not the case in real world scenarios where congestion and obstacles, such as walls and doors, impact communication (often) severely, as highlighted by Sendra et al. [14]. Moreover, the simplified Friis transmission equation 2.1 does not take into account the transmitting antenna's power and it is not measured with the same SI unit as the one used by conventional Wi-Fi based network cards.

$$FSPL = 20 \log_{10} \frac{4\pi d f}{c} \tag{2.1}$$

where $FSPL$ is the free-space *path loss* expressed in decibels (dB), f is the frequency of the wave in Gigahertz (GHz), d is the transmitter-receiver distance in meters (m) and c is the speed of light in meters per second (ms^{-1}).

It should be noted that standard network cards in most computer devices use a variant of this, termed Received Signal Strength Indicator (*RSSI*),

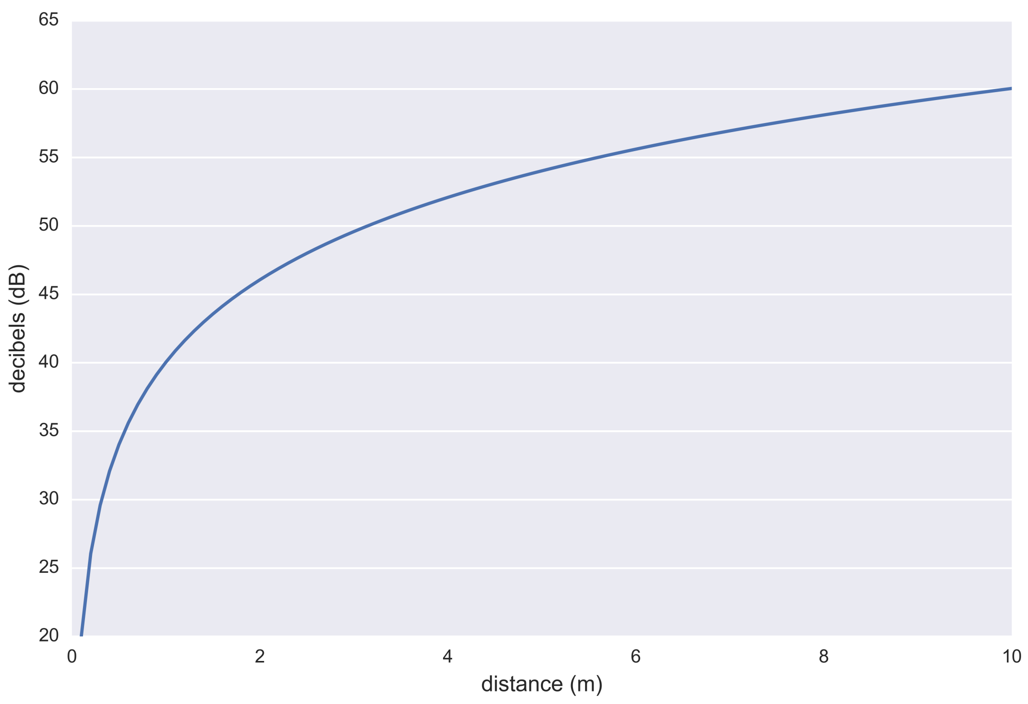


FIGURE 2.1: The theoretical free-space *path loss* experienced by a 2.4 GHz radio signal assuming (ideal) propagation.

which is a subjective measurement, measured in decibel-milliwatts (dBm). This means that the *RSSI* result depends on the (unique) device being used and is subject to change. It is important to distinguish that dB measured in equation 2.1 and dBm represent different but related concepts. Furthermore, a general equation 2.2 can be used to obtain dBm, however a specific equation (2.3) is required to interpret RSSI, as described by Lau and Chung [15]. The raw RSSI results obtained from transmitter and receiver network cards (in the operational environment) are used in the design of some network parameters in Chapter 3 of this thesis and throughout most of Section 5.3 in Chapter 5.

$$P = 10 \log_{10} \frac{P_{abs}}{1\text{mW}} \tag{2.2}$$

where P is the received power expressed in dBm, P_{abs} is the absolute power from the transmitter expressed in Watts (W), measured over 1 milliwatt (mW). It is important to note that this value is still a ratio and a receiver cannot receive a greater power than that of the transmitter, for example 0 dBm indicates a perfect signal with power received of 1 mW and -10 dBm indicates a good signal with power received of 0.1 mW, etc. (unless a transmitter stronger than 1 mW is used and therefore a receiver could receive a signal over 1 mW). Consequently, the RSSI value (dBm) is generally always negative (i.e. equation 2.3) with 0 dBm indicating a perfect signal and any lower negative value indicating a worse signal.

$$RSSI = -(10_n \log_{10} d + A) \tag{2.3}$$

where n is the (unique) signal propagation constant (exponent), d is the distance between the transmitter and receiver and A is the received signal strength at 1.0 meter distance.

The Fresnel zone is another factor to consider for signal quality, which is described as an ellipsoid region in space between and around a transmitter and receiver, described by Hristov [16]. In this region of space obstructions and reflections off the surface of objects may cause minor or major, negative or positive effects to signal transmission that can be predicted. Furthermore, if this region is unperturbed by any obstacle or reflective surface then signal quality *should* be improved. An unobstructed Fresnel zone between transmitter and receiver does not necessarily guarantee good or improved signal transmission, as other factors (i.e. environment and signal power) also impact this. However, this factor is generally considered for the localisation and (permanent) installation of antennae or nodes in a Wireless Wide Area Network (WWAN) or a Wireless Local Area Network (WLAN). An example of the Fresnel zone is illustrated in Figure 2.2. Although I acknowledge the Fresnel zone, there is no practical reason to model the zone in the operational environment as the obstacles in the environment are ever-changing and dynamic. However, the Fresnel zone is useful to describe a design choice made specifically in Section 5.3.

The Signal-to-Noise Ratio (SNR) is the ratio of the signal power to the

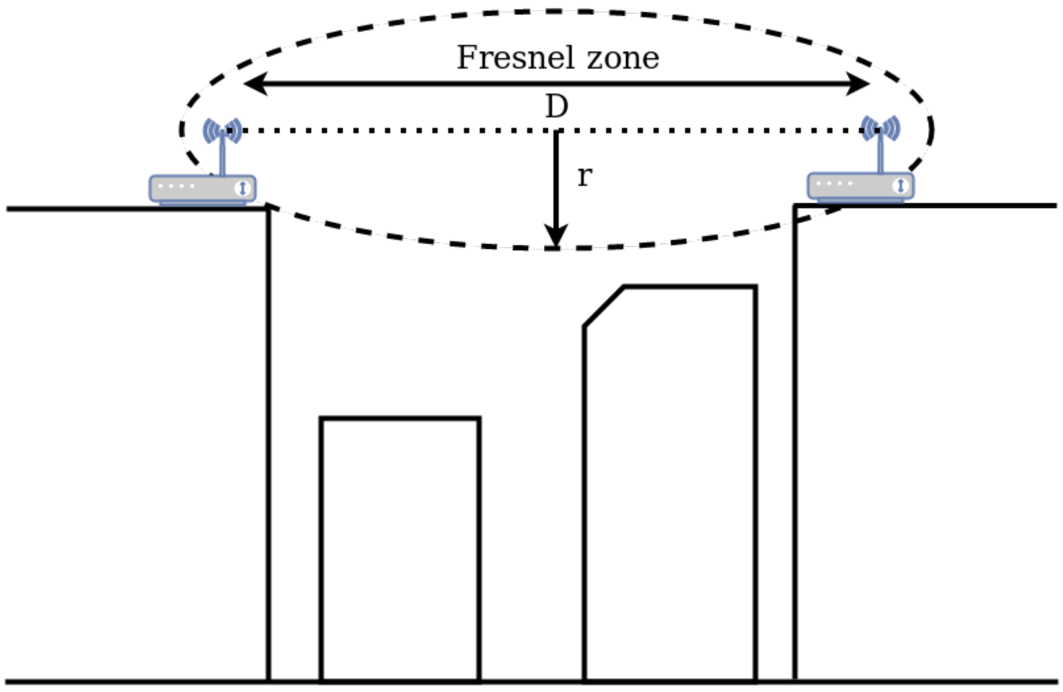


FIGURE 2.2: Fresnel zone: D is the distance between network nodes (transmitter and receiver) and r is the radius of the Fresnel zone (not to scale).

noise power and is commonly expressed in decibels (dB) using equation 2.4, as described by Otung [12]. SNR is a measure of the amount of noise (interference) that can be tolerated before data transmission starts to fail (messages are dropped or corrupted). The SNR is obtained from analogue data and difficult to predict as it is dynamically changing, i.e. network nodes (robots) are constantly moving, environmental changes, etc. Therefore, in this thesis different levels of SNR are assumed and simulated that cause a static number of messages to fail to be delivered. The motivation and reasoning behind the use of simulated message dropping is discussed in Section 3.8.3.

$$SNR(dB) = 10\log_{10}\frac{P_{\text{signal}}}{P_{\text{noise}}} \tag{2.4}$$

The transportation of data is very important for a communication network. It determines the addressing (sending/receiving), congestion control, error control and finally how the connection is managed. There are two commonly used transport protocols in the Internet, such as User Datagram

Protocol (UDP) and Transmission Control Protocol (TCP).

The TCP transport protocol, described in detail by Wu and Irwin [17], is known as a *connection oriented* service, as a connection to the receiver is established before data transmission begins. A connection is established (addressing) using a three-way handshake protocol. This type of transport protocol requires a *server* (sender) that waits for a connection, and a *client* (receiver) that contacts the server. Data traffic is bidirectional, therefore both client and server can send and receive data. A data stream is broken into segments and wrapped in a packet of data along with a TCP header and IP header portion, shown in the top section of Figure 2.3. An IP (Internet Protocol) address has two functions, namely to identify the host network and as a location address in the network. The TCP header portion of the packet is used for acknowledgements of received data, congestion and error control. A data stream arrives in order thanks to TCP's protocol to wait for a receiver to send acknowledgement, which contains the order of arrival of the segments. Furthermore, a connection is managed as it needs to be established before transmission begins and it needs to be terminated once all data has finished transmitting. This makes TCP reliable but complex, which means that transmission throughput can be severely slowed down depending on the reliability of the connection.

The UDP transport protocol, also described by Wu and Irwin [17], is known as a *connectionless* service, as it does not require a connection to be established to a receiver and instead data is transmitted with "*best effort*". Unlike TCP, which requires the establishment of a connection from client to server before beginning to transmit data, UDP "*blasts*" data to a receiver with no requirement to acknowledge receipt. UDP is a very simple protocol and is effectively used for transmitting multimedia applications. The UDP header portion is less than half the size of the TCP header, as illustrated in the bottom section of Figure 2.3, and contains only a port forwarding number, which is used to identify the sending and receiving process. Finally, UDP can be used over a client-server connection but does not require it, e.g. peer-to-peer (P2P) connections. UDP is simple but unreliable, which means that transmission throughput is very fast, but heavily dependant on the reliability of the connection; there is no guarantee that messages will be received, and it is very insecure (i.e. no check is done on the device(s) that receives the data).

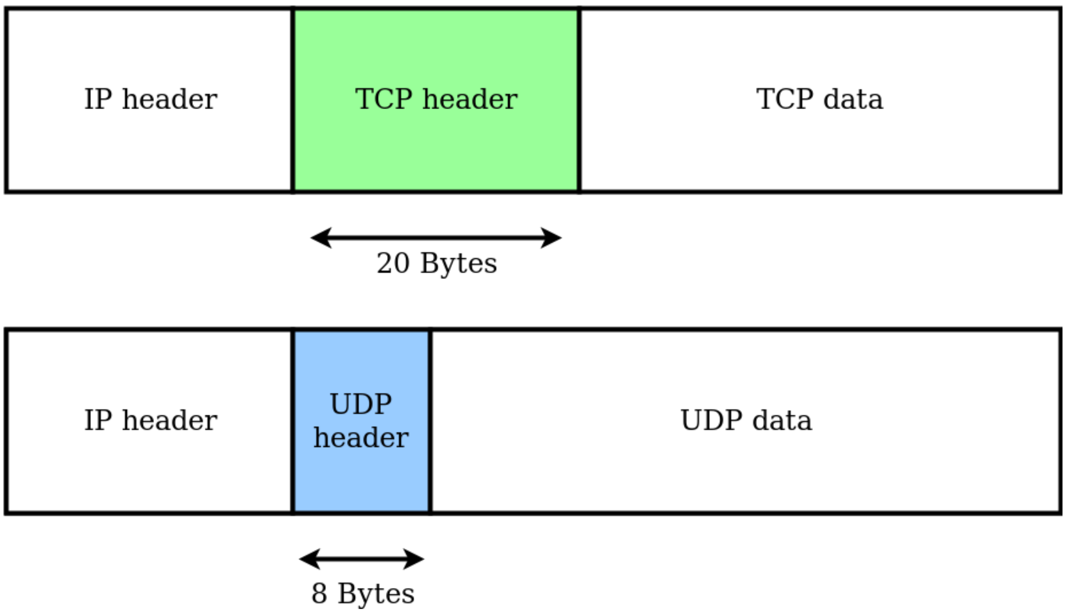
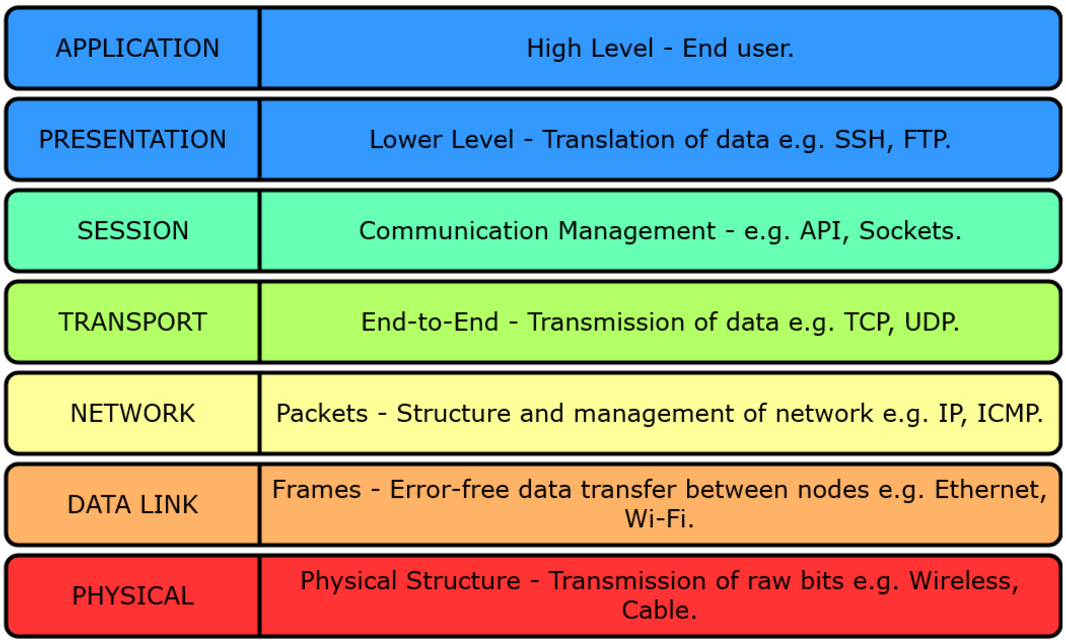


FIGURE 2.3: The top diagram shows a simplified TCP packet in transmission with corresponding byte size. The bottom diagram shows a simplified UDP packet in transmission with corresponding byte size.

The seven-layer Open Systems Interconnection (OSI*) reference model is used to make it easier to discuss certain concepts of the communication network. As can be seen in my interpretation of the OSI reference model in Figure 2.4, based on the book by Wu and Irwin [17], the important layers that will be focused on in this thesis have been defined above. The Physical and Data Link layers in this work are defined as Wireless (radio waves) and Wi-Fi (IEEE 802.11n) respectively. The final two layers that I am interested in are the Network and Transport layers, which are IP (packets) and a variant of TCP called TCPROS, described in Section 3.4. This is all that is required for the design of the communication network that is used in this work.



APPLICATION	High Level - End user.
PRESENTATION	Lower Level - Translation of data e.g. SSH, FTP.
SESSION	Communication Management - e.g. API, Sockets.
TRANSPORT	End-to-End - Transmission of data e.g. TCP, UDP.
NETWORK	Packets - Structure and management of network e.g. IP, ICMP.
DATA LINK	Frames - Error-free data transfer between nodes e.g. Ethernet, Wi-Fi.
PHYSICAL	Physical Structure - Transmission of raw bits e.g. Wireless, Cable.

FIGURE 2.4: An interpretation of the OSI reference model [17].

2.2.2 Wireless Local Area Network (WLAN)

There are many different applications (network types) of Wi-Fi, such as connecting large networks of devices, Wireless Wide Area Networks (WWAN) and also smaller more localised network of devices, Wireless Local Area Networks (*WLAN*). WWAN and *WLAN* are largely very similar in functionality, albeit WWAN is more complex in design and deployment as it is fundamentally made up of multiple *WLAN*s. *WLAN* is the most suitable Wi-Fi network used in office buildings to connect user computer devices and mobile devices. A common *WLAN* setup is shown in Figure 2.5. Moreover, it is the most likely network type to be used to connect MRTs.

WLAN is one of two network types used to investigate communication quality in MRTs, which is subjected to either one or more network perturbations. King's College London is connected using the European institution wide *eduroam* campus Wi-Fi, which is considered both a WWAN and a *WLAN*. The *eduroam* system is deployed and maintained locally in each campus building, which are characterised as *WLAN*. However, a user of *eduroam* can access the network in a different university and/or campus, which is the WWAN aspect of the system. *Eduroam* is sophisticated in its design as it assigns devices automatically to the correct channel bandwidth (i.e. either 2.4 GHz or 5 GHz) to prevent network congestion and improve speed. Moreover, *eduroam* supports both UDP and TCP transport protocols. The campus IT services have strategically positioned three hot-spots (network nodes) in the operational (office) space that is used for conducting experiments. Finally, *WLAN* and WWAN require network infrastructure to continue functioning and serving client devices, meaning that physical network nodes and switches are all connected with physical cabling across different locations. Furthermore, they are complex and extremely expensive to design, deploy and maintain.

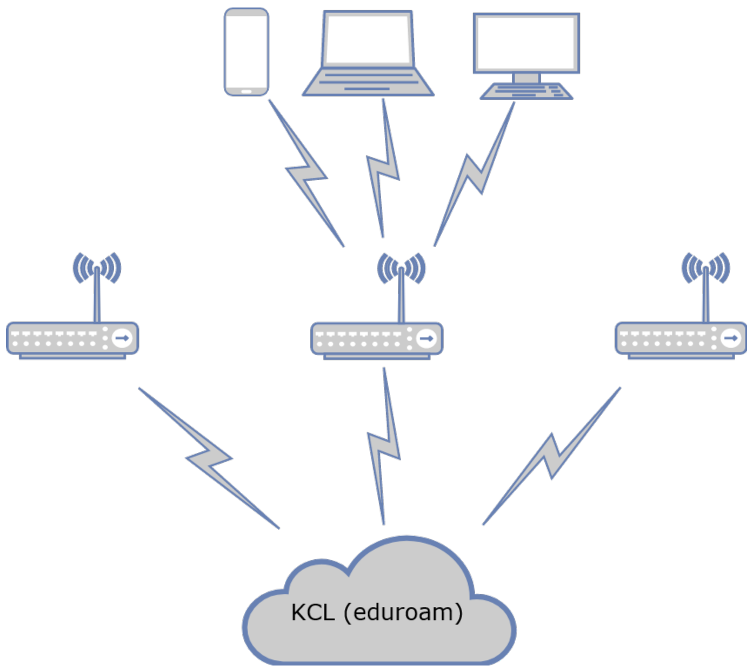


FIGURE 2.5: A diagram showing multiple hubs connecting multiple devices to a WLAN (or WWAN).

2.2.3 Ad-Hoc Network (AH)

An ad-hoc network, also referred to as Wireless Ad-Hoc Network (WANET), is different in some of its functionality to Mobile Ad-hoc Network (MANET) [18]), but the concept is the same. WANET is a more generic term and not as well established as an internet protocol scheme as MANET. According to Prakash et al. [18], MANET could be considered a subset of WANET. An ad-hoc network does not use conventional network architecture (protocols) and is considered either decentralised or distributed. Furthermore, such networks generally do not rely on infrastructure, which means there is no traditional client-server services. Instead, services are provided on a P2P fashion where information is sent/received on a need-to-know basis. A common ad-hoc setup is shown in Figure 2.6.

In this work, the ad-hoc network that comes standard with the Linux Ubuntu Operating System (OS) is the second network type used to investigate communication quality in MRTs. It has additional functionality similar to the well established MANET. The particulars of the OS and ad-hoc network are further described in Section 3.8.1. The ad-hoc network specifically used in this thesis is subsequent denoted as *AH*. *AH* has the functionality to allow for wireless multi-hopping, which is the ability to send information via different nodes (devices) to the intended destination node, as illustrated in Figure 2.7. This functionality needs to be implemented to be used and for the experiments presented in this thesis it is not required. Finally, *AH* has the ability to re-establish a connection with a device that has previously disconnected. Communication between reconnected devices continues as normal once it is re-established.

The *AH* network is a popular method of communication; it is especially useful in multi-robot teams as it can be used to create a simple P2P network solution to re-establish communication in an area where no network infrastructure exists. To create an *AH* network, robots connect directly to one another (P2P) and rely on the close proximity of neighbouring robots to maintain connectivity. As mentioned, robots can leave (e.g. move out of range) and thereafter automatically rejoin (e.g. move back in range) the network freely without issues. However, shared information is only available as long as a connection between robots is maintained.

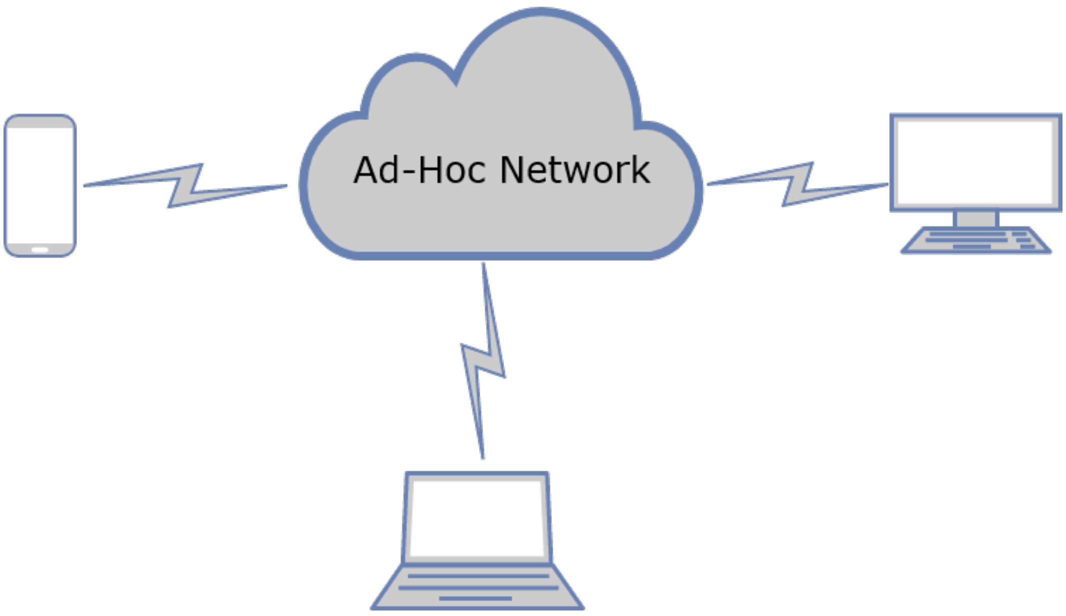


FIGURE 2.6: A diagram showing multiple devices connected to a common ad-hoc network communicating P2P.

2.3 Multi-Robot Systems

MRCP is a problem that falls in a sub-category of a much larger field of study, multi-agent systems (MAS). It is important to define MAS and common terms before introducing the research problem investigated. Two of the field-defining books on artificial intelligence by Russell and Norvig [19] and Weiss [20] similarly define agent. I elicit two fundamental definitions for agent by Weiss [20]: the first "*...a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.*"; the second "*...is an active object with the ability to perceive, reason, and act.*". From the definition it is clear that an agent is an entity that can perform autonomous actions in an environment that it has

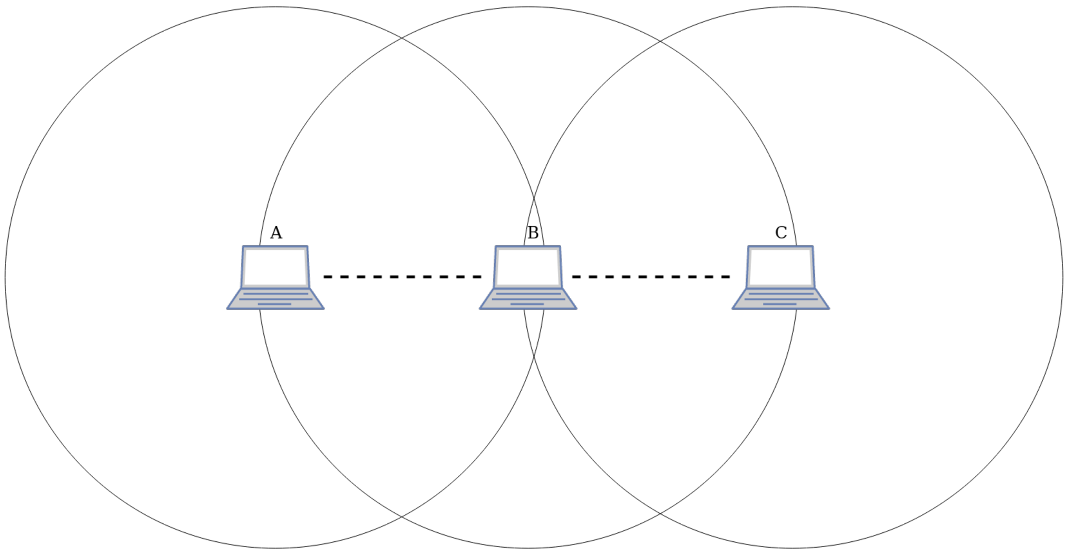


FIGURE 2.7: A diagram showing three interconnected computer devices in Ad-Hoc mode. The dotted line represents direct connection and the circles around the devices represent the connection range. Communication is possible as follows: A → B, B → C, C → B, B → A, A → B → C and finally C → B → A. The latter two cases is where multi-hopping occurs, where point A needs to communicate with point C or vice versa (i.e. B is the hopping point).

a full (or partial) perception of. MAS is multiple agents observing and acting in the same environment to complete a goal(s). Furthermore, MAS have potential advantages over single-agent systems in particular scenarios, e.g. in collaborative task fulfilment, where tasks can be either distributed or decomposed into sub-tasks among agents and simultaneously executed [21, 22]. However, the complexity of the problem increases when using multi-agent solutions compared to single-agent solutions.

A multi-robot system (MRS) is a type of MAS in which the differentiating characteristic is that the agents are embodied and manoeuvre in the environment to perform a common goal. Typically a MRS is a type of system realised in a physical environment [23], however a high fidelity simulator can also be used where the physical environment, robots and sensors are modelled and symbolically represented in the simulation [24]. The main goal of MRS is multi-robot coordination, which means that *"...robots should work together to accomplish a given task by moving around in the environment"* as stated by Yan et al. [23]. However, there are many other uses of MRS, the most common of which are cooperative human-robot interaction [25–27], where a variety of mobile robots, robot manipulators [28] or humanoid robots [29] can be employed to perform unique interaction tasks, and adversarial human-robot/multi-robot interaction [30, 31]. The main topic at hand in this thesis is on mobile robots and cooperative coordination. Therefore, unless directly related to the research problem being investigated, other types of MRS are not reviewed.

Andre et al. [32] evaluate the available software solutions for multi-robot coordination with the focus of exploration and contribute their own communication based software for ad-hoc network communication, which they have integrated into the Robot Operating System (ROS) ecosystem. According to Andre et al. [32], completely decentralised multi-robot coordination requires three main components, which are a global map construct that is shared between robots, a method of communication (not necessarily using common infrastructure, i.e. ad-hoc) and finally a coordinated method of exploration.

2.3.1 Mobile Robot Navigation

To understand mobile robot navigation, the more fundamental problem of localisation needs to be described first. Localisation in mobile robotics asks a simple question, which is *"where am I now?"*. This simple question spans an entire research field and requires quite an arduous and detailed answer. I briefly cover the essential research that explains this concept. The foundational works on mobile robot localisation are by Dieter et al. [33] and Thrun et al. [34], who present early versions of Adaptive Monte-Carlo Localisation (*AMCL*). Most current mobile robot systems have variations of *AMCL* deeply ingrained in their navigation systems, which require the knowledge of transforms (shape, dimensions and position of sensors and robot base), type of sensors and wheel odometry calibration to all work together. The MRS framework designed in this thesis is built on the foundation of the "Multi-Robot Task Allocation" and "Teamwork" (*MRTeAm*) framework by Schneider [9]. The *MRTeAm* framework uses the Dynamic Window Approach (*DWA*) planner [35] for routing paths and obstacle avoidance, which makes use of *AMCL*. The *DWA* navigation planner is continuously updated while a robot is moving, by periodically drawing a predetermined window (perimeter) around the robot. The *DWA* planner uses the outlined window to analyse the next possible navigatory actions it could take, which greatly reduces the impact on hardware resources. The navigatory actions could be, for example acceleration, deceleration, stopping, starting and re-routing (i.e. obstacle avoidance or task re-allocation)

2.3.2 Team Composition

As mentioned previously, a MRS is made up of multiple robots, which make up a multi-robot team (MRT). A MRT consists of a group of two or more robots working together towards a common goal. There are some clear advantages to using a MRT over a single robot, which depend on software, hardware and/or task specific requirements. Some of these advantages are:

- more efficient area coverage, if multiple tasks can be performed in parallel;
- increased robustness;
- decrease the possibility of a single point of failure;
- allow for a greater part of the mission area to be *connected* at any given moment.

The latter two advantages depend on multi-robot organisation, discussed further in Section 2.3.5.

A MRT can be *homogeneous* or *heterogeneous*. A homogeneous MRT is composed of robots that have identical capabilities, whereas heterogeneous MRT may have robots with different physical capabilities, configurations or be entirely different robots altogether [36, 37]. However, heterogeneity can also arise from robots having the capacity to behave differently or implement unique roles under certain constraints, as explored in the early works by Balch [38] and Balch and Arkin [39]. There are also more recent works that further investigate heterogeneity that arises from unique behaviour [40–42]. The robots used in this thesis all have the same capabilities, however employ different roles (behaviours) depending on the experiment. Therefore, it can be argued that the robots used for experiments in this thesis are either homogeneous or heterogeneous.

2.3.3 Task Composition

Task allocation in the MRS field, which is referred to as Multi-Robot Task Allocation (MRTA), is a broad research problem. The literature on MRTA examines different task compositions to investigate a variety of problems in MRS, such as optimisation, planning, cooperation and competition. In this section, task composition is briefly defined, and related work and rationale for the task classification that is chosen for experiments in this thesis is examined.

The works by Gerkey and Mataríc [43], Landén et al. [44] and Schneider et al. [45–47] classify and draw distinctions between the different possible task compositions. The following task classifications are identified from these works, single-task (ST) versus multi-task (MT), single-robot (SR) versus multi-robot (MR), instantaneous assignment (IA) versus time-constrained assignment (TA), independent tasks (IT) versus constrained tasks (CT), external task assignment view (EV) versus internal task assignment view (IV) and finally static assignment (SA) versus dynamic assignment (DA), which are briefly defined in Table 2.1.

Classification	Description
ST	Robots can perform a single task at a time.
MT	Robots can perform multiple tasks at a time.
SR	A task requires a single robot to be complete.
MR	A task requires multiple robots to be complete.
IT	Tasks are independent from each other and have no constraint on ordering etc.
CT	Tasks are constrained and have some dependency, e.g. precedence ordering.
EV	The agent that performs the task assignment is external to the robot team.
IV	The agent that performs task assignment is part of the robot team.
SA	Task assignment is static and usually occurs at the start of a mission.
DA	Task assignment is dynamic and happens over time.

TABLE 2.1: Task composition and descriptions [43, 45].

In this thesis, a specific task composition is not required to solve the particular research problem being investigated. However, with communication being the focus, the most important aspect is investigating the success or failure of communicating task and status messages to robot team members.

The first category of task classification is chosen to be **ST**. The reason being as there is no benefit to the investigation in using MT over ST, at least for the communication analysis experiments conducted in this research. Moreover, the reason ST is chosen over MT is that it is easier to isolate a single task and analyse the particular communication metrics of it. These metrics reveal which task failed, why and when it failed.

The second category of task classification chosen is **SR**. This research does not examine robot task performance in the conventional way, as mentioned previously tasks are hypothetical. A task is simply a set of coordinates on a two dimensional plane, which represents a physical (or simulated) map, that a robot navigates to. Once a robot reaches the task coordinates it sends a success status message or if it does not reach the target coordinates it would send a failed status message. For this particular use case and to make task classification as simple as possible SR is preferred over MR.

The third category of task classification chosen is **IT**. The IT and CT category of task classification makes a big impact particularly on research investigating task allocation optimisation or focusing on particular task performance. However, in this research it does not carry much significance in answering the research problem. It is more beneficial to analyse the communication performance between the robots when they are performing independent tasks (IT). This certifies that if there is no communication backbone (infrastructure), the robot team is not succeeding or failing tasks because of specific CT factors (i.e. precedence ordering), but that task outcome solely depends on successful communication between robots.

The fourth category for task classification is **EV**. This is an important distinction to make for the physical experiments conducted in Chapter 6. In these experiments some configurations distinctly have no network infrastructure. Moreover, the task assigner agent is executed individually in software and technically does not share any knowledge with the task execution agent that performs navigation. In those cases the power of a truly distributed multi-robot system is demonstrated, where the task assigner agent is physically part of the robot team and functioning individually. Apart from the justification given above, there is no particular performance impact or communication specific reason to prefer **EV** over **IV** or vice versa.

However, from the particular requirement of the research and from a software design point of view the task classification here is **EV**.

The fifth category for the task classification chosen is **SA**. To prevent interference with the analysis of communication between the robot team, experiments were designed to minimise the amount of responsibility of the task assigner agent. For that purpose, experiments conducted in Chapters 4, 5 and 6 have the task assigner allocate tasks only once, at the beginning of a mission to prevent further interference.

It has been emphasised that the research focus is not on MRTA and that specific choices have been made that best suit the MRCP being investigated. Therefore, the task composition used in experiments is $\langle ST, SR, IT, EV, SA \rangle$.

2.3.4 Behaviour-Based Control

The earliest related work comes from biologically inspired distributed behavioural modelling in agents, by Reynolds [48]. This work takes inspiration from flocks of birds, herds of land animals and schools of fish, and mimics the observed behavioural patterns. It influenced the work by Balch and Arkin [39], which was the ground work that demonstrated early behaviour-based formation control, on different mobile robot (vehicular) platforms, in both simulation and physical experiments. Balch and Arkin [39] evaluate formation-holding tasks, in which mobile robots maintain certain velocity and distance goals. To accomplish each formation, either certain robots are assigned a "leader" role or a spatial reference point is used to centre the team. The method used (i.e. leader or reference point) would manoeuvre the team in the desired formation. Similarly, the early work of Parker [49] looked into the mitigation (tolerance) of faults in multi-robot cooperation. Parker's [49] framework is designed to be distributed and allows multi-robot teams to explore their surroundings and predict from a selection of possible behaviours, which is the best to use to complete their goal. Unlike

Balch and Arkin, Parker does not employ any formation or control strategy, but base their behaviour selection on sensory feedback.

Behaviour in multi-robot teams has progressed; instead of uniform formation-holding and assigning static roles to robots, recent research allows for the arrangement of formations on the fly to achieve their goal [50] or assign roles dynamically [41, 51]. Moreover, to this day biologically inspired (behaviour-based) modelling is not a solved problem and the research field continues to evolve. Examples of this are in the works by Timmis et al. [52] and Broecker et al. [53]. Timmis et al. [52] examine the use of an algorithm inspired by the immune system's defense mechanism (granuloma) that reacts when a swarm of robots need to recover (self-heal) from faults or failures, whereas Broecker et al. [53] investigate the use of insect-inspired (ants and bees) coordination techniques to improve multi-robot performance. Although biologically (socially) inspired behaviours are the most prominent, they gave rise to artificial behaviours (strategies) such as serried ranks, diamond and phalanx formation, which are mainly used by military and special forces as drill training and movement techniques. Artificial behaviours are inspired by biology, but are usually executed perfectly and created statically, rather than emerging reactively/dynamically. Examples of artificial behaviours in multi-agent systems are seen in the work by Takahashi et al. [54], by Chowdhury and Sklar [55] and inspire the work by Meng et al. [40].

2.3.5 Multi-Robot Organisation

There are two main types of organisation (coordination) that are commonly used in multi-robot systems, which are centralised and distributed. There is an abundance of earlier research on centralised and distributed MRS, for example central control to aid a robot football team to coordinate on the field [56], and a global vision system employing a consensus algorithm for distributed multi-robot formation control [57]. However, over the years the multi-robot research field has advanced considerably and trying to define MRS using two common types of organisation is no longer feasible. Indeed, Das [58] highlights this by noting that an extensive amount of research over the years has been carried out classifying MRS based on many different criteria. Das lists the different types of classifications/components that the literature makes mention of, such as robot capabilities, inter-robot communication, decision making topology (system), organisations and types of co-operation [58]. Figures 2.8 and 2.9 taken from Das [58] illustrate decision making topologies and organisation structure respectively. However, from my extensive research conducted on MRS, it has become clear that communication has an overwhelming impact on many of the aforementioned MRS components. Therefore, I propose an additional layer centred on communication infrastructure in which different MRS components are assigned to a spectrum of organisation types. Three types of organisation are identified, i.e. centralised, decentralised and distributed. The spectrum of organisation types is a high-level amenable abstraction that describes the requirement on each MRS component to perform under different or mixed communication conditions (i.e. infrastructure/infrastructure-less).

In this section, two of the three main components that make up the MRS have their organisation types examined and discussed, namely control (decision system) and communication. The final main component, task composition, is not investigated as it is not beneficial toward the MRCP described in this work. Moreover, I introduce and discuss a fourth component, data recording, that is widely overlooked by the literature, but which the design of MRS is heavily dependent on.

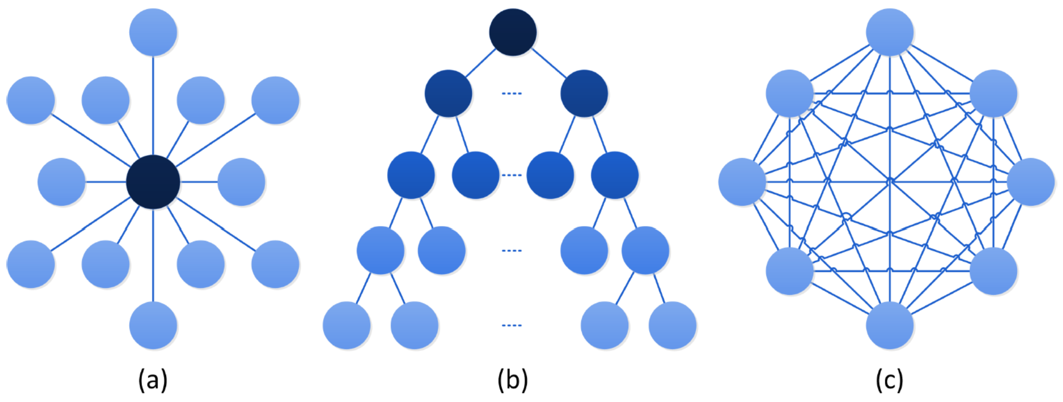


FIGURE 2.8: Decision making topologies in MRS, by Das [58], (a) Centralised (b) Decentralised and (c) Distributed.

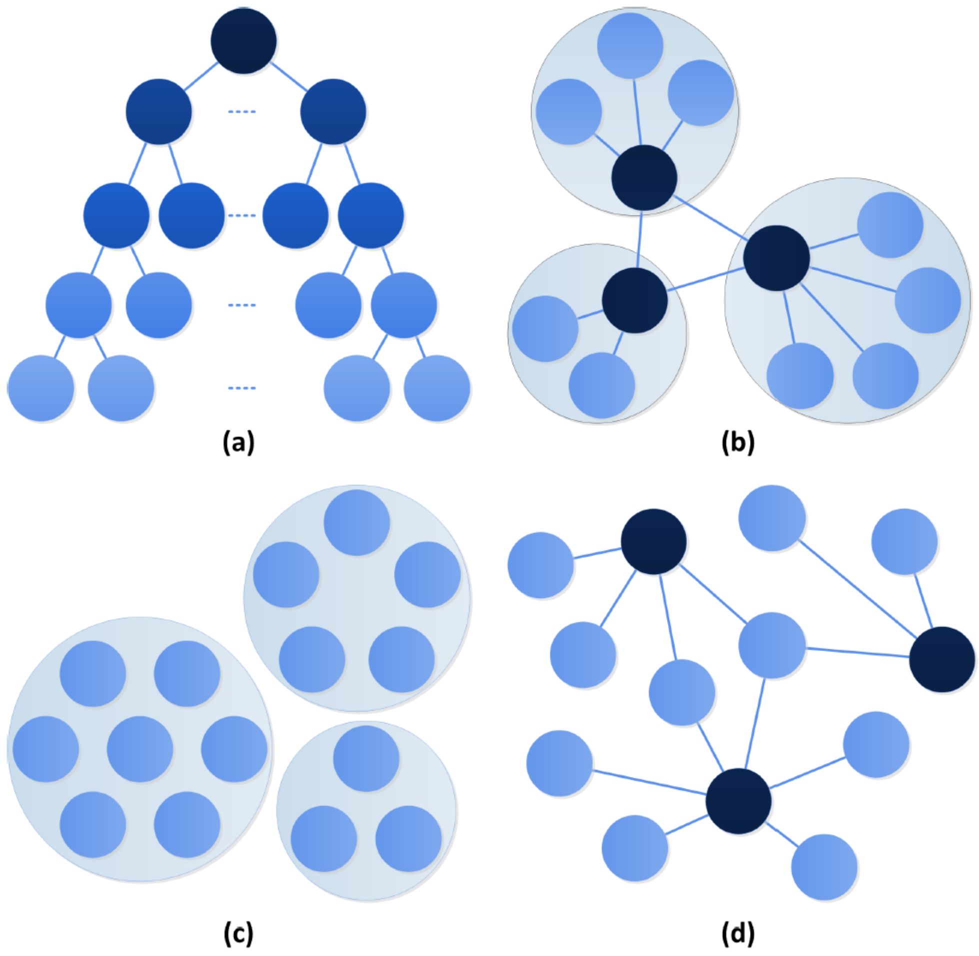


FIGURE 2.9: Organisational structures in MRS, by Das [58], (a) Hierarchy, (b) Federation, (c) Coalition and (d) Market.

Control

Multi-robot control refers to the control and decision making of robots in the environment. There are many methods used to control robots, such as finite state machines (FSM), bio-inspired, reactive or hybrid algorithms. Here I describe how communication infrastructure impacts control for each of the three levels of organisation, i.e. centralised, decentralised and distributed.

Centralised control [56] is the simplest controller used in MRS. The controller enables all robots centrally and can easily detect and react to changes in the environment or goal parameters. However, there is a substantial dependency on a central server or robot to incorporate this type of control. If an environment has no infrastructure or infrastructure were to break down suddenly, this type of control will be unusable. Moreover, the robots using this type of control usually have very little on-board computing power and no individual control.

Decentralised control [9, 47] has become one of the most common methods of control, which was greatly helped by the introduction of middlewares, such as the Player Project [24], RT middleware [59] and the now widespread adoption of ROS [10]. Decentralised control allows multiple control servers or robots to enact control policies and control a subset of robots. The risk of mission failure due to communication breakdown using this controller is reduced, but not eliminated, because if a control server fails and there are no fault prevention techniques, the subset of robots being controlled will become stranded and unrecoverable. Decentralised control methods are generally more complex than centralised control and require more computationally capable robots.

Distributed control [20, 22, 60, 61] has recently gained tremendous traction. Many of the aforementioned robot middlewares have been designed with distributed control in mind. In particular, every new iteration of ROS has expanded its support for distributed robot systems. The risk of mission failure due to communication breakdown using distributed control is reduced greatly. Even if a controller fails, a robot or any subset of robots being controlled by it should be able to continue navigating and carrying out their tasks, albeit hindered (i.e. sub-optimally). Generally, distributed control requires robots with good or even exceptional on-board computing power.

Communication

The most important aspect of designing a MRS is the type of communication. Unfortunately, communication is unpredictable and the best course of action to take is to design an MRS according to certain expectations or assumptions that can be made based on the environment and critical nature of the tasks that the robots will be performing. However, it is not always possible to design an MRS with such specific communication constraints. Therefore, an MRS that is communication-aware is critical in some task environments (i.e. search-and-rescue, agri-robotics and warehouse robotics). Communication networks have many different topologies, but these are extracted and simplified to fit in the three organisational categories outlined in this section.

Centralised communication refers to the *Star* network topology and any other similar variant. This is a very important distinction, especially in MRS design. For example, a hierarchical topology, which may be viewed as decentralised, has a single point of failure (i.e. root node). Although the functionality of the root node is decentralised into child nodes , if the root node fails and there are no preventive measures in place, the entire communication network will break down. In this setup, all communicated messages go to a central node (hub), which are then distributed to the target robot(s).

Decentralised communication refers to generally mesh networks. In these networks, there are usually two or more nodes which provide communication paths to other nodes. Therefore, if one node fails, communication does not break-down for the entire network.

Distributed communication refers P2P and specific variations of mesh networks, which use a P2P style of infrastructure. In these networks, communication is achieved by a source node sending information directly to a target node. Moreover, distributed or P2P networks are commonly infrastructure-less. Therefore, such networks can have individual nodes fail, but overall communication to any other active nodes will be unaffected as communication is self-maintained by each node.

Data Recording

Data recording is a simple component of MRS and many other software based systems, which is the data output that derives common performance metrics. In fact, data recording is immensely important as it enables researchers and data scientists to get new insight, optimise and provide feedback from data. Just like any component in MRS it can be implemented in a centralised, decentralised or distributed organisation pattern. In the research literature, the method of data recording is generally not described, taken for granted and is more akin to a black box. In fact, many works on centralised [62] and distributed [61, 63, 64] MRS design and control have mentioned that experiments are either performed in a hypothetical simulation or physical environment using communication network infrastructure (i.e. not distributed or infrastructure-less). This implies there is still some centralised or decentralised dependency in the system to assist in data recording, which is simpler to implement. Contrarily, distributed data recording is more complex as the robots' on-board clocks have to be synchronised and data has to be merged together to finalise the overall performance metrics.

Centralised data recording is when a single device is used to collect all robot task, status and experiment data. The simplicity of a central data recording device is that it can exist internally or externally of the MRS and does not require synchronisation or merging of data. As with any centralised organisation pattern discussed in this section, there are limitations to using this data recording method. For example, there is single point of failure, and data metrics recorded from different devices are not 100% accurate, as there is a delay and uncertainty in physical wireless communications.

Decentralised data recording is generally better when the architecture of an existing MRS is well designed. This method relies on multiple devices recording specific data for specific performance metrics. For example, robots record their own position data, task status is recorded by an auctioneer/assigner agent and experiment data is recorded by an external recording device. If a single device recording fails, only some of the performance metrics will not be recorded, while others will be unaffected. Depending on the size and complexity of the system, this data recording method could be preferred over using a centralised data recording method.

Distributed data recording is preferable when on-board robot computation is capable enough and when communication is distributed and infrastructure-less. There are instances when communication break-down or robot failure can occur, during which distributed data recording can continue uninterrupted. This is useful for a number of reasons, such as being able to recover individual robot data and any other device's data, and aiding feedback and identifying unforeseen problems in the system through the analysis of the data.

Control, communication and data recording are identified as the three most important components for the MRCP described herein. The additional organisation type associated to the design of each component proposed in this section helps determine if an MRS is capable of functioning in an infrastructure-less communication environment. Moreover, if all components' organisation type is in the extremity, i.e. centralised or distributed, the overall MRS is denoted as truly centralised or truly distributed. The truly distributed denomination suggests that each robot in the MRS is fully capable of communicating, navigating and logging mission data individually in any communication condition, i.e. infrastructure-less environment. The experiments in Chapter 6 are conducted on a truly distributed MRS.

2.3.6 MRTeAm Framework

The *MRTeAm* framework [9] is partly used for experiments in Chapter 4. As there is no standardised method for MRS design or implementation yet, most MRS need to be either recreated from the ground up or need to build upon existing framework (design). The *MRTeAm* framework incorporates many of the systems and design needed by the MRCP research problem; a simple architectural diagram is displayed in Figure 2.10. *MRTeAm* is used as the basis for the research conducted in this thesis and the proposed *MR-Comm* framework is built on top of it. *MRTeAm* implements a "bridge" that deals with communication across multiple robots and connected devices. Figure 2.10 shows with arrows how the ROS Master bridge connects devices and how information flows between the devices.

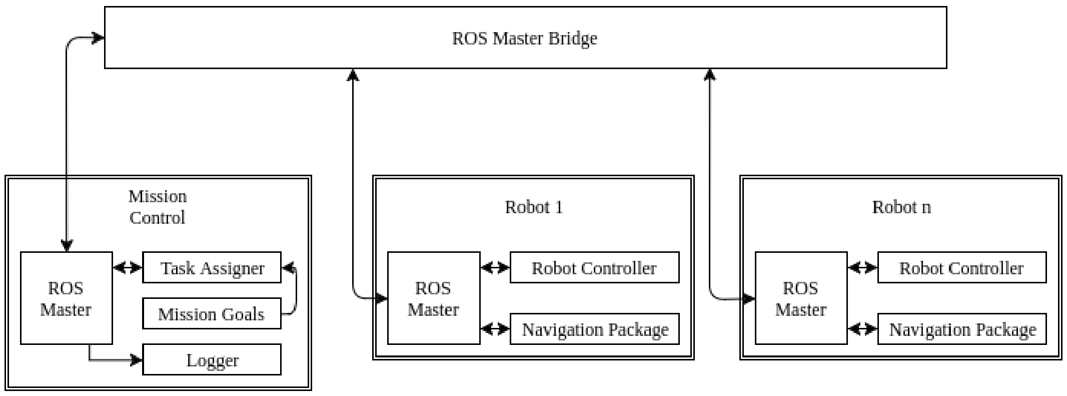


FIGURE 2.10: Simple architectural diagram of MRTeAm [9].

2.4 Multi-Robot Team Communication

There are certain advantages of multi-robot (*MR*) communication, for example the creation of formations, redundancy in task execution and *multi-robot* task execution (i.e. tasks requiring more than a single robot to execute). However, *MR* communication is considerably more complex than single-robot to server (controller) communication. This section reviews some related work to formation strategies and behaviours, restricted communication and exploration, and finally unique communication strategies.

Takahashi et al. [54] use an agent-based software simulation to analyse the deployment of rescue agents using a hypothetical ad-hoc network in a disaster situation where conventional communications networks are unusable. They analyse two strategies to maximise rescue agents' communications, such as *"Rendezvous Point Strategy"* (*RPS*) and *"Serried Ranks Strategy"* (*SRS*). *RPS* disperses agents widely to search for victims and contains a point of contact where the agents return periodically to communicate information. The *SRS* strategy makes rescue agents form up in close ranks and enforces clustered movement. This does not allow for a wide search of the mission area, but it does allow for information to be continuously shared. The strategies reveal that a trade-off exists, in that *RPS* is generally faster at search and rescue, whereas *SRS* is much better at maintaining a consistent and continuous ad-hoc network connection between all agents. It was found that on occasion *RPS*, because it required agents to wait at the rendezvous point to share knowledge with the team, performed close to *SRS*. However, a baseline strategy (Random Walk) was used, which occasionally failed to find all the victims in some experiments and where communication quality between rescue agents was lower by an order of magnitude. Takahashi et al. [54] assert that maintaining communication in rescue operations is an important research issue to consider and few works exist on the topic of communication that utilise ad-hoc networks. Caccamo et al. [65] demonstrate a novel robot navigation planner, in simulation, that is communication-aware. They propose a *"Resilient Communication-Aware Motion Planner (RCAMP)"* system to compute robot trajectories taking into account distance travelled, communication quality and environmental constraints. The system is to be used in USAR missions to sustain and repair wireless connectivity. The authors note from previous real-world experience that inherent limitations of wireless networks can compromise the success of a USAR mission. Most notably, as a mission progresses, the most consequential issue was that Access Points (APs) supporting communications needed to be regularly moved to re-establish communication. From the literature specifically on multi-robot communication there seems to be little focus on team formation and maintenance, which is why most predefined communication solutions in a physical environment fail according to Gunn and Anderson [42]. Gunn and Anderson [42] propose a framework that combines many of the methodologies and networks reviewed in Sections 2.5.1 and 2.5.2. The framework is for coordinating robots in a complex environment such as USAR. A human expert (operator) is required to initially assign tasks, set team roles and create team compositions after which the multi-robot team has the ability to perform team management, hold formation, task discovery and task assignment. Experiments are conducted using the *Stage* software simulator, which is modified to introduce simulated and unreliable communication between robots. However, the authors make assumptions for the simulated experiments which they remark would be difficult to carry over to any future experiments conducted in the physical environment. For the simulated experiment results, the framework shows improvement over the base case (random walk) in percentage of mission area covered and victim identification. However, even in the simulated environment and with the assumptions put in place, when there is high communication failure (80% message loss) the performance of all methodologies is poor. Gunn and Anderson [42] conclude with the following statement: *"An implementation in a physical environment using real robots would be valuable, as it would increase the difficulty of operation considerably"*.

The focus of Jensen et al. [41] is maximising coverage space and minimising cost while maintaining communication. Jensen et al. [41] use the Robot Operating System (ROS) along with the *Stage* software simulator package to test their *Sweep Exploration Algorithm* (*SEA*) for coverage of an unknown environment. The algorithm expands like a tree, and branches are explored on a sequential basis. The main use-case of their algorithm is for multi-robot team exploration in unsafe environments before human rescue teams to help plan rescue and prioritise tasks. The robots are assigned roles at the start of a mission, which is done dynamically unlike in the work by Meng et al. [40]. Robots move in a follow-the-leader fashion and use "repel-beacons", which have the main purpose of notifying other robots that a certain path has been completely explored. The exploration method using "repel-beacons" improves communication for the multi-robot team and has safety procedures put in place in case of a failure. The authors argue that the "repel-beacons" that are dropped are low cost, thus making them practical for use even in a physical environment. However, the authors note that in a physical setting there is more noise and the low cost beacons will require extensive evaluation. Moreover, there is a certain rate at which these low cost beacons fail. Notably the authors developed an algorithm, *Singularity-Robust Task-Priority* (*SRTP*), for a re-configurable ad-hoc network that creates a chain of interconnected "communication robots", denoted *nodes*, to achieve mobile robot exploration and basic obstacle avoidance while remaining in communication with a base station (they refer to this as a *MANET*). Anton et al. [66] look more at self-configuring (robot) networks, but include some aspects of creating formations. Experiments are limited as they are executed using a numeric simulator (two-dimensional) with robot-like agents. Furthermore, the algorithm assumes prior knowledge of the first and last antenna in the chain, the obstacle avoidance is primitive (i.e. performed on a single static obstacle) and the algorithm is yet to be tested in a more complex simulator with noise. Witkowski et al. [67] use low-power ZigBee communication (IEEE 802.15 standard) to create reconfigurable multi-robot *MANET*s, similar to Anton et al. [66] and Jensen et al. [41], in USAR based scenarios to allow for simultaneous communication between robots and first responders. In their work, some robots act purely as beacons for communication while others are used to localise in certain positions. Although the authors use a very restricted platform, they show promising results and use triangulation strategies to improve uncertainty in localised positions, which is useful information sent to the remote operator. The work by Rahman et al. [68] investigates optimal robot placement for relay locations to allow communication between a remote operator and a remote unit, which is disconnected. Their work is unique as the formations are created using a minimum spanning tree algorithm and are computed once and reused in case a remote unit is disconnected in more than one instance. This work is similar to that of Anton et al. [66], however it is more recent and experiments are done on simulated and physical robots, the latter of which have limited functionality. In the early work by Meng et al. [40] multi-robot clustering strategies are investigated, similar to *SRS* in [54], and extended to re-configurable networks, comparable to the work by Anton et al. [66]. However, the simulated and physical experiments by Meng et al. [40] are not identical (i.e. physical experiments are conducted in a smaller and more easily traversable map), which leads to contradictory results in performance of overall search time not highlighted by the authors. Furthermore, they assign robots with static roles, such as *host robot* and *search robot*, creating sub-teams. This presents common points of communication failure (i.e. *host robots*). Meng et al. [40] evaluate four main strategies for clustering and maintaining communication among the robot team, namely *Static Rally Point*, *Mobile Rally Point*, *Mobile Integrator* and *Mobile Integrator with Time-Out*. The work by Kashino et al. [69] looks at optimal predetermined delivery of static-sensor networks using multi-robot teams (MRTs) to cover an area to enable complete communication. The authors are motivated from the need to create network infrastructure in an infrastructure-less environment similar to the work by Reich and Sklar [70].

Finally, I end this section by briefly discussing unique solutions that allow communication in restricted environments. A unique yet simple implicit communication technique allows for information sharing only when team-members are in line-of-sight of each other using visual sensors (cameras) [71]. A modified single-robot coverage algorithm is used that produces back-and-forth motion, also known as Boustrophedon decomposition, in a multi-robot exploration scenario. The focus of the algorithm is on covering as much unexplored space in an environment as possible while reducing repeat coverage. This work [71] demonstrates a good example of how implicit communication could be used in multi-robot teams to help recover from situations when there is restricted or no explicit communication. In another work [72] a system is developed for coordinated information gathering in a team of robots that are tracking a target. Information is shared only periodically when the value of the information exchange is highest (reward based communication).

2.5 Broader Context and Rationale of the Communication Domain

Apart from *MR* communication there are two additional communication domains, i.e. human-human (*HH*) and human-robot (*HR*). I explore some of the research in each domain to examine the similarities in the broader context and highlight the importance of communication. The *MR* communication domain shares many similarities with those of the *HH* and *HR* domain, such as the use of ad-hoc networks for communication and the different methods of messaging, discussed in Sections 2.5.1 and 2.5.2.

2.5.1 Human-Human Communication

The *HH* communication domain is vast and has many complex categories, of which the most simplified premise is that humans use a system of communication (language) to fulfil an intent (goal). The work by Davidson and Noble [73], Kimura [74] and Fitch [75] look at the evolution of language from its inception and all show evidence that, as tools advanced and changed thanks to new technology, the language used for communication evolved and vice versa. The information age brought on many new technological advancements in tools and altered the way language was used. However, the most significant technological advancement of all occurred in 1983 with the birth of the internet, which was soon after largely improved by Tim Berners-Lee who invented the World Wide Web (WWW). The WWW effectively allowed for a common and simplified means of access to online information. The advancement of the internet and WWW greatly evolved how humans interacted with one another. Pioneering works on computer-aided *HH* communication appeared in the early nineties [76–78], which primarily investigate how online communication altered group cooperation and changed the use of language. Particularly in the latter work [78], the term *computer-supported social networks* is used to denote computer-aided *HH* communication. More recent examples of how technology has evolved and continues to evolve language are in the works by Keating and Mirus [79] and Oz and Leu [80]. The first work investigates how technology impacts language practices in the case for deaf users who use computer-aided video communication, and the latter work shows how Artificial Neural Networks (ANN), using a smart glove with sensors to extract joint angles between fingers from hand gestures, are able to translate up to fifty American Sign Language words successfully.

A review of research in online communication between cooperating groups of human users, showed that a lot of focus was put on investigating the improvement of communication during disaster scenarios. In disaster scenarios, communication infrastructure could be damaged or absent, which usually makes communication between victims and first responders impaired or impossible. Some works [81, 82] had common themes, such as analysing social networks during a disaster situation and the prevalent mode of communication in these situations, i.e. mobile phones. Mobile phones are part of mobile systems, which consist of multiple connected devices that are portable and allow for communication and access to information. Three tools for mobile communication applications are briefly reviewed [83–85]. The *"SUCRE"* (*"Supporting Users, Controllers and Responders in Emergencies"* [83]) application collects and analyses contextual information supplied by users during an emergency situation to reinforce information and infrastructure. The service provided by *"SUCRE"* requires regular online (WLAN) feed of information via a server. In summary, *"SUCRE"* contains components to improve communication and disseminate information, however it assumes that communication network infrastructure is unaffected by the emergency situation. Another tool introduced is *"Help Beacons"* [84] a mature, truly adhoc and research-intuitive system. However, it is very simplistic and lacks additional features, unlike the *"SUCRE"* application, and only provides adhoc communication to a basic level. The main drawback of this application is the dissemination of information, which is not as good or predictable as in later works [85] or as fine-tuned as *"SUCRE"* [83]. The greatest advantage of the *"Help Beacons"* application is its maturity and lightweight messaging compared to the rest of the works. Finally, a tool created to establish a *MANET* [85], which is more advanced than its predecessor *Help Beacons*. The *MANET* [85] based mobile application investigates the impact of using opportunistic ad-hoc networks to assist communication when network infrastructure breaks down. A special feature introduced allows any mobile phone to be selected as a *central device* (i.e. router), which connects other devices, caches data and disseminates information. Each mobile phone may become a *central device* or a client device. Data "propagates" along the chain of connected mobile devices and collected by *central devices*, which push it to other *central devices* until it reaches a device that is connected online. All the data is then offloaded online, as illustrated in Figure 2.11.

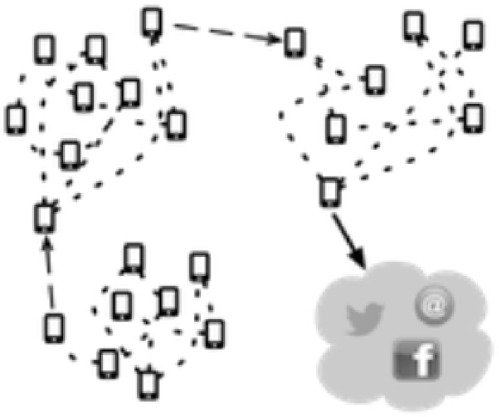


FIGURE 2.11: A diagram demonstrating how ad-hoc network based application connects offline client to an online server [85].

2.5.2 Human-Robot Communication

Humans in close proximity to each other rely on non-digital forms of communication, such as speech and gestures; there is much attention paid, in human-robot interaction and artificial intelligence, to the investigation of methods for robots to communicate in similar ways, using speech recognition, natural language generation and gesturing. *HR* communication could either be collaborative [25], or competitive [30]. In addition, robots will frequently be deployed in task domains where they are not co-located with humans, such as search-and-rescue, humanitarian de-mining or nuclear plant surveillance. Two works are briefly reviewed in the *HR* domain [26, 86] that focus on message propagation and message types in disaster scenarios, which are related to multi-robot communication. The work by Lujak et al. [27] investigates messaging and the concept of mobile devices assisting robots in the field to perform tasks efficiently.

Murphy et al. [86] research message types, content and the best and safest (ethical) form of communication between a robot, which is teleoperated by a human, and trapped victim(s). A possible solution to the problem is making the robot(s) completely autonomous in its decision-making and victim triage, thereof the core of the problem is shifted more toward single-robot to server and multi-robot communication. However, this solution is far too complex and may require direct human-robot interaction between robot and victim (*HRI*), which raises a new set of ethical issues, for example explaining actions to victims, requesting consent from victims, etc. The robot used in the field experiments is equipped with audio and visual sensors. Experiments are executed using four different 'interaction' schemes: *"Two-way Video with Two-way Audio, One-way Video (from Robot to Responders) with Two-way Audio, Two-way Video with no Audio, and One-way Video (from Robot to Responders) with no audio."* [86]. Furthermore, the authors do not consider communication issues in field experiments, as the robot to remote server communication is tethered through an extension cable. Nonetheless, the authors acknowledge that even in a controlled environment and with direct connection to the robot, communication was frequently delayed or video/audio feedback was intermittent. The interaction schemes with less or no video feedback minimised network connectivity issues, but at the cost of losing communication capabilities with the victim. The authors demonstrate that a robot needs to leverage the use of different sensors depending on the type of task it is required to perform, as wired communication capabilities are a major constraint.

The work by Zadorozhny and Lewis [26] use the Unified System for Automation and Robot Simulation (*USARsim*) for simulating their communication and experiments. *USARsim* is a high fidelity simulation software for robots and environments built with the widely used Unreal Engine. It is designed for the research and study of *HRI* and robot coordination. The authors explore how a new method of humans aiding ("crowdsourcing") in victim detection affect information fusion for robots in the field during urban search and rescue operations. The task of the robots is autonomous navigation and extraction of image sensory data of predicted victims. The task of the human operators is to confirm if image data contains victims. Sensor image data is processed and filtered by the robots in the field and then sent to human operators to be analysed. After multiple passes of the same victim by the robots and human operators the estimated victim coordinates begin to converge to true-coordinates, which the authors describe as "crowdsourcing". It is acknowledged that in a physical environment the communication constraints will differ, therefore it is likely that the performance of the human-robot team will be impacted.

Lujak et al. [27] leverage the network capabilities of surrounding smart devices (ambient intelligence) to prevent congestion and keep robots and all other devices connected. They propose the *"Robots-Assisted Ambient Intelligence" (RAmI)* system to assist robots with successful task completion (scheduling) and furthermore prevent (robot) network communication issues. The system will leverage a distributed network optimisation to assist individual users and also to minimise conflict and congestion in hot-spot areas where many users are located and resources are limited. They show their system design and use a case study to demonstrate its effectiveness.

2.6 Summary

In summary, all multi-robot systems require: a method of communication, a design decision on how messages are to be communicated and finally to maintain successful multi-robot operation; it is crucial to have in place a method to prevent communication issues. The related work shows that multi-robot coordination systems either do not consider communication, or consider communication but do not examine the associated communication issues. Moreover, the related works present the ad-hoc network as a popular form of communication for multi-robot teams. However, they are underutilised and physical experiments are rarely conducted with prevalent communication issues. This thesis investigates the performance impact that network perturbations have on a variety of performance metrics in MRTs. A new framework is built on top of the existing *MRTeAm* framework, introducing a messaging method and behaviours, which are devised to help reduce the impact of network perturbations. Moreover, I design a novel simulated network perturbation based on a machine learning method of signal strength (in the operational environment), which affect MRT communication. The performance of the simulated and actual network perturbation are compared and analysed, demonstrating how accurate network perturbations in simulated environments are compared to the physical environment. The final point of interest is to appreciate the negative and positive repercussions of using behaviour-based control to minimise the effects of network perturbations on MRT performance.

Chapter 3

MRComm: Multi-Robot Communication Testbed

3.1 Introduction

The overarching focus of this thesis is on analysing the impact that communication issues have on MRT performing coordination tasks. Of particular interest is investigating how this affects the design of multi-robot communication networks and organisation (centralised/distributed). I evaluate the *MRComm* and *MRTeAm* frameworks, and expand the collected performance metrics in Chapter 4. The focus of experimentation and design is on physical robots and the real environment, however I examine the performance of simulated robots and a simulated environment in Chapter 5. I evaluate empirical results using a team of physical robots in a feasible search-and-rescue environment where uncertainty and noise from external sources are present in Chapter 6. Moreover, in Chapter 6 I establish a physical ad-hoc network that is self-maintained by physical robots, and I test a novel high-level messaging protocol and employ a novel leader-follower behaviour that aims to guarantee effective mission completion in perturbed communication networks, the design of which is described here. As mentioned in Section 2.3.6, *MRComm* is a MRS framework based on *MRTeAm* [9, 45–47], which is capable of executing either simulated or physical robot experiments using the same functional robot agents across either type of experiment (i.e. the same robot functions and controls are used for both simulated and physical robots).

The *MRComm* framework is based on the *Robot Operating System* (ROS) [10], a collection of mainly robotics-based systems (packages) and a common middleware to allow these systems to communicate, further described in Section 3.2. There are two type of agents employed by the *MRComm* framework. Originally, *MRTeAm* employed the *auctioneer* agent to allocate tasks and guide the overall course of the experiment. However, the *auctioneer* agent's functionality and role is changed in *MRComm*. It is henceforth denoted the *task assigner* agent and is the first of two agents employed in *MRComm*. The *task assigner* agent uses the Round-Robin (RR) method [9] to initiate task assignment messages, which is described in Section 3.3. Furthermore, the *task assigner* agent for the experiments conducted in this thesis is a robot (virtual) agent, however it could also be a human operator. The second agent, denoted as the *task execution* agent, receives tasks (goals) to carry out, is responsible for navigating the environment and can be implemented with a behaviour accompanied by roles dependent on communication status. Moreover, it is responsible for sending and receiving team position and task status messages. Section 3.4 discusses how ROS communication works, what messages are shared between the agents in a mission scenario and the novel high-level messaging protocol used in *MRComm*. Section 3.5 describes the operational environment used for missions, the physical robot platform and the simulator. Section 3.6 briefly outlines improvements to the *task execution* agent. Section 3.7 describes the agents' purpose in a mission scenario and the behaviours used. The network parameters, Section 3.8, describe the network types used for communication by the robot team and introduce network perturbations. Section 3.9 describes and provides redesigned formulae for the general performance metrics. *MRComm* expands upon the original performance metrics captured by the *MRTeAm* framework, which are described on a per chapter basis. Finally, Section 3.10 defines the common experimental design that is used throughout the experiments.

The contribution described here builds upon the exploration of the multi-robot team coordination domain and specifically investigates the importance of reliable communication within this domain [87] and on methodologies to mitigate communication issues [88, 89]. A portion of the system design presented in this chapter was published in [89].

3.2 ROS: Robot Operating System

ROS is a middleware that contains many robotics-based systems (packages), which allow for the integration and communication between these and other systems. It contains software for both single-robot and multi-robot systems and is used in research and industrial settings. The foundation of *MRComm* is built using multiple different software packages that provide sensor fusion, mapping[1], localisation, path planning[2], messaging system[3] and multi-node communication[4]. All of the aforementioned systems, except the latter two systems, can be summarised in a single integrated framework known simply as *ROS Navigation*[5] (also ROS Navigation Stack).

3.2.1 Multi-Node Communication

ROS is inherently built as a centralised system (with aspects of decentralisation) comprising of shared name-spaces and directories initialised as nodes

[1] http://wiki.ros.org/map_server
[2] http://wiki.ros.org/dwa_local_planner
[3] http://wiki.ros.org/msg
[4] https://github.com/fkie/multimaster_fkie
[5] https://github.com/ros-planning/navigation

and services. A typical ROS hardware setup has either a single device (robot) or multiple devices (robots) connected and communicating via a single *roscore*. An example of this is illustrated in Figure 3.1. Furthermore, this could mean that multiple devices are connected by wire or wirelessly and are either on the same network or on multiple local networks, communicating in a cloud-type infrastructure. This has one obvious and major flaw, which is evident in Figure 3.1: if *Laptop 1* encounters an issue and the central *roscore* stops working then all connected devices will be disconnected and communication will cease.

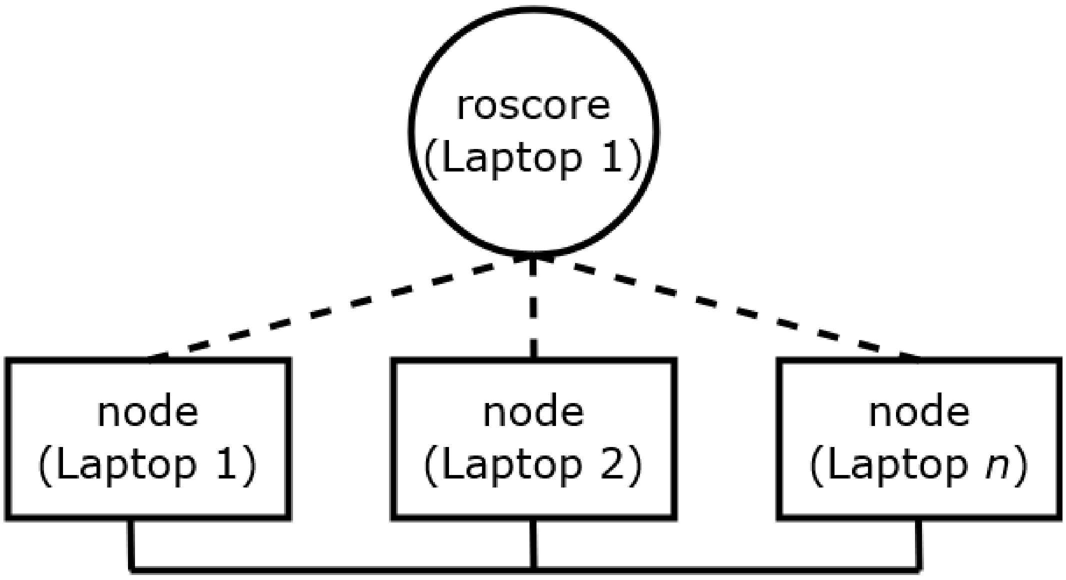


FIGURE 3.1: A simple ROS setup showing interconnection, of the *roscore* master managing the routing of communication (dashed lines) and the transmission of communication via topics and services (solid lines), between devices.

The ROS software package, *Multimaster FKIE*[6], hereon denoted as *FKIE*, introduces a multi-node (multiple *roscore*) system, which enables distributed robot networks. Effectively *FKIE* allows for each robot in a MRS to enact its own *roscore* master within a common network [11]. In particular, this means that each different robot is capable of running identical software packages, albeit with different parameters, without causing issues in the

[6]https://wiki.ros.org/multimaster_fkie

centralised name-space and directory within the *roscore* running on the same network. Moreover, *FKIE* theoretically allows for more flexibility with respect to system dependency on version-specific packages and even the overall ROS framework. *FKIE* is seamlessly integrated in the *MRComm* framework, which contributes toward a truly distributed MRT, escaping ROS's centralised nature and preventing the single-point of failure issue common with this.

A brief description of *FKIE* is presented here and is illustrated in Figure 3.2. An in-depth technical report is available by Juan and Cotarelo [11]. Each robot that is to be connected to the distributed MRT needs to be running its own *roscore* and additionally two *FKIE* nodes that will allow for the multi-robot communication. These two nodes are *master discovery* and *master sync*, and their functionality is summarised below. The master discovery node functionality is to:

- Periodically broadcast current *roscore* master to other potential ROS masters and detect any other *roscore* masters;
- Check for local changes to *roscore* master and update other potential *roscore* masters of changes.

Next, the master sync node functionality is to:

- Obtain information from other master discovery nodes and disclose topics and services to the local *roscore* master;
- Push information to the local master discovery node and update information on topics and services to the other *roscore* masters.

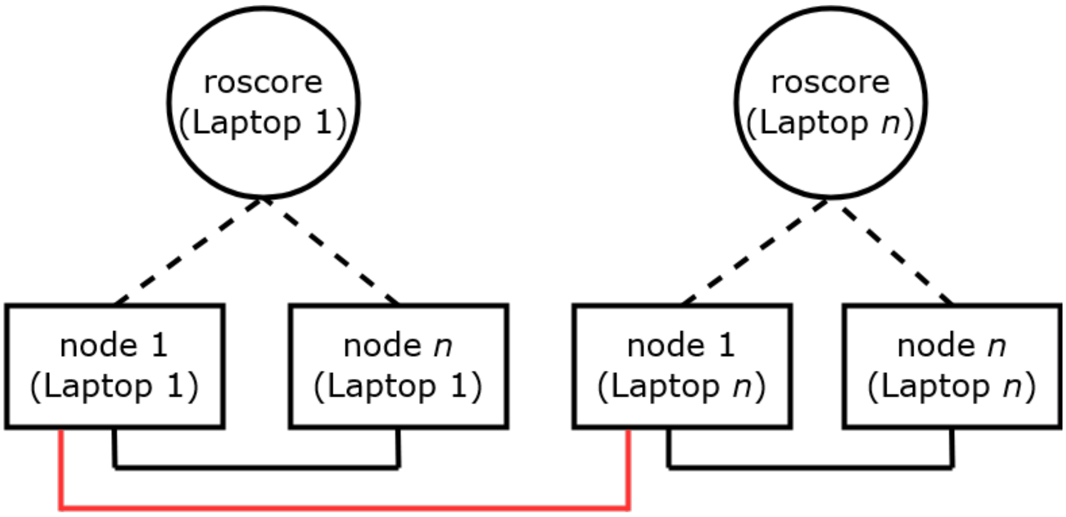


FIGURE 3.2: A ROS setup using FKIE to link nodes (topics) across multiple *roscore* masters (red solid line). Each individual *roscore* manages the local communication (dashed lines) and the communication of topics and services between devices (solid lines). This is my own depiction of a diagram from the report by Juan and Cotarelo [11]

3.3 Task Assignment

Initially, when designing the experiments for this thesis, it was observed that using multiple different task assignment mechanisms, as is the capability of the *MRTeAm* framework, would lead to contrasting results between experiments. The difference in results due to changes in the task environment needs to be minimised as much as possible. Therefore, the *task assigner* agent needs to always allocate tasks in a static and predictable manner. That is why the RR assignment mechanism, presented in Algorithm 1 by Schneider [9], is used to assign tasks to robots. In RR two lists are used, one containing the list of robot team-members r and the other an ordered list of tasks T. For each task $t \in T$ a cyclic iterative process takes place in which the next robot r_{next} will be awarded a task. At the end of this process, the tasks assigned to each robot are stored in their *agenda*[7].

MRTeAm incorporates additional auction mechanisms, apart from RR.

[7]The "agenda" is the list of tasks a robot has been assigned by the *task assigner* agent.

Algorithm 1 Round-Robin (RR) [9]

```
1: $ci \leftarrow 0$
2: procedure CYCLEITERATOR()
3:     $r_{next} \leftarrow r_{ci}$
4:     if $ci < (|R| - 1)$ then
5:         $ci \leftarrow ci + 1$
6:     else
7:         $ci \leftarrow 0$
8:     end if
9:     return $r_{next}$
10: end procedure
11: for all $t \in T$ do
12:     for 1...$t_{req}$ do
13:         $wr \leftarrow CycleIterator()$
14:         $T(wr) \leftarrow T(wr) \cup \{t\}$
15:         $T \leftarrow T \backslash \{t\}$
16:     end for
17: end for
```

However, using any of other auction mechanisms has two undesirable effects, such as:

- The potential of causing a change in the agendas of the robot team on a per experiment basis, i.e. tasks are assigned inconsistently to robots. For example, there are two predictable and simple cases for assigning tasks using the RR mechanism. Firstly, an even number of tasks will always be sequentially and evenly assigned to robots, and secondly an odd number of tasks will be sequentially assigned to robots, while the remaining tasks are wrapped around and assigned from the beginning of the list of robots.

- There is an overhead of time taken to process the different assignment mechanisms, denoted as *deliberation time* by Schneider [9], which will affect the performance metrics. In contrast, the RR mechanism has negligible effect (Figure 3.3) on the time taken to process the task assignment.

There is a similar algorithm to RR known as First Come First Serve (FCFS). It can be argued that the two algorithms are very similar, however examining the differences more closely shows that RR is preemptive and non-blocking and FCFS is non-preemptive and blocking. Preemptive algorithms would theoretically enable the *task assigner* agent to prioritise higher valued tasks to be assigned before those that are of lower value. However, the tasks used in experiments throughout this thesis have no priority value associated to them, so this is a non-issue. The current functionality of the *task assigner* agent and RR is that task assignment is performed at the beginning of a mission in "one-shot", which is denoted as non-blocking. For FCFS, the *task assigner* agent would need to assign each robot in the team a task and wait until a robot has completed a task and then assign tasks again, which is denoted as blocking. If FCFS were to be used, the blocking functionality would cause pauses in robot behaviour during mission execution, which would affect performance metrics, and it may potentially cause the overall agenda of a robot to change on a per experiment basis. This proves that using RR for the MRCP outlined in this research, eliminates uncertainty and time spent processing task assignment and focuses the analysis on task completion as a standard metric, rather than task assignment. Finally, it is important to note that once tasks are assigned to a robot's agenda, the ordering of tasks to be completed uses a simple greedy algorithm approach, e.g. in ascending order from the closest to the furthest task from the robot's current position.

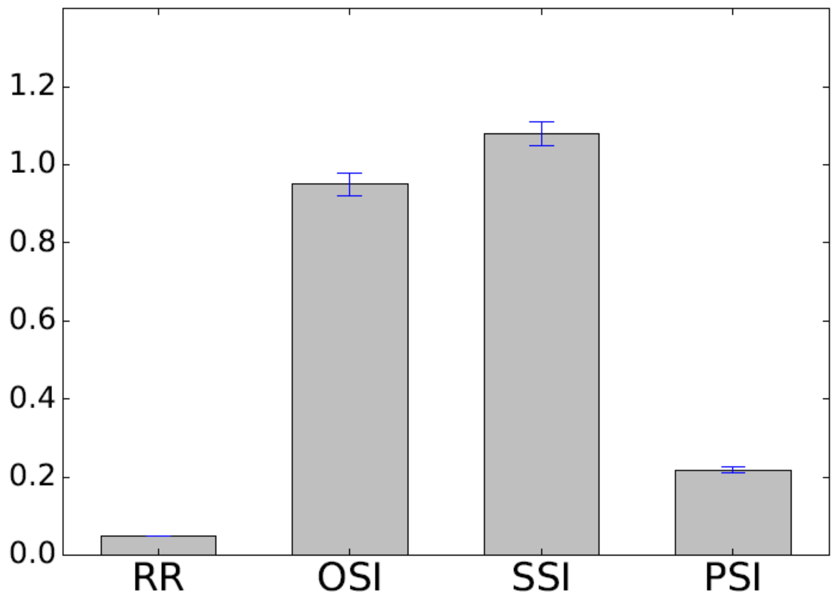


FIGURE 3.3: The deliberation time of the different assignment mechanisms incorporated by Schneider [9]

3.4 Shared Messages

The ROS transport layer typically uses TCPROS protocol but can additionally use UDPROS, which are both based on the standard variants of TCP and UDP respectively. The TCPROS protocol is used in *MRComm*. However, the transport protocol is only as good as the middleware used for the communication. The ROS communication middleware is very robust and universally used, however it is limited. It works as follows: a device that needs to transmit messages creates *"topics"* in which messages are published, also known as a publisher. A publisher queues messages and publishes them in the common network. However, there is no way for the publisher to target a specific device to receive messages and messages could be lost without the publisher ever knowing. A device needs to subscribe to the particular publisher (*"topic"*) to receive those messages, known as a subscriber. This is called the publisher-subscriber patter or simply *"pub-sub"* messaging. The advantages of such messaging are that communication is simple, reasonably scalable and robust (does not rely on network topology). However, it lacks network security, messaging can be slowed when many devices try communicating all at once (i.e. no traffic control and message surges can cause congestion), and there is no real-time messaging. Many of the these limitations make the standard *pub-sub* middleware, regardless of the transport protocol used, very similar in behaviour to the UDP protocol and very difficult to use for the specific experiments and requirements that *MRComm* is designed for. There is another possible ROS software service available that mitigates many of the issues that the standard *pub-sub* middleware has, and this is ROS *Services*[8]. Although ROS *services* is a good solution, it has its own existing issues. For example, a client could require a persistent connection to a service to update its local knowledge base, however the server providing the service may need to change the message being sent or may need to serve another client, which cannot happen as the server is locked in a persistent service call. Furthermore, apart from needing a persistent service, an important requirement is to be able to have direct access to the number of sent and received messages. To improve the accuracy of the message analysis, in this work it is conducted internally and is used to aid in interpreting communication quality and in communication processes.

The setup for the shared *topics* (messages) that are passed between agents within the *MRComm* framework and for any (other) experiments conducted in this thesis are:

- *task assignment messages* sent by the *task assigner* agent to *task execution* agents using the $\langle ST, SR, IT, EV, SA \rangle$ task composition, from Section 2.3.3, and the RR assignment, described in Section 3.3;
- *pose messages* provide robots' positions, which are sent by *task execution* agents to other *task execution* agents, as input to the task navigation process and to facilitate collision-free movement;

[8]http://wiki.ros.org/Services

- *task status messages* sent by *task execution* agents to all other agents, accompanied by sensor data acquired as part of the task, if applicable.

3.4.1 Improved Messaging: All-Or-Nothing

All-Or-Nothing (*AON*) is introduced and discussed here, which is a high-level improvement to the standard *pub-sub* messaging. The protocol of the novel *AON* messaging function is designed in a way that does not require any additional low-level changes to the transport protocol used by the *pub-sub* middleware, while offering best effort acknowledgement of message transmission.

The standard TCP protocol [90] inspired some of the functionality of *AON*, namely TCP's control and acknowledgement flags, which are important and necessary for continuous and robust communication even when communication quality is critically reduced. To keep *AON* as simple and non-disruptive as possible for communication, two high-level flags are introduced *received* and *finished*, which correspond to TCP's ACK (acknowledge receipt of message) and FIN (notification of final message) flags. This gives any shared message using the *AON* messaging function the ability to acknowledge receipt of messages and simple termination control. Only *task status messages* employ the *AON* messaging function as this is the most crucial shared message that requires to be delivered so that all agents have full knowledge of the mission. Algorithm 2 best describes the process of the *AON* message function, and partial code is available in Appendix A.2.

AON is fast and easy to implement within an already existing *pub-sub* middleware. The number of message propagation calls in the ideal case, per robot, made to the *AON* function is $n + 1$ (where n is the number of robots in the team). The total number of calls is $n(n + 1)$. It is important to note that this is not counting any repeat messages that are sent as a result of

Algorithm 2 AON Message Function

```
 1: INPUT msg, ri, messageReceived
 2: OUTPUT messageReceived
 3: teamSize ← 3
 4: receivedData ← new List[]
 5: msg                                          ▷ message being sent
 6: ri                                   ▷ the robot sending the message
 7: messageReceived ← False                      ▷ confirmation receipt
 8: procedure AON(msg, ri, messageReceived)
 9:     done ← False
10:     repeat ← 0
11:     receivedTime ← Time.now
12:     runningTime ← 0.0
13:     while not done do
14:         endTime ← Time.now
15:         runningTime ← endtime - receivedtime
16:         if not ri in receivedData[msg] then
17:             receivedData[msg].append({ri, messageReceived})
18:             procedure CALLBACK(msg, ri, messageReceived)
19:         end if
20:         if lengthOf(receivedData) = teamSize then
21:             messageReceived ← True
22:             repeat ← repeat + 1
23:             procedure CALLBACK(msg, ri, messageReceived)
24:         end if
25:         if runningTime > 15.0 seconds then
26:             procedure AON(msg, ri, messageReceived)
27:         end if
28:         if repeat > 1 and all(receivedData[msg]messageReceived = True)
    then
29:             done ← True
30:             return messageReceived
31:         end if
32:     end while
33: end procedure
34: rc                              ▷ robot sending/receiving message
35: procedure CALLBACK(msg, ri, messageReceived)
36:     if rc ≠ ri and not rc in receivedData[msg] then
37:         receivedData[msg].append({rc, messageReceived})
38:         return
39:     end if
40: end procedure
```

the *pub-sub* queue. The TCPROS communication protocol used for messaging provides adequate data throughput, lying on top of it, *AON* is designed to be as simple and light as possible and thus for reasonable size of n does not slow down the messaging system. However, if the multi-robot system is made up of too many robots (i.e. large n), *AON* will not scale well and the communication complexity and network congestion will increase considerably. In this work, the practical limits of the *AON* messaging function are note tested (i.e. tests up to $n = 3$). Moreover, as the size of the robot team increases there is a higher chance that messages will arrive out of order, which will (on average) double the number of propagation calls to fix the issue. If the total number of calls $n(n+1)$ take time t, then in the case for messages arriving out of order, the estimated increase in time will be $2t + m$ (where m is the time delay due to incorrect message arrival). The maximum value of m is equal to the time-out value of the *AON* function. During this time duration the *AON* function will keep trying to acknowledge if a message has been received by all team members and terminate once communication is successful. The time-out duration is chosen to be 15 seconds. This long period of time is chosen because it is firstly a fail-safe and secondly it is generally a rare occurrence that messages arrive out of order, at least for small values of n. Moreover, the long time-out period gives the robots a chance to catch up or slow down if they are in an unexpected state (desynchronised). The message propagation is demonstrated in Figure 3.4

To proceed in calculating the theoretical limit of n for *AON* the wireless technology and hardware limitations need to be known. The wireless network card used by the robots is the Intel AC-7265 [91], which has a theoretical maximum (receiving) speed of 867Mbps (megabits per second). However, the theoretical maximum speed can be achieved using multiple antennas, specifically 2 transmitters and 2 receivers or 2x2 802.11ac (MIMO). In this mode, the network card can theoretically receive 867Mbps, but has been reviewed to have significantly lower transmitting throughput near 150Mbps in ideal conditions. This gives rise to two problems for the theoretical calculation using 2x2 802.11ac. Firstly, the discrepancy between the received and transmitted throughput is too large, the maximum receiving throughput will be capped by the maximum transmitting throughput. Secondly, it is more unpredictable and not as stable over increasing distance, 802.11n is used for experiments. To make this calculation theoretically feasible and to compare it to the wireless communication used for experiments in this work it will be assumed that the maximum theoretical throughput is 72Mbps, both received and transmitted, using 2.4GHz 802.11n. The calculations will be performed in bytes, therefore 72Mbps translates to 9MBps (megabytes per second), as there are 8 bits in 1 byte. Each message in the *MRComm* framework is allocated 80 bytes and this allocation is recorded after every experiment to confirm there is no discrepancy in the transmitted message size. It is assumed that a medium TCP data size of 40 bytes is used to transport messages, therefore it is theorised that two TCP packets need to be sent per message. As mentioned, there are three types of shared messages, *task assignment*, *pose* and *task status* messages. *Task assignment messages* are allocated only at the start of an experiment before any other messages are transmitted, therefore they will not be factored into the theoretical limit as they are negligible. *Pose messages* are transmitted by each robot in the team, updated at a frequency of 5Hz and each message requires two TCP packets of 40 bytes, which gives the following equation $2n \cdot 5 \cdot 80 \times 10^{-5}$. For *task status messages* using *AON* it will be assumed that there will be messages arriving out of order. Finally, the message frequency of *task status messages* is set to 3Hz (not real-time messaging, i.e. no guarantee) and two TCP packets of 40 bytes are required per message, which gives the calculation $3 \cdot 2 \cdot 2(n^2 + n)$. Although *task status messages* use the *AON* method it is important to include *pose messages* in the calculation as the frequency of message transmission will impact the overall communication. Equation 3.1 shows the general calculation in the case that messages arrive out of order, as there is a higher probability of this occurring as more robots are added to the team, and when hardware and physical limitations are not taken into account.

$$9.0 \times 10^6 = 3 \cdot 2 \cdot 2(n^2 + n) \cdot 80 \times 10^{-5} + 2n \cdot 5 \cdot 80 \times 10^{-5} \tag{3.1}$$

Equation 3.1 gives the value of n equal to 30,618 robots. However, this value of n is highly improbable for many reasons, such as that on-board wireless cards cannot typically host many devices as they are not designed to perform router functions, the ROS *pub-sub* middleware is not designed for large device communication and has never been tested with such a large number of connected devices, and TCP is a protocol that requires good flow control and processing power, which as previously mentioned are not supported by ROS. Although the value of n from equation 3.1 is useful, as it shows that if network hardware limitations and efficiency were not factors, theoretically the size, number and frequency of messages being sent over the network do not limit *AON*'s scalability of the robot team size. However, a typical wireless router on a local network can handle 255 devices. I argue that the theoretical maximum limit of n for physical robot experiments is closer to the typical wireless router limit of 255 robots. But, even for 255 robots, whether in infrastructure or ad-hoc network mode, a more complex messaging design, sophisticated network architecture and highly optimised *AON* functionality are required.

3.5 Environment

The physical environment consists of an office space on the first floor of the Strand building, Strand Campus, King's College London. The simulated

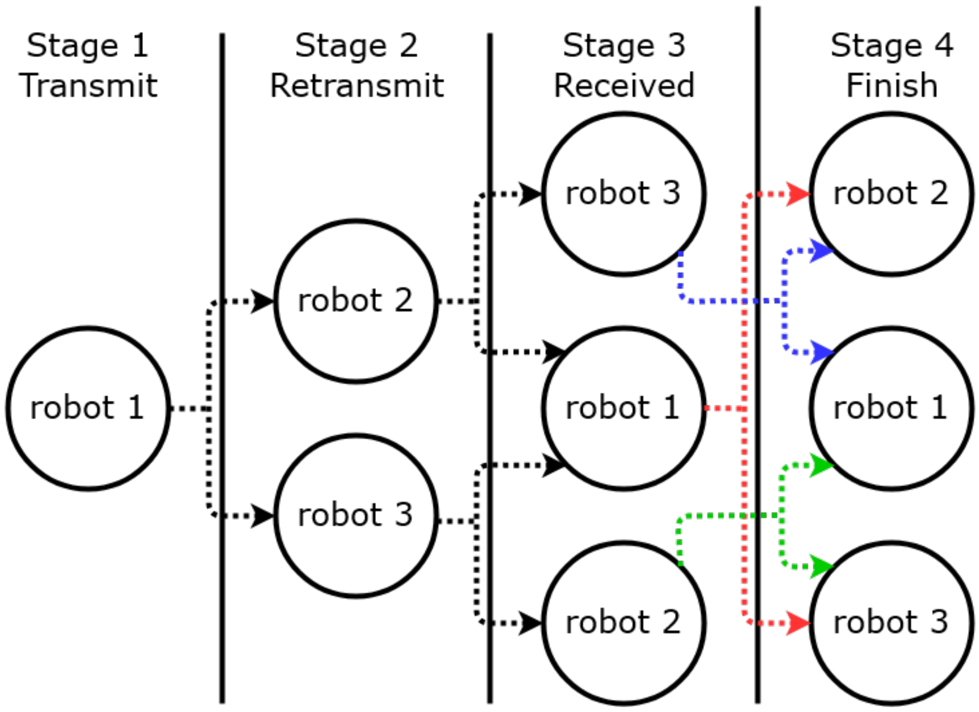


FIGURE 3.4: Message propagation is shown by dotted lines in the diagram. In stage 1 a status message is transmitted; in stage 2 team members retransmit the received message; in stage 3 the entire team knows that the message has been *received*. The *received* Boolean is set to true and sent to each team member (i.e. this is further indicated by the red, green and blue dotted lines by each robot, between stage 3 and 4); in stage 4 the message is received by all team members and communication is *finished*.

environment is modelled according to the floor plan of this office space. The corridor has multiple sets of fire doors, which typically stay closed. The physical experiments were therefore restricted to that portion of the corridor which could be reached without needing to open the fire doors. Moreover, there is a long arching hallway and multiple offices on both sides and one larger conference room that is of an irregular half-oval shape. Figure 3.5 represents the floor plan of the office space used. As mentioned previously, I use the campus Wi-Fi for the communication network of the robot team, which is a local instantiation of *eduroam*[9].

The motivation for using the office space is its similarity and significance to existing operational environments for MRS in urban search and rescue

[9] https://www.eduroam.org/

(USAR) and warehouse robotics. A multi-robot team performing a USAR scenario in this environment could be allocated points of interest or given potential victim locations to observe. Alternatively, a multi-robot team performing a warehouse scenario in this environment could be sent to pick up an order of packages that may or may not have certain constraints.

The robot team is placed in a central "safe" location awaiting deployment to predetermined task locations in case of a USAR scenario. As a robot reaches a task location, it could perform a unique function that no other robot on the team can, e.g. take a snapshot, use on-board end-effector to pick up payload etc. Furthermore, it is assumed the robot team is heterogeneous (i.e. each robot has specific functionality) and the environment has already been mapped, and that the map has already been disseminated to the robot team.

3.5.1 Physical Platform

As mentioned previously, *MRComm* defines the *task execution* and *task assigner* agents, while the *ROS Navigation* stack provides localisation and path-planning capabilities and the ROS based *FKIE*(Section 3.2.1) package provides the intra-team communication. *FKIE* is a newly integrated system to the *MRComm* framework and is an extension to the original *MRTeAm* framework. The *task execution* agent is executed on the Turtlebot 2[10] robot platform and a detailed diagram is presented in Figure 3.6. The robot base is the iClebo Kobuki[11], which has differential drive, robot vision is made possible with an RGB-depth camera (the Asus Xtion Pro[12], which is a clone of the Microsoft Kinect [92]), and an on-board laptop[13] performs all the processing (the Acer Travelmate B117[14] running Ubuntu 14.04). The robots[15]

[10]http://www.turtlebot.com/
[11]http://kobuki.yujinrobot.com/about2/
[12]https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/
[13]The *laptop* refers to all the processes that are contained within the MRS.
[14]https://www.acer.com/ac/en/GB/content/professional-series/travelmateb
[15]The *robot* refers to the Turtlebot 2 platform as a whole.

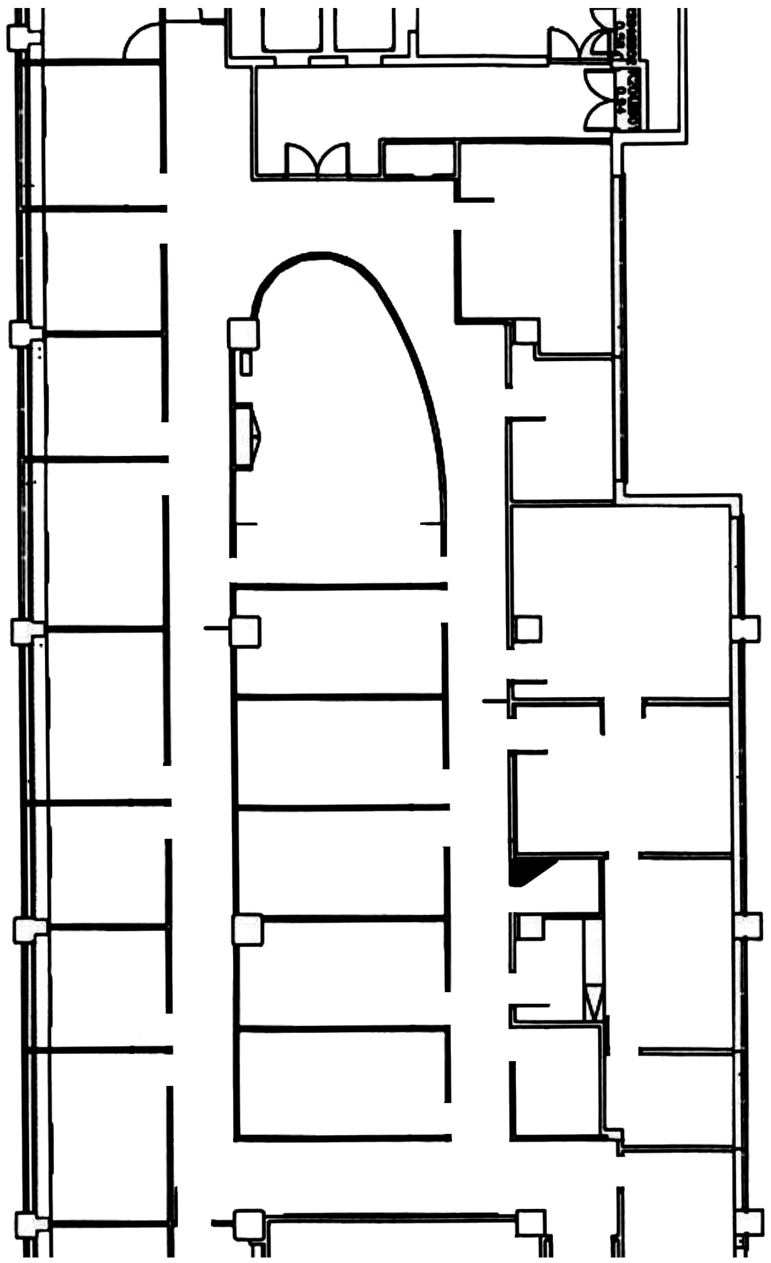


FIGURE 3.5: Office space (map) used for experiments (based on the actual floor plan of the building).

communicate utilising *FKIE*, which discovers other devices that are running ROS and synchronises any specified message topics. Once an experiment starts and the robots are assigned their tasks, navigation in the environment is autonomous. Moreover, the robots will attempt to avoid any unexpected (new) obstacles that appear in the environment.

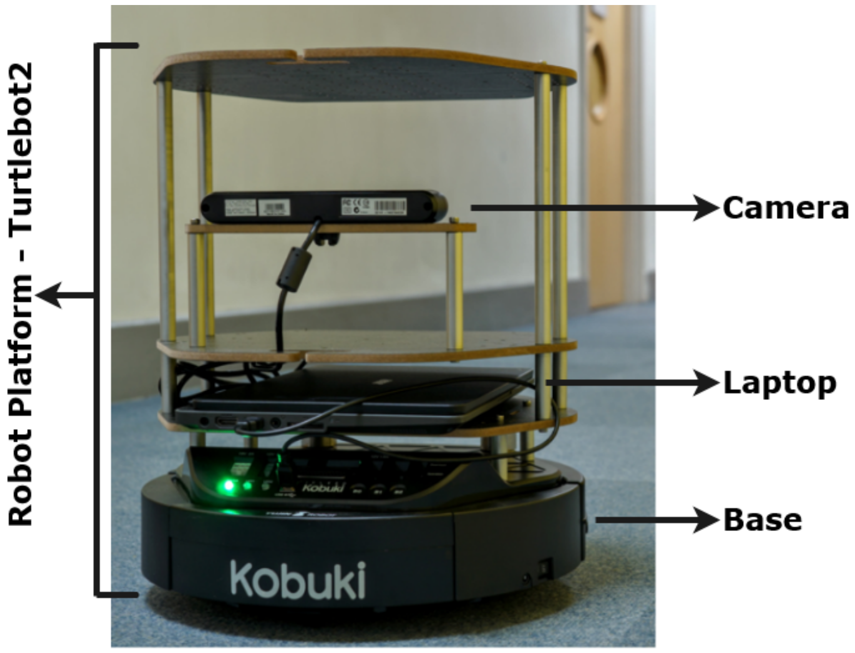


FIGURE 3.6: An overview of the Turtlebot 2 robot platform. The camera is used for detecting obstacles, but other functionality can be implemented. The laptop contains the agents and other processes that are needed for the functionality of the robot. The base provides mobility to the robot.

3.5.2 Simulation Platform

MRComm utilises the ROS variant of the *Stage* simulator [24] to simulate the robots. *Stage* is used for experimentation and research, and provides high fidelity simulation of sensors, sensor fusion and noise to the environment. Moreover, *Stage* includes an emulator for the Turtlebot 2 base and the particular sensors on the base that are used by the physical platform here. It is written in C++, which makes it fast and scalable for multi-robot system simulation. The work by Schneider et al. [45] shows that the difference between simulation and physical experiments is acceptable across most of the performance metrics evaluated.

3.6 Functional Robot Improvements

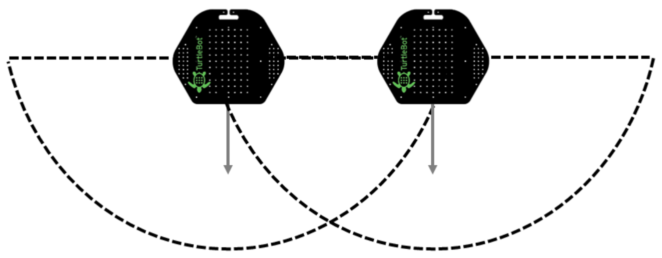
A number of improvements are introduced to *MRComm* to streamline the debugging and development process, help reduce inconsistency during mission execution and in experimental data collection. The initial change is updating the architecture software, the second is directionally aware collision avoidance and finally a simple dead reckoning recovery behaviour.

An extensive architecture change was performed to upgrade the agents from the older and more general *fysom*[16] finite state machine (FSM) to a more robot orientated FSM *SMACH*[17]. *SMACH* is a task-level architecture for the rapid creation of complex robot behaviour. It is a Python library independent of ROS, which allows for the building of hierarchical (nested) state machines. It allows for easy maintenance and very modular state implementation. Furthermore, it provides ROS-built FSM visualisation tools, which can be used for debugging individual states and state flow inspection. Both the *task assigner* and *task execution* agents were updated to use *SMACH* FSM. Moreover, the *task execution* agent implements a more complex hierarchical state machine, as shown in Figure A.3 in Appendix A.1.

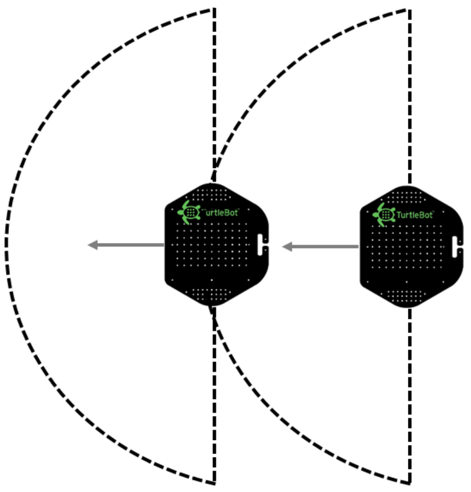
To make collision avoidance possible each *task execution* agent records a list of *pose messages* (positions), which is updated dynamically. The original *MRTeAm* framework that *MRComm* is built on has finely tuned system parameters for the robots to be able to perform their best possible navigation in a known environment. These parameters were further tuned for the larger and more dynamic office environment used for experiments conducted in this thesis. However, in early testing it was noted that the internal collision prevention system seemed to occasionally either miss near collisions, while robots were performing another action, or prematurely initiating collision prevention, which occurs when robots were moving simultaneously

[16]https://pypi.org/project/fysom/
[17]http://wiki.ros.org/smach

near each other or behind each other, illustrated in Figure 3.7. The original collision prevention approach used a predefined “danger zone”, which is a semi-circle in the direction of travel of a robot (3.7) and whenever a *task execution* agent receives a team member's updated position it checks if it falls in this zone. If a team member is in the “danger zone” then both robots will pause moving and the robot closer to its intended goal location has right of way (ties are broken randomly). This behaviour was unwanted, therefore collision avoidance for the *task execution* agent was updated to check the direction of heading together with the position of the team member and a new “event horizon” area was defined, Figure 3.8. The additional functionality allows for the improved collision avoidance to be triggered in two ways. The first way that collision avoidance is triggered is when the position and heading of a team member is received and the following conditions are satisfied: the team member is in the “danger zone” and is heading in the same direction or at an angle (i.e. perpendicular) to the other robot. Both robots pause and the right of way is given to the robot closer to its intended goal location, as described above. This improves the issues of reducing premature collision avoidance when robots are travelling in close proximity to each other or behind each other. However, it introduces a new problem. Consider the following event: a robot is travelling behind or at an acute angle in the same direction of a team member, which has to suddenly stop moving to initiate a new action. There are two possibilities, either the robots collide, as they are unaware of the sudden change in team member behaviour, or the robot behind could get stuck, as it comes too close to the team member and stops in front or behind it (i.e. if it is backed up and too close to a wall, ROS Navigation cannot recover). The second way that collision avoidance is triggered, which overcomes the above issues, is by using the “event horizon”. The “event horizon” area, represented by the red circles in Figure 3.8, is in close proximity to the robot's actual footprint (the dimensions of the physical/simulated robot in the environment). When a team member enters this area both robots immediately pause for a collision, each check if the team member is still within the "event horizon" and if they are, they perform the above described right of way check. The improved collision avoidance functionality gives the *task execution* agent more flexibility and accuracy when dealing with collisions.



(a) Shows that robots travelling in close proximity in the same direction can enter the "danger zone" even when there is no risk of collision.



(b) Shows that robots travelling behind each other in the same direction can enter the "danger zone" even when there is no risk of collision.

FIGURE 3.7: *MRTeAm* collision avoidance. The dotted semi-circle represents the "danger zone" of a collision risk.

ROS Navigation is not perfect and on occasion the system fails due to

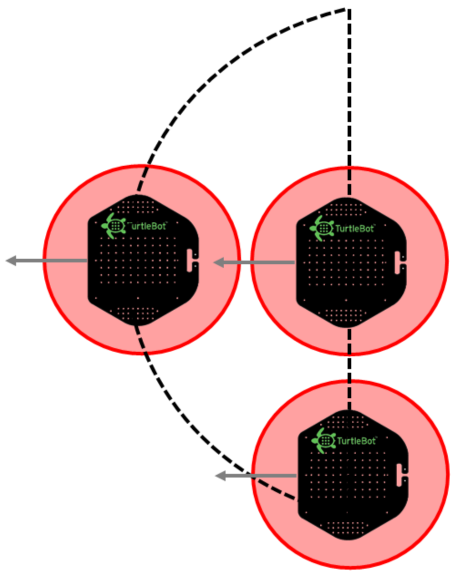


FIGURE 3.8: *MRComm* improved collision avoidance. The dotted semi-circle represents the "danger zone" of only a single robot and the transparent red circles represents the "event horizon" area for all the robots.

some internal error. Therefore, the *task execution* agent has the added functionality to check periodically whether the robot location is changing as expected. If a robot pauses for too long in the same location, due to a failed collision avoidance attempt or navigation failure, a time-out timer starts. At the end of the time-out, if the robot is not already stationary, it is forced to pause whatever it is doing, the *ROS Navigation* system is cleared and the robot uses dead reckoning to slowly reverse approximately half a meter. Dead reckoning is the "determined" position and velocity using only the estimate from the differential drive wheels, also known as odometry. After this brief process, the *task execution* agent will attempt to get back to the action it was attempting before the failed collision avoidance or navigation system failure.

3.7 Robot Agents and Behaviours

As previously mentioned, the *MRComm* framework employs two different robot agents, namely a *task assigner* and a *task executioner* agent, to conduct missions. The introduction of *FKIE* (Section 3.2.1) and the implementation of ad-hoc networking, have enabled communication and data recording to be distributed. Moreover, as discussed in Section 3.6, *SMACH* introduces modular and hierarchical FSM, which enables distributed architecture and control. As defined in Section 2.3.5, this makes *MRComm* a truly distributed MRS. Therefore, an agent (robot) is defined as one that is autonomous, functions independently from other agents and performs processes individually. However, the physical environment experiments require that the *task assigner* agent be "integrated" with one of the *task execution* agents on a single robot platform. The "integrated agents" functionality developed in *MRComm* satisfies the system requirement to maintain true agent independence (truly distributed). Even though two agents are running on the same robot they communicate and function completely independently of each other.

MRComm compares two different robot behaviours: a baseline *No-Behaviour* (*NB*) and a novel *Leader-Follower* (*LF*) behaviour, which is designed to respond to and maintain communication regardless of network type and perturbation. Furthermore, both *NB* and *LF* use the standard *pub-sub* communication protocol, however *LF* additionally uses the novel *AON* message function.

The *task assigner* agent is responsible for loading a mission configuration by distributing tasks (using RR assignment 1, as described previously in Section 3.3) to *task execution* agents sequentially. Every experiment is started in this fashion, however the network type used determines which behaviour the robot team[18] will inherit. The *task assigner* also records, without interfering, received experiment and team messages. The *task assigner* agent loads a mission configuration the same way for both *NB* and *LF* behaviours. However, the *task assigner*'s recording functionality changes between the two behaviours. For *NB*, the *task assigner* simply records the received *task status messages*. However, the *AON* message function, described in 3.4.1, introduced to the *LF* behaviour, alters the way messages are communicated. The *AON* messaging methodology forces the robot team to continue sending a message until the entire team confirms receipt of that message. The *task assigner* records all the messages, including repeated messages before the final confirmation receipt. Therefore, at the end of an *LF* experiment the *task assigner* checks its records to make sure each *task status message* has actually been received by all agents. A single FSM, Figure 3.9, is used to represent the *task assigner* that describes the above process, because it performs the same function no matter the *task executioner* behaviour that is being exhibited during experiments. During the "Load" state the task list is read from file and converted to a list of tasks that can be presented to the robots, after which a state transition occurs by identifying the amount and name of the robots in the team. The "assign" state is described in Section 3.3 and contains two possible outcomes, which is either complete, the *task assigner* begins to idly wait for the experiment to end, or assignment failure, the experiment is terminated and the output is "Incomplete". There are three possible transitions in the "Idle" state. The first transition is wait, it means not all robots have completed their tasks and have not sent the final signal. The second is experiment failure, this usually means a hardware failure or unrecoverable error has occurred in one of the robots and the experiment is terminated as before. The final transition is experiment success, which returns the output "Complete", this signifies that the experiment successfully

[18]Unless otherwise specified, **robot team** refers to all agents acting in a mission (i.e. *task assigner* and *task executioner*).

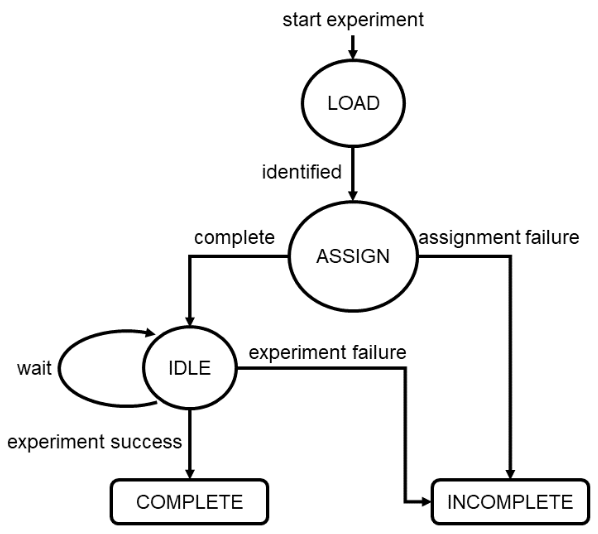


FIGURE 3.9: Finite State Machine of the *task assigner* agent (complete FSM in Appendix A.1)

terminated with all robots either completing their tasks or have reached the final state of the *task executioner* FSM (e.g. in the case that communication failed).

Much like the *task assigner*, the *task executioner* has some basic functionality that is the same for both behaviours. For instance, a *task executioner* is responsible for receiving tasks and navigating to them, using many of the base systems mentioned in 3.2. The *task execution* agent is initialised with parameters for behaviour (discussed below), network type (3.8.1), network perturbation (3.8.2) and scenario (3.10) at the start of an experiment. Thereafter, it receives tasks from the *task assigner* and begins to execute them. The *task execution* agent is also responsible for announcing to all agents when it has completed its tasks, which is important for the *task assigner* that is recording messages. Furthermore, it sends continuous update messages of received and completed (or failed) tasks back to the *task assigner* and other *task execution* agents. This functionality is open-ended and in future work may accommodate fault tolerance for critically reduced communication or robot failure. The in-depth explanation and supporting algorithms of how the *task executioner* adjusts when executed with either the *NB* or *LF* behaviour is shown in Sections 3.7.1 and 3.7.2, respectively.

The robot behaviour (i.e. *NB* and *LF*) is influenced by the network type that the robot team is currently connected to, for example the WLAN or AH network discussed in 3.8.1. However, for the purpose of the investigation done in this thesis , the network type is set at the start of a mission and kept static throughout the duration of each experiment. Furthermore, this requires the robot behaviour to be statically set, rather than dynamically changing, at the start of a mission.

3.7.1 No-Behaviour (NB)

A mission initialised with *NB* modifies the actions (or lack thereof) that agents exhibit when subjected to different network parameters. The robots executing *NB* do not adjust their behaviour based on network quality. They attempt to complete their assigned tasks, disregarding network parameters or loss of communication, and perform standard navigation and obstacle avoidance behaviours.

Algorithm 3 is a simple single procedure abstraction of the *task executioner* functionality for *NB*. A more detailed and supporting abstraction of the *NB* FSM is illustrated in Figure 3.10. The algorithm takes as input the *robotAgenda* and the output is a boolean flag (*completedAgenda*), which is used to verify if all the tasks have been completed in the agenda, and a recursive call to the algorithm itself (NA). Initially, the closest task to the robot's current position is found (*argmin*) and stored as task *t*. There is a single main loop, which checks if the robot agenda is empty, i.e. all the tasks have been done, and navigates the robot to task *t*. There are three conditions that are checked with every iteration. Firstly, if the robot has arrived at *t* and it contains more tasks in robot agenda, it will remove *t* from the agenda and recursively call NA with the updated robotAgenda. Secondly, if the robot arrival status is unknown (i.e. hardware or software failure) and tasks remain in *robotAgenda*, then task *t* is not removed and NA is recursively called. This will attempt to force the robot to navigate to the incomplete task *t* again, however the same task *t* will not necessarily be called again, as *argmin* for each task is re-calculated. The final condition is the stop condition, which checks if the *robotAgenda* is empty and returns *completedAgenda* as True. It should be noted that *completedAgenda* returning False will end the experiment and flag that it was unsuccessful.

Algorithm 3 No-Behaviour (NB)

```
1: INPUT robotAgenda
2: OUTPUT completedAgenda, NA
3: robot                              ▷ the robot initialised with NB behaviour
4: robotAgenda                                    ▷ the list of tasks of the robot
5: $t \in$ robotAgenda
6: procedure NA(robotAgenda)
7:     $t \leftarrow argmin$(robotAgenda)      ▷ task $t$ with shortest distance to robot
8:     completedAgenda $\leftarrow$ False
9:     while not robotAgenda empty do
10:        robot goto $t$
11:        if robot arrived $t$ and not robotAgenda empty then
12:            robotAgenda.remove($t$)
13:            goto procedure NA(robotAgenda)
14:        else if not robotAgenda empty then
15:            goto procedure NA(robotAgenda)
16:        end if
17:        if robotAgenda empty then
18:            completedAgenda $\leftarrow$ True
19:            return completedAgenda
20:        end if
21:    end while
22: end procedure
```

The FSM, Figure 3.10, is also used to clearly define *NB*'s focus on "moving" to tasks and providing "collision" and "recovery" actions. Moreover, when a robot reaches the "arrived" state it does not necessarily entail that the task has been reached, and if it is flagged as failed then that task will be returned to the agenda and either a new task or the same task will be selected, depending on *argmin* as discussed above. This implies that a task should never fail as a result of a robot failing to reach the task (i.e. hardware or software issues), but instead it can only fail if the task status cannot be communicated successfully. The same is true for the *LF* behaviour.

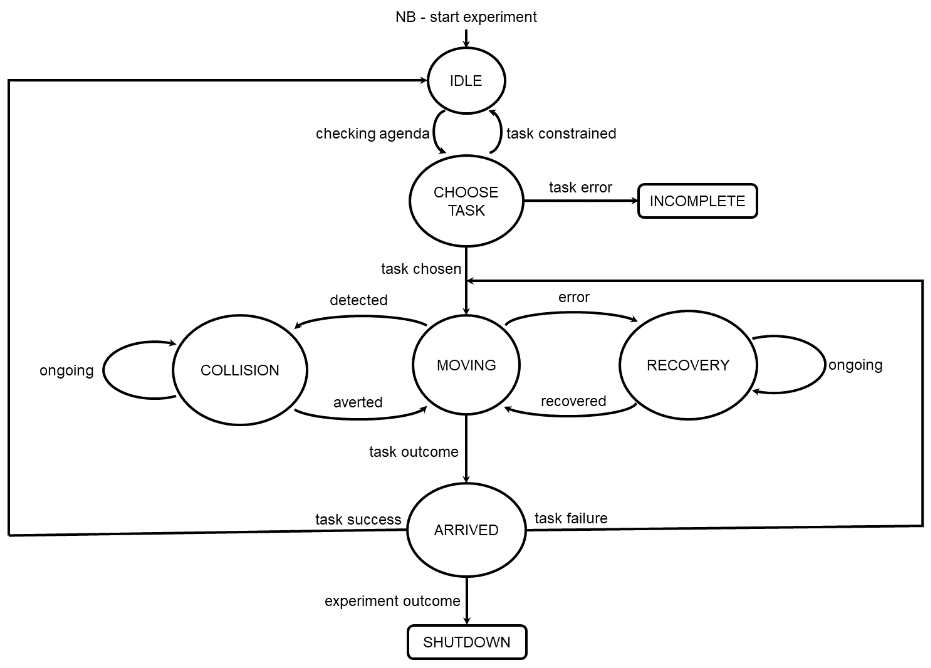


FIGURE 3.10: Finite State Machine abstraction of No-Behaviour (complete FSM in Appendix A.1)

3.7.2 Leader-Follower (LF)

The *LF* behaviour is inspired by the *AH* network type, which enables *task execution* agents to react to changes in simulated/actual signal strength or distance of other *task execution* agents. The *LF* behaviour reacts to changes differently, depending on the type of network perturbation being analysed. If the network perturbation used is *Simulated Loss Threshold* (*SLT*), introduced in Chapter 5, then *LF* reacts to the other agents' change in distance. However, if the network perturbation used is *Simulated Signal Degradation* (*SSD*), introduced in Chapter 5, then *LF* reacts to other agents' predicted signal strength. Finally, if the network perturbation is *Effective Signal Degradation* (*ESD*), introduced in Chapter 6, then *LF* reacts to other agents' actual signal strength. The change in signal strength/distance is detected as the robots move away from each other, triggering the action of regrouping, which is characterised as *"switching"* (described below), to maintain communication. Another functionality added to the *LF* behaviour is improved pub-sub messaging, as described in Section 3.4.1.

LF contains three roles: *not assigned (NA)*, *leader* and *follower*. Initially, all robots are *NA*. Upon the event of disconnecting from a team member, the *task execution* agents are alerted and begin to dynamically switch roles to either the *leader* or *follower* role, based on a utility score u, defined in equation 3.2.

$$u = d_{score} \times n_{incomplete} \times rec_{complete} \tag{3.2}$$

where:

- d_{score} = distance score, computed as $1/distance\_to\_goal$ (task location);
- $n_{incomplete}$ = number of incomplete tasks remaining on the robot's *agenda*, which is computed as the total number of tasks assigned minus the number of tasks completed;

- $rec_{complete}$ = 0.5 if the robot has just completed a task or 1.0 if it has not (this value is reset with every change in role and/or completion of a task).

The $rec_{complete}$ variable acts to balance out the priorities of tasks amongst the team. This is because the *follower* behaviour prioritises staying in communication with team members over completing its assigned tasks, whereas the *leader* robot prioritises completing its current task. In the general case, this prevents bias from occurring in role assignment, for example having the same robot as leader multiple times. Effectively, this factor ensures that all tasks are given priority at some point during the mission. The robot with the highest u score is selected as *leader* each time a disconnect event occurs. Moreover, there can only be one *leader* at a time, but multiple *followers*. In the simulated experiments, the *leader* is a proxy for the robot that could potentially initialise the ad-hoc network in a physical setup. The *follower* robots will connect to this ad-hoc network. Finally, when the *leader* has completed its goal, the behaviour clears all robots of their roles and returns (i.e. *switches*) to *NA*.

The Oxford English dictionary [93] defines *"Switch"* in Computing as *"A program variable which activates or deactivates a certain function of a program."*. In the *LF* behaviour, *switching* describes the internal process that occurs when a robot is about to disconnect from a team member. *LF*'s design is object orientated and the procedure for role switching creates an entirely new *task execution* agent object for the required role (i.e. *leader/follower*), which is encapsulated within the existing *task execution* agent. This newly created object is destroyed once the current *leader*'s task is complete and then the original *task execution* agent, which is implemented with the *NA* role, takes over. Thus, the motivation of referring to *LF* as *dynamically switching behaviour*.

The *task executioner* functionality for *LF* has been split into algorithms 4 and 5, as the behaviour is more complex and more steps are required to explain it compared to that of (*NB*), which is easily explained with a single algorithm. Moreover, no algorithm for the NA role is supplied for *LF*, because it is very similar to that of *NB*'s NA role in algorithm 3. Albeit with the extra functionality of detecting when the robots are about to disconnect, as depicted by the FSM for *LF*, Figure 3.12.

Algorithms 4 describes an abstraction of the ASSIGN procedure. The input for the algorithm is the current robot that needs to be assigned a role and the output is calling (assigning) either the leader or follower procedure (FSM), based on the collected utility scores. The leaderFound flag is preventative in case more than one robots have the same maximum score, to ensure the first robot that is assigned leader will be the only one. Firstly, the utility score is calculated with equation 3.2 and a loop is used to iteratively send and receive utility scores until each robot has collected all the scores from the team, which is achieved by checking the amount of utility scores obtained against the team size. After all the utility scores are collected, the maximum score is found and the associated robot is stored by the entire robot team (i.e. *robotLeader*). Thereafter, the process of assigning roles begins, if the current robot is the *robotLeader* then the LEADER procedure is called, and if not then the robot is assigned as a *robotFollower* and the FOLLOWER procedure is called instead.

Algorithms 5 contains three procedures. The LEADER procedure input is the ID of the *robotLeader* and the output is the NA procedure (i.e. the goal was reached) or to recursively call the LEADER procedure, and in its functionality it contains two important loops. The first loop sends the goal location to the leader enabling it to move, while checking if it is about to disconnect from a follower. If the leader disconnects from a follower the second loop begins, which constrains the leader to wait, while checking the distance to the followers until they are within connection, and thereafter the

LEADER procedure is recursively called. The FOLLOWER procedure takes as input the ID of the *robotFollower* and the *robotLeader* is already known and the output is either procedure NA, IDLE or recursively calling the FOLLOWER procedure. Furthermore, this procedure has a single loop, which simply goes to the last known position of the leader, and conditional checks are performed to decide if the follower should go to a new position of the leader, stop following the leader and resign roles or if the follower has no more tasks in its agenda it will idly wait. The IDLE procedure input is the ID of the robot that is idling and the output is the ASSIGN procedure. The single loop checks if any other robot is about to disconnect while the robot is idling. If the idling robot has sensed a disconnection, it is assigned with a utility score of 0 and appended to the list of utility scores, this is to reinforce that the idling robot has completed its agenda of tasks and will guarantee it is assigned as a follower. Partial code is available for the Role Assign procedure in Appendix A.2.

Algorithm 4 Role Assign (LF)

```
 1: INPUT robot
 2: OUTPUT LEADER, FOLLOWER
 3: robot                                  ▷ the current robot initialised with LF
 4: utilityScores ← new List[]  ▷ collect utility scores and associated robots
 5: leaderFound ← False
 6: teamSize ← 3
 7: procedure ASSIGN(robot)           ▷ assign roles based on utility score $u$
 8:     calculate score $u_{robot}$
 9:     utilityScores.append($u_{robot}$)
10:     while lengthOf(utilityScores) < teamSize do
11:         send $u_{robot}$
12:         receive $u_{robot_i}$                          ▷ $robot_i \in$ robot team
13:         utilityScores.append($u_{robot_i}$)
14:     end while
15:     $utility_{max} \leftarrow argmax$(utilityScores)
16:     robotLeader ← $utility_{max}$(robotID)
17:     if robot = robotLeader and not leaderFound then
18:         leaderFound = True
19:         goto procedure LEADER(robotLeader)
20:     else
21:         robotFollower ← $utility_j$(robotID)          ▷ $utility_j \in$ utilityScores
22:         if robotFollower $\neq$ robotLeader then
23:             goto procedure FOLLOWER(robotFollower)
24:         end if
25:     end if
26: end procedure
```

Algorithms 4 and 5 alone are not enough to describe the structure and process of *LF*. Therefore, a diagram and multiple abstractions of FSMs are used to describe the hierarchical structure and processes of *LF*. The diagram in Figure 3.11 depicts the hierarchical FSM of *LF*. It starts with the *NA* (FSM) role, which points to the switching process, denoted network, that is triggered by the network parameters set for the experiment, which has two outbound arrows, one pointing to the *leader* and the other to the *follower* roles (which are sub-FSMs). The hierarchical FSM in Figure 3.11 is in a closed loop, as it can be seen that *leader* and *follower* have outbound arrows going back to *NA* (FSM). The first FSM in Figure 3.12 is similar to that of *NB*, however with the included "network" process, which is the trigger to switch the *task execution* agent to a new role (FSM). The other two

Algorithm 5 Leader, Follower and Idle (LF)

```
1: INPUT robotLeader
2: OUTPUT NA
3: disconnected ← False
4: networkPerturbation ← limit        ▷ threshold depends on perturbation
5: procedure LEADER(robotLeader)                ▷ robot assigned as leader
6:     $t$ ← $argmin$(robotAgenda)     ▷ task $t$ with shortest distance to robot
7:     while not disconnected do
8:         if networkPerturbation > limit then
9:             disconnected ← True
10:        end if
11:        robotLeader goto $t$
12:        if robotLeader arrived $t$ and robotAgenda empty then
13:            goto procedure IDLE(robotLeader)
14:        end if
15:        if robotLeader arrived $t$ and not robotAgenda empty then
16:            robotAgenda.remove($t$)
17:            goto procedure NA(robotAgenda)
18:        end if
19:    end while
20:    while disconnected do
21:        if networkPerturbation < limit then
22:            disconnected ← False
23:            goto procedure LEADER(robotLeader)
24:        end if
25:        wait
26:    end while
27: end procedure
28: procedure FOLLOWER(robotFollower) ▷ robot(s) assigned as follower
29:    while robotLeader $\neq$ arrived do
30:        robotFollower goto robotLeader
31:    end while
32:    if robotFollower arrived and not robotLeader arrived then
33:        goto procedure FOLLOWER(robotFollower)
34:    end if
35:    if robotLeader arrived and robotAgenda empty then
36:        goto procedure IDLE(robotFollower)
37:    end if
38:    if robotLeader arrived and not robotAgenda empty then
39:        goto procedure NA(robotAgenda)
40:    end if
41: end procedure
```

Figures 3.13 and 3.14, depict the *leader* and *follower* FSMs, respectfully. Figure 3.13 is unique as it clearly highlights the priority of the *leader* role, which

```
42: procedure IDLE(robotID)                ▷ robot has completed its agenda
43:     while not disconnected do
44:         if networkPerturbation > limit then
45:             disconnected ← True
46:         end if
47:         wait
48:     end while
49:     if disconnected then
50:         $u_{robotID}$ ← 0
51:         utilityScores.append($u_{robotID}$)
52:         goto procedure ASSIGN
53:     end if
54: end procedure
```

is to complete its current task with the auxiliary action of pausing for *followers* to catch up. Similarly, Figure 3.14 highlights that the *follower* robot's focus is on the "leader position" and "moving" toward that position, as can be seen by the two arrows pointing in opposite directions connecting these states. What is not immediately clear is that *followers* in the "moving" state periodically monitor the distance between themselves and the *leader*. For example, if the *leader* is too far, the *followers* get more frequent updates of its position or if the *leader* is too close they go to the "waiting" state. Moreover, when in the "moving" state the navigation planner is not directly restricted by the network parameters and may try to re-plan a route outside of the connection area. However, *MRComm*'s network perturbation process does not allow this to happen, this is achieved by alerting the robot that it will disconnect and forcing it to re-plan a direct route to the *leader*.

In order for *switching* to be triggered, *task execution* agents using *LF* react to either a distance (i.e. *SLT*) or signal strength threshold limit (i.e. *SSD* or *ESD*), which could be either static or dynamic depending on the network perturbation that is initiated at the start of a mission, signal thresholds will be covered in more detail in later chapters. The baseline network perturbation for *LF* is *SLT*, which has a static distance threshold limit of approximately 4.0 meters. However, it could be represented by a dynamic signal

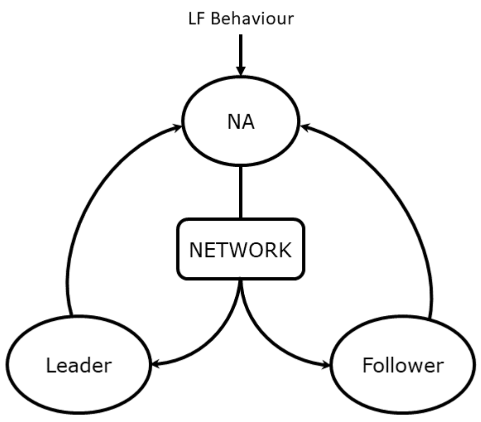


FIGURE 3.11: A diagram illustrating *LF*'s hierarchical FSM structure

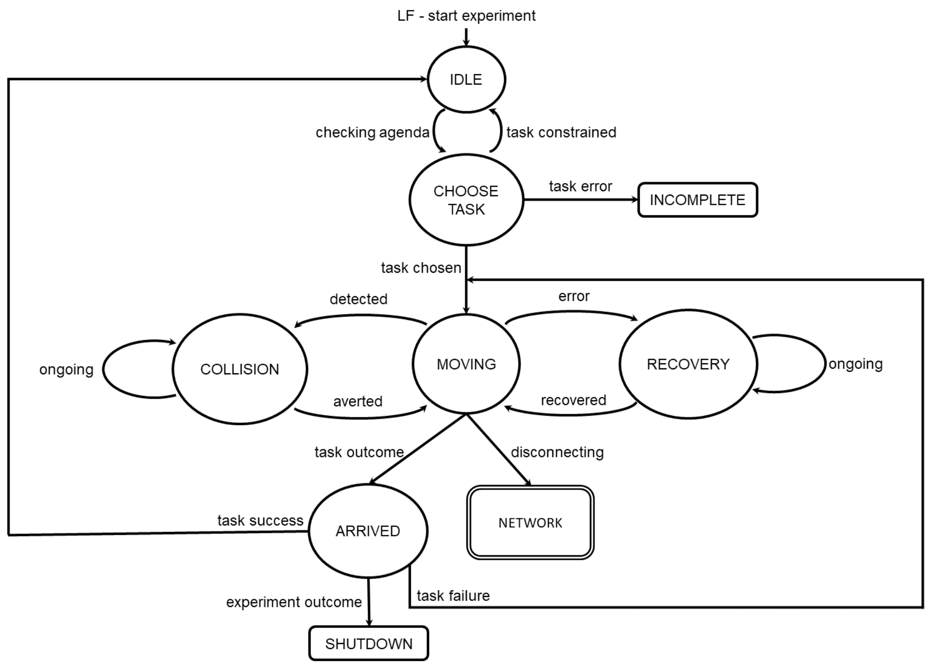


FIGURE 3.12: *LF task executioner* FSM for the *NA* role (complete FSM in Appendix A.1)

strength value, e.g. using either *SSD* or *ESD*.

Generally, robots using *LF* should not be running missions with the *WLAN*

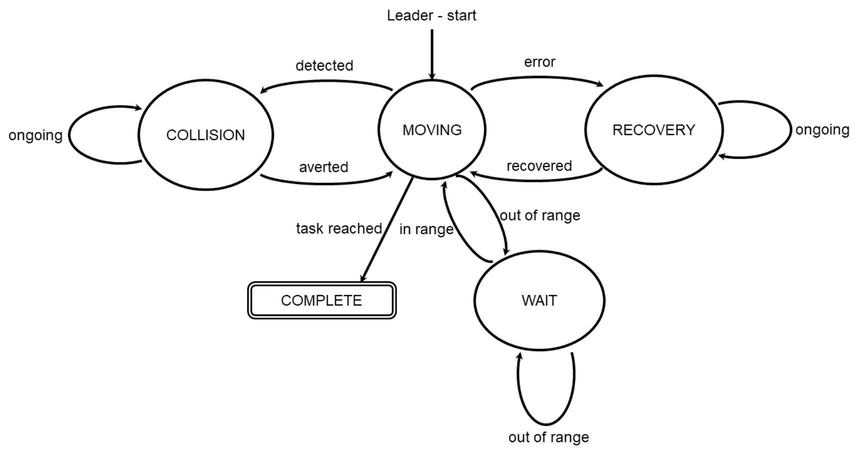


FIGURE 3.13: *LF* FSM for the *Leader* role (complete FSM in Appendix A.1)

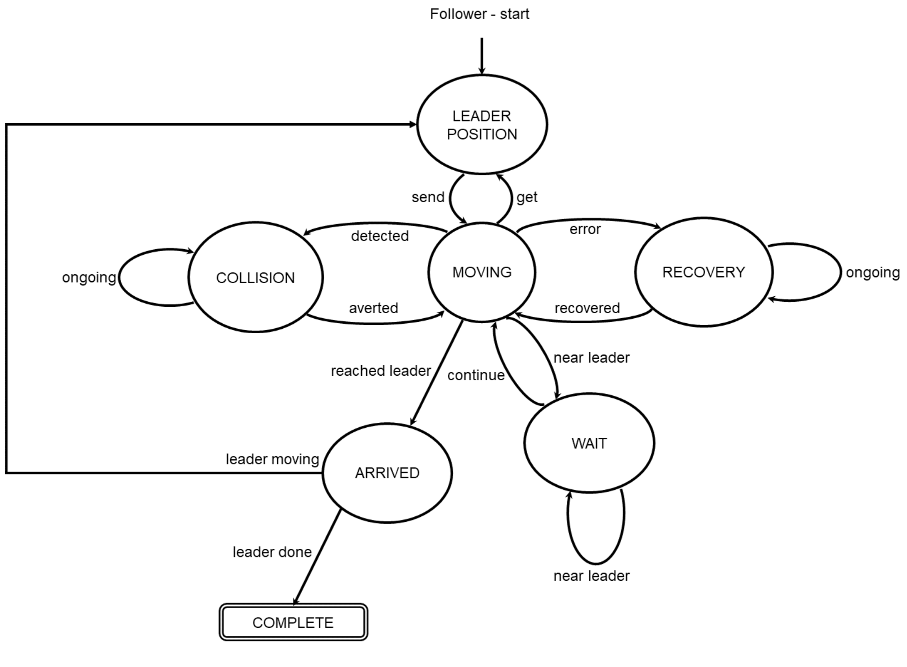


FIGURE 3.14: *LF* FSM for the *Follower* role (complete FSM in Appendix A.1)

network type for two reasons. Firstly, for the *WLAN* network type it is **assumed** that there is complete and uniform radial coverage of the environment, as described in Section 3.8.1. Secondly, that by design *LF*'s *switching* mechanism will **not** trigger, by virtue of the first reason, as the signal strength will never go below the threshold. However, the *SLT* network perturbation, described in Section 5.2, is designed for the purpose of enabling direct comparisons between any network type and environment (simulated/physical). Therefore, experiments with *LF* and *WLAN* can be initiated, but the results are predictable, as discussed in Chapters 5 and 6.

3.8 Network Parameters

There are multiple ways of achieving communication in MRTs. For example, explicit communication, in which the most popular form of communication is a radio signal (IEEE 802.11X standard), and implicit communication, an example of which could be gesture recognition using computer vision. *MRComm* focuses on explicit wireless communication of the radio signal type. In this section I introduce some of the network parameters used for experiments in this thesis. In Section 3.8.1, I define the two different wireless network types that are used in experiments, in Chapters 5 and 6. In Section 3.8.2, I define network perturbation within the context of this research and the primary perturbation that is present throughout all experiments.

3.8.1 Network Types

The network type refers to the type of connection that links two or more devices (robots) together. A type of connection can have supporting infrastructure, which is generally referred to as Basic Service Set (BSS) or

Extended Service Set (ESS). This essentially means *WLAN* with support-/extended support by Access Point(s) (AP). In such a connection, the AP usually detects, controls and manages the communication between devices. This is simply denoted as the *WLAN* network type and is further expanded upon below. The other type of connection is one with no infrastructure, which is generally referred to as Independent Basic Service Set (IBSS), and is synonymous with peer-to-peer and ad-hoc networking. This connection requires the devices (robots) to self-maintain the network and communication is direct (peer-to-peer) as there is no mediator (i.e. AP). This is denoted as the *AH* network type and is further expanded upon below.

Wireless Local Area Network (WLAN)

WLAN is the standard network type used for analysing communication quality in MRTs when subjected to *Simulated Packet-Loss* (*SPL*, defined in Section 3.8.3). However, as mentioned previously, due to the design of the *SLT* network perturbation, it can also be used with *WLAN*. As previously mentioned, the *WLAN* used is the university campus Wi-Fi, *eduroam* (IEEE 802.11 standard), which was setup to use the 2.4GHz band. The *WLAN* makes use of the three APs positioned throughout the operational (office) environment to mediate the communication between the robots.

As with all forms of communication, wireless communication cannot guarantee the successful transmission and acquisition of messages. Even though ROS employs TCPROS for its standard messaging protocol, when using message queues in a *pub-sub* middleware, there is always the danger that a message will fail to be communicated or will be delivered out of order, as this type of middleware has no higher-level protocol to acknowledge receipt of a message sent from the queue nor does it care about the order of arrival. The aforementioned implies that some limitations and assumptions need to be asserted. Generally, Wi-Fi in the 2.4 GHz band has a wide range of theoretical operating distances, most of which exceed $\approx$ 20 meters, as long as the transmitted signal has direct line-of-sight of the receiver. Moreover, the maximum range that the robots could be from one of the *eduroam* APs at any one moment is less than $\approx$ 15 meters. However, this is not necessarily in direct line-of-sight, making it difficult to know the exact coverage of all possible permutations of routes in the office space. Therefore, for all experiments, including those described in Chapter 4, perfect uniform radial coverage is assumed in the operational environment. To affirm there is complete coverage, a communication test was carried out analysing the signal strength using a laptop as a receiver, while circumnavigating the operational environment. As a result of this test, the above assumption is acceptable. Finally, it is assumed that *signal-to-noise-ratio* (SNR) is static and uniform (i.e. *SPL* network perturbation), and that external SNR from other devices (not the robot team) is negligible.

Ad-Hoc (AH) Network

AH is the second network type introduced here. The *AH* network is a popular method of communication; it is specifically useful in multi-robot teams as it can be used to create a simple peer-to-peer network solution for reestablishing communication in an area where no network infrastructure exists. Furthermore, with the introduction of the *AH* network, signal strength, which is another very important network parameter for communication quality, is added to the possible network perturbations being investigated without breaking any of the assumptions formulated for *WLAN* or *AH*. The characteristics of the *AH* network are: no infrastructure, quick dissemination of information and distributed control (i.e. no single point of failure).

An investigation was conducted in an open (field) outdoor environment between two robots signalling to each other the signal strength at a variable distance. The outdoor environment gives a good indication of how the specific network cards of the robots perform when there are less obstructions to distort communication (Fresnel zone). The analysis from this investigation is shown in Figure 3.15, which is used to establish a default communication limit of 8.0 m for the *AH* network. It should be noted that this communication limit does not strictly imply that communication will **not** occur after 8.0 m, however it is an imposed restriction on the *AH* communication. Furthermore, experiments configured with different network perturbations **do not** affect this default communication limit of the *AH* network. This limit is put in place to make the research problem more tractable and equitable for all experiment configurations using *AH*. Figure 3.15 shows that further than $\approx$ 8.0 m signal strength is on average -70 dBm. The signal strength at this distance signifies a weak signal or that the signal is almost indistinguishable from background noise, which makes predicting distance using signal strength very inaccurate. Similar to *WLAN*, for *AH*, it is assumed that SNR experiences uniform loss and SNR interference from other devices (not the robots) is negligible.

To create an *AH* network, devices connect directly to one another and rely on the close proximity of neighbouring devices to maintain connectivity. Generally, devices can leave (e.g. move out of range) and thereafter automatically rejoin (e.g. move back in range) the network freely without issues. However, shared information is only available as long as a connection between devices is maintained. It is important to note that *MRComm* using the *AH* network type provides a completely distributed MRS.

3.8.2 Network Perturbations

A wireless network could be subject to network issues (perturbations), which most commonly affect message transmission and signal quality. Therefore, network perturbations developed in *MRComm* introduce internal network

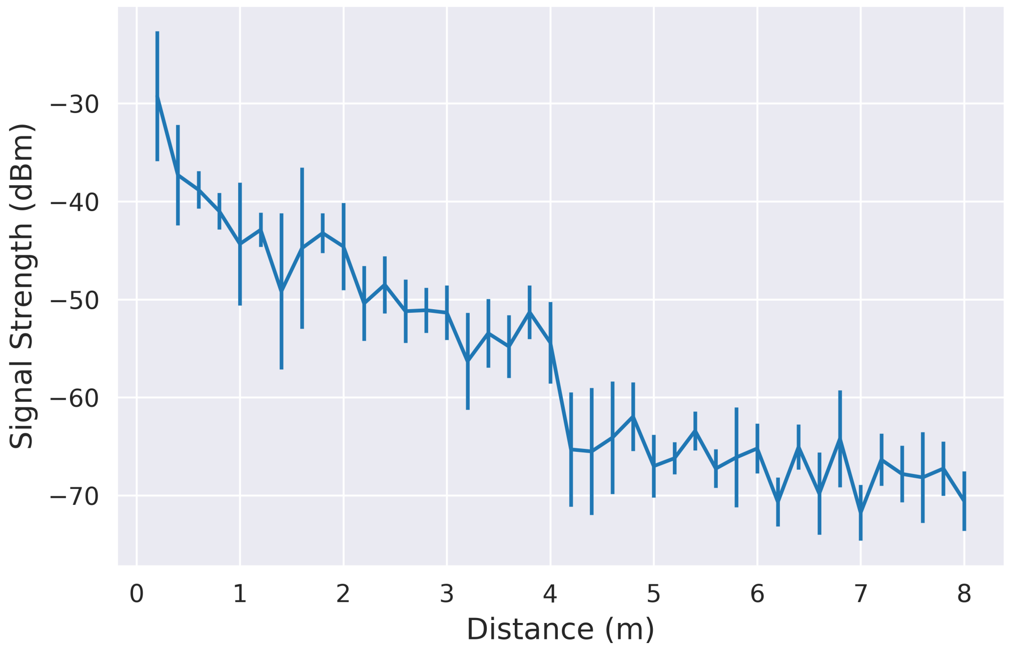


FIGURE 3.15: Average signal strength (30 repeat readings per data point) vs. distance in an outdoor environment.

noise, which affects message transmission and internal monitoring of fluctuation in signal quality during experiments. In networking, these two perturbations could also be referred to as packet-loss and signal strength degradation. Furthermore, they are used to evaluate how a disrupted network affects communication performance during simulated or physical robot experiments. All of the network perturbations except *ESD*, introduced in Chapter 6, are directly transferable between simulated and physical robots. This ensures that the effects are comparable between the different environments. In Section 3.8.3, I introduce the fundamental network perturbation used in all experiments, which is *SPL*.

3.8.3 Simulated Packet-Loss (SPL)

The *SPL* perturbation is a static internal software-engineered packet-loss (i.e. message dropping) method, which is used to mimic degrading communication quality. The methodology used here is inspired by Chowdury [94], who considers *quality of the mechanism* (essentially a term for packet loss), that measures the quality of communication between different populations of agents when communication is interrupted at a percentage of 0%, 25%, 50% and 75%. In this work, it represents the characteristics of an unreliable transmitter or a highly saturated network. *SPL* is a very plausible perturbation to test the ROS messaging middleware (i.e. *pub-sub*), which is used to communicate information between devices. Moreover, *SPL* is great for testing the robustness of the *MRComm* framework, as each agent drops messages received *internally* and directly from the communication pipeline. Therefore, *SPL* allows each agent to internally keep track of how many messages are dropped with no extra computational requirement on the system. There are more computationally complex methods for internal message dropping with no foreseeable benefit. For example, *unbiased* message dropping, which can be achieved by combining shared messages received by topic and processing the messages using the message drop method, or Gaussian noise generation for message dropping, which is not ideal as the normal distribution would yield different results with each experimental run. Moreover, message dropping could be performed externally to the *MRComm* system using the Linux based software, *Network Emulator* (NetEm) [95]. However, the procedure NetEm uses for packet dropping has some similar functionality to the internal *SPL* method used in *MRComm*. Furthermore, the external nature of this software will make it very problematic to record all received shared messages (i.e. specifically, those being dropped), which *MRComm* accomplishes. An external software performing the message dropping would require each agent sending and receiving shared messages to have separate functionality to record all outgoing and incoming traffic to enable the system to record the number of messages being dropped, which is not ideal.

At the start of a mission, all agents operate under unbounded rationality (not affected by *SPL*), and immediately thereafter this assumption weakens by the amount of static packet-loss applied to the agents, i.e. *SPLX*, where $X \in \{0, 25, 50, 75\}$ is the percentage of packet-loss. An example of how SPL functions is demonstrated in Figure 3.16. The function to generate the random probability uses python's *random* [19] package. The *random* package uses pseudo-random generators to provide probability distributions.

Pseudo-randomness is generated using an internal source within the executing device to generate the randomness, which is traditionally a specific clock frequency of the Central Processing Unit (CPU). The *SPL* network perturbation makes use of the pseudo-random uniform distribution similar to that of [96]. The performance of the uniform probability is evaluated with a test which returns the percentage that a number in the range of 1-to-10 is chosen from a total of 10,000 samples using a random seed value of 42 (please see Appendix B.2). As expected, the result is in fact not exactly 10% for each value proving it is pseudo-random. However, the results are considered acceptable for the requirements of the research conducted here.

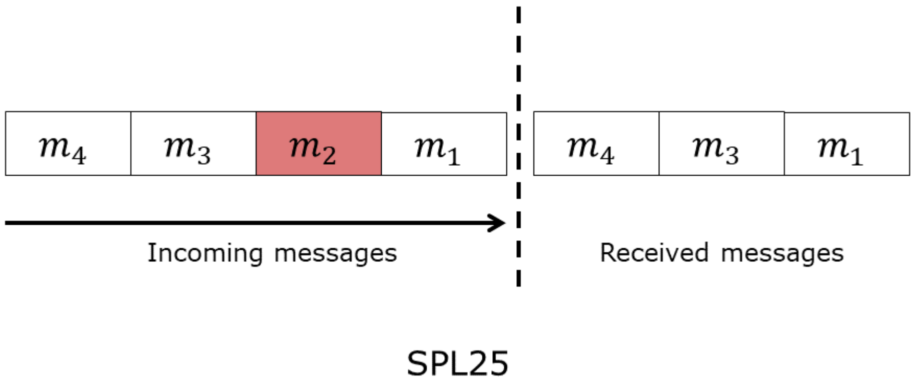


FIGURE 3.16: Abstract representation of *SPL25* not receiving message m_2 after it has been transmitted.

[19] https://docs.python.org/2/library/random.html

3.9 Performance Metrics

Here, I present and reason the choice of the general performance metrics that are recorded per experiment. The *MRTeAm* framework [9] contains thirty performance metrics that are recorded per experiment. Many of the performance metrics are either combinations of different metrics or are specific metrics that examine the MRTA domain problem. This provides a very in-depth analysis of the MRT performance, specifically in the MRTA domain. However, not all the performance metrics designed for the *MRTeAm* framework experiments [9] were used, as it was concluded that some performance metrics did not reveal or provide any new data. For similar reasons, those performance metrics will not be included, and additionally all performance metrics that are focused on the MRTA domain will be removed. The centre of attention will be on the six core performance metrics. The six core performance metrics will be analysed for all experiments, but of these only important or newly designed performance metrics may be represented or discussed on a per chapter basis. The six performance metrics are, namely: **Execution Phase Time**, **Total Movement Time**, **Total Distance Travelled**, **Overall Near Collisions**, **Overall Delay Time** and **Overall Idle Time**. Each of these performance metrics show the mean results of all robots over each experiment configuration, unless otherwise specified.

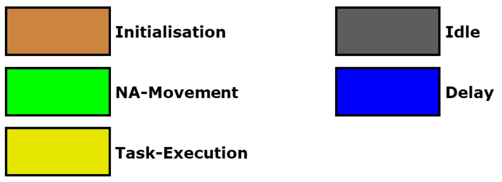
There are two additional metrics that are recorded, but are not individually analysed as they do not contribute to the overall research conducted in this thesis. These are *Initialisation Time* and *Task Execution Time*, they are described below and shown in Figure 3.17.

- *Execution Phase Time:* is recorded from the moment all tasks have been assigned and robots are ready to begin the mission until the final robot has completed the final task of the mission. This includes the time taken for robots to navigate to tasks, the time spent idly waiting for other robots to complete their tasks, obstacle avoidance and any amount of processing and communication delay time taken to complete a task (i.e. send and receive task success/failure messages). This excludes start-up time, task allocation time and any delays in system shutdown after an experiment has concluded. The *Execution Phase Time* is represented by the red vertical dotted lines in Figure 3.17.

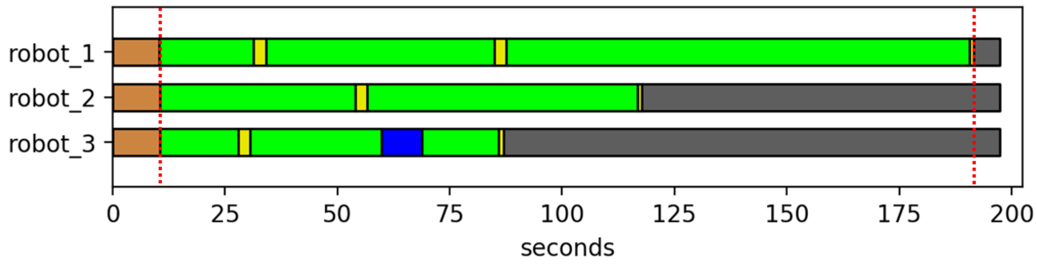
- *Total Movement Time:* is the time the robots spend navigating toward a goal, including re-planning routes when being blocked. It should be noted that a robot being blocked by an object or another robot causing it to re-route, is not the same as a near collision or avoiding a collision. The robots can observe blocked passageways and doorways from a good distance away ≈ 2.0 m and start re-routing immediately. Whereas the robots enter a near collision state when they are less-than ≈ 1.0 m from another robot. Moreover, the total movement time is not strictly indicative of the distance travelled by a robot. It is represented by the light green colour in Figure 3.17.
- *Total Distance Travelled:* is measured in meters and is the total distance that all robots spent travelling toward tasks, avoiding obstacles and collisions, and performing recovery behaviour. The distance is recorded from the odometry of the robot wheels in the physical environment or from the simulated odometry and distance calculations in the simulated environment.
- *Overall Near Collisions:* is the recorded total of near collisions that occurred during a mission. A near collision is generally described as two robots being in the proximity of less than 1.0 m from each other. More accurately, it is when two robots are travelling head-on and enter the "danger zone" or when two robots are in any orientation within the "event horizon" area, as described in Section 3.6.

- *Overall Delay Time:* is the time the robots spend avoiding or waiting for collisions to be concluded during a mission. The *Overall Delay Time* is represented by the dark blue colour in Figure 3.17.
- *Overall Idle Time:* is a specific amount of time, during which robots that have completed their agenda idly wait for the remainder of the team to complete their tasks. Once all tasks are complete all robots, including any that are idly waiting, transition to mission end. *Overall Idle Time* is represented by the dark grey colour in Figure 3.17.
- *Initialisation (Assigner) Time:* metric is defined as the time from when the *task assigner* has finished assigning all tasks to the robot team and sends an experiment start message. Thereafter, a robot behaviour is initialised and the robots begin to move. This is represented by the light brown/dark orange at the left hand side of the timeline in Figure 3.17.
- *Task Execution Time:* metric is the small amount of processing and messaging time it takes for a robot to communicate that a task has been reached and completed. This is represented by the yellow colour in Figure 3.17.

The two metrics *Overall Near Collisions* and *Overall Delay Time* are analysed, however they may not necessarily be included in all results as they may represent negligible difference across some experiment configurations.



(a) Timeline legend



(b) A timeline presenting time-based performance metrics and how they are calculated during an experiment (example configuration F_1 from Table 4.6). The vertical red dotted line represents *Execution Phase Time*.

FIGURE 3.17: Timeline of an experiment using the core performance metrics.

3.10 Experiment Design

Experiment design is important to plan for the specific purpose of the research problem being investigated. As such, in this thesis the network parameters, described in Section 3.8, and accompanying agent behaviours, described in Section 3.7, are more important than the task and team composition parameters. Each experiment is formally defined using an experiment schema, which is the organisation and structure of a particular experiment. An experiment schema is made up of network parameters and a scenario, which are described below.

I initially define a *scenario*, denoted S_i, where i identifies a specific scenario configuration, which extends the task composition, to include additional parameters. A general *scenario* is defined in equation 3.3.

$$S_i = T_{COMP} \times S_{COMP} \times M \times N \times T \tag{3.3}$$

where T_{COMP} is comprised of the task composition $\langle ST, SR, IT, EV, SA \rangle$, as described in Section 2.3.3 and a set of task locations on the map $\{(x_1, y_1)...(x_T, y_T)\}$, S_{COMP} is the starting composition, which contains the starting strategy, i.e. $\{clustered\}$, and the robot starting locations on the map $\{(x_1, y_1)...(x_N, y_N)\}$, M is the map label that identifies a specific map (operational environment), N is the size of the robot team, and T is the number of tasks in a mission. There are two parameters in the starting composition (S_{COMP}) that remain fixed across experiments in this thesis, which are the starting strategy, which is always $\{clustered\}$, and map M, which is always $strand\_office$ (Figure 3.5). The choice of keeping most of T_{COMP} and the two aforementioned S_{COMP} parameters fixed was to make experiment results more consistent and comparable. Specifically, to focus on changing the network parameters and analysing communication performance for the MRCP. However, due to unforeseen circumstances, such as lab space changes and department overhaul, the operational area of the $strand\_office$ map was limited. Therefore, the task locations (T_{COMP}) and starting locations (S_{COMP}) parameters had to be altered, as noted in Chapter 4. Furthermore, previous works [9, 45–47] have defined the starting strategies parameter as $\{clustered, distributed\}$, which is used by experiments designed for *MRTeAm*. A clustered robot starting strategy is where robots start a mission adjacent and in close proximity to each other, i.e. within the same room/space. A distributed robot starting strategy is where robots start a mission distant from each other, i.e. not in the same room/space.

The network parameters are split into network type, which can either be *WLAN* or *AH*, and network perturbation, which is *SPL*. The network perturbation parameter is expanded upon in each chapter, therefore the selection increases. Finally, the behaviour parameter, which can either be *NB* or *LF*, is introduced in experiments from Chapter 5 onwards. The behaviour parameter is linked to the network parameters, because the robot behaviour changes depending on signal strength, which is affected by the network type.

The general experiment schema equation 3.10 is used in Chapter 4. However, it is revised from Chapter 5 and thereafter, because of the introduction of the *AH* network and expanded network perturbations.

$$\begin{aligned} F_y = \{S_i\} \times \{NET\_TYPE\} \times \{NET\_PERTURBATION\} \\ \times \{ROBOT\_BEHAVIOUR\} \end{aligned}$$

where S_i is the scenario, NET_TYPE is the network type, i.e. *WLAN* or *AH*, $NET\_PERTURBATION$ is the *SPL* network perturbation and $ROBOT\_BEHAVIOUR$ is the specified MRT behaviour, i.e. *NB* or *LF*.

3.10.1 Experiment Sample Sizes

It is important to determine the pros and cons of different sample sizes in conjunction with the method that will be used for statistical analysis. There are obvious advantages and less obvious disadvantages to having a large sample size of data. Table 3.1 covers the major advantages and disadvantages of large and small sample sizes.

The general advantages of large sample sizes, hundreds, thousands or greater sample sizes, are that the variability (standard deviation) of data is very small, i.e. more reliable, which helps reduce the biasing in data and overall more powerful statistical analysis are possible. For example, variation caused by the environment, which could be people moving around in the office space or ambiguous sensor or network noise, that affects the robot performance during experiments. Biasing of this kind can be revealed by leveraging different statistical techniques on large sample sizes of data. Identifying outlier data points can further reveal software issues in the architecture or design of a system or hardware issues that are otherwise hard to detect. Generally the reverse is true for small sample sizes, for example variability is higher, i.e. less reliable, bias is increased (this is experiment/-data dependant) and outliers are weak indicators of potential issues.

The disadvantages of large samples sizes of data mirror the advantages of small sample sizes. In particular, there is one considerable advantage of small sample size data (disadvantage of large sample sizes), which had an impact on the sample size decision of the experimental work presented in this thesis and it is "Quick Turnover" time. Turnover time is the time taken to collect, process and analyse experimental results. Each part of the turnover process, collection, processing and analysis, takes time. The issue arises when the robotic frameworks (*MRTeAm* and *MRComm*) were being developed, which led to frequent changes in the code base. During the baseline experiments with *MRTeAm*, less code base changes were required and experiments could be executed more regularly. However, during *MRComm*'s development hundreds of experiments were constantly run with a mixture of baseline and extended features. Executing a mixture of baseline and new feature experiments (e.g. adding AH, SLT, etc.) ensured that a new feature would not impede the functionality of any baseline feature. The results from these experiments were analysed but unusable. However, these results were used to guarantee that the consistency between the frameworks was identical. For example, running the baseline experiments from Tables 4.1 and 4.2 in Chapter 4 (without incorporating *FKIE*, *AON* or robot behaviours) would yield similar results regardless of whether *MRTeAm* or *MRComm* are used. There is the argument, that large sample size data can have powerful statistically significant results at the cost of more complex analysis methods, whereas small sample sizes are not as powerful statistically, but simpler analysis methods exist. Therefore, due to the ease of statistical analysis, manageability and quick turnover time for the development of the experimental *MRComm* framework, collecting small sample size data was preferred over large sample size data. The particular statistical method

	Large Sample Size	**Small Sample Size**
Advantages	Reduced Variability	Quick Turnover
	Reduced Bias	Simple Analysis
	Identify Outliers	
Disadvantages	Slow Turnover	Increased Variability
	Complex Analysis	Increased Bias
		Weak Outliers

TABLE 3.1: The advantages and disadvantages of large and small sample sizes.

used for analysis is discussed in later chapters.

3.11 Summary

In this chapter, I introduced the *MRComm* framework, the general parameter design and overview. Furthermore, the general experiment design was introduced, which is used to run both simulated and physical robot experiments in this thesis. Although, as mentioned, each chapter extends some parameters and uniquely modifies the general experiment design.

Chapter 4

Baseline Experiments on Communication Quality

4.1 Introduction

Providing correct information to robot team members and having up-to-date local knowledge are two of the critical functions that depend on networked communication facilities. An unreliable network connection can mean that messages get dropped and robots lose their ability to receive commands, transmit sensor data and generally interact with the environment and robot teammates. Furthermore, most research in MRS assumes 100% network connectivity, 100% of the time; but this is unrealistic for real-world domains. This chapter presents two sets of experiments. The first is conducted on a standard MRS framework, *MRTeAm* by Schneider [9], which is used to represent the deteriorating and inconsistent communication performance experienced by a robot team when connectivity is compromised at increasing levels. The set of experiments conducted using the *MRTeAm* framework, presented in Section 4.2, are a baseline to understand how poor communication can impact a multi-robot team. The second set of experiments are conducted on a MRS framework, *MRComm*, which has been built with the purpose of improving and providing consistent communication performance for all the analysed metrics of a robot team. The *MRComm* framework, in Section 4.3, uses the ROS *FKIE* package for communication and shows improved consistency of the performance metrics and additionally introduces two performance metrics, which are specific to communication performance.

A simple network perturbation is employed to disrupt multi-robot communication, *SPL*. *SPL* is the baseline network perturbation and it is described in Section 3.8.3. It is a probabilistic message-loss function, which is integrated in *MRComm* and applied to the two shared message topics, *pose messages* and *task status messages*, of the robot team. The probabilistic message-loss function will cause shared messages to "fail" to be delivered at a set percentage rate (i.e. 0%, 25%, 50% or 75%) at the start of an experiment.

The rest of this chapter is made up of Section 4.2, which contains the experimental setup 4.2.1, results 4.2.3 and discussion 4.2.4 sections of the *MRTeAm* set of experiments. Section 4.3 contains the experimental setup 4.3.1, extended performance metrics 4.3.2, results 4.3.3 and discussion 4.3.4 sections of the *MRComm* set of experiments. In Section 4.4, the results of both frameworks are analysed and discussed. Section 4.5 concludes the chapter.

4.2 MRTeAm Baseline Empirical Results

The experiments conducted with the *MRTeAm* framework are a stepping stone used to analyse the effect that degrading communication has on MRT performance. Moreover, the experiments are also a benchmark highlighting the components of the system that require attention and possible adaptation to improve communication.

Two different scenarios are used for the experiments, described in Section 4.2.1, to conduct simulated and physical experiments. At the time of designing the initial (simulated robots) scenario, the lab where the robots were kept was in fact an office, which was also the initial starting location of the robots(Figure 4.1). Furthermore, many of the offices used for task locations were either not occupied or were thought to be accessible. However, soon after completing the simulated robot experiments, the lab was moved and many of the unoccupied office spaces were no longer vacant. Therefore, the scenario was re-designed for the physical robot experiments, illustrated in Figure 4.2. The two scenarios are closely related to give comparable results. However, the resulting performance metrics are marginally different. Although results may differ, the trends are similar, which allows for some association of analyses.

4.2.1 Experiment Configuration

To signify that a scenario belongs to the experiments conducted with *MRTeAm*, a leading zero is added to the identifying scenario number, i.e. S_{0X}. The simulated robots scenario is denoted S_{01} and is illustrated in Figure 4.1. The physical robots scenario is denoted S_{02} and is illustrated in Figure 4.2. Scenarios S_{01} and S_{02} use identical parameters, apart from minor changes in starting locations and task locations, as shown in Figures 4.1 and 4.2. The common setup for the scenarios is as follows:

- Tasks' composition is $\langle ST, SR, IT, EV, SA \rangle$ and task locations are represented by crosses, in Figure 4.1 for the simulated and Figure 4.2 for the physical experiments;
- Starting composition (S_{COMP}) is *clustered* and starting robot locations are represented by circles, Figure 4.1 for the simulated and Figure 4.2 for the physical experiments;
- The map $M = strand\_office$;
- The number of robots $N = 2$;

- The number of tasks $T = 6$.

The RR algorithm 1 is used to assign tasks, the task agenda of each robot is as follows; ROBOT_1 (r_1) is assigned tasks $t_1 \in 1, 3, 5$ and ROBOT_2 (r_2) is assigned tasks $t_2 \in 2, 4, 6$.

The *MRTeAm* framework is used as a benchmark for these experiments to analyse communication quality, therefore a single network parameter was initially identified and defined at the time, which was the network perturbation *SPL*. However, the expansion of the network parameters in later experiments (i.e. Chapters 5 and 6) helped to identify another baseline network parameter. For instance, the network type used in *MRTeAm* experiments is *WLAN* and the *SPL* network perturbation is functionally identical to the one integrated in the *MRComm* framework. The complete experiment configurations for this section are shown in tables 4.1 and 4.2. The simulated experiments in Table 4.1 are each repeated 30 times and the physical experiments in Table 4.2 are each repeated 10 times, the decision for the sample size and the statistical analysis is discussed in Section 4.2.2.

Experiment schema
$F_{01} = \{S_{01}\} \times \{WLAN\} \times \{SPL0\} \times \{NB\}$
$F_{02} = \{S_{01}\} \times \{WLAN\} \times \{SPL25\} \times \{NB\}$
$F_{03} = \{S_{01}\} \times \{WLAN\} \times \{SPL50\} \times \{NB\}$
$F_{04} = \{S_{01}\} \times \{WLAN\} \times \{SPL75\} \times \{NB\}$

TABLE 4.1: Experiment configurations for the simulated robots using *MRTeAm* and the S_{01} scenario.

Experiment schema
$F_{05} = \{S_{02}\} \times \{WLAN\} \times \{SPL0\} \times \{NB\}$
$F_{06} = \{S_{02}\} \times \{WLAN\} \times \{SPL25\} \times \{NB\}$
$F_{07} = \{S_{02}\} \times \{WLAN\} \times \{SPL50\} \times \{NB\}$
$F_{08} = \{S_{02}\} \times \{WLAN\} \times \{SPL75\} \times \{NB\}$

TABLE 4.2: Experiment configurations for the physical robots using *MRTeAm* and the S_{02} scenario.

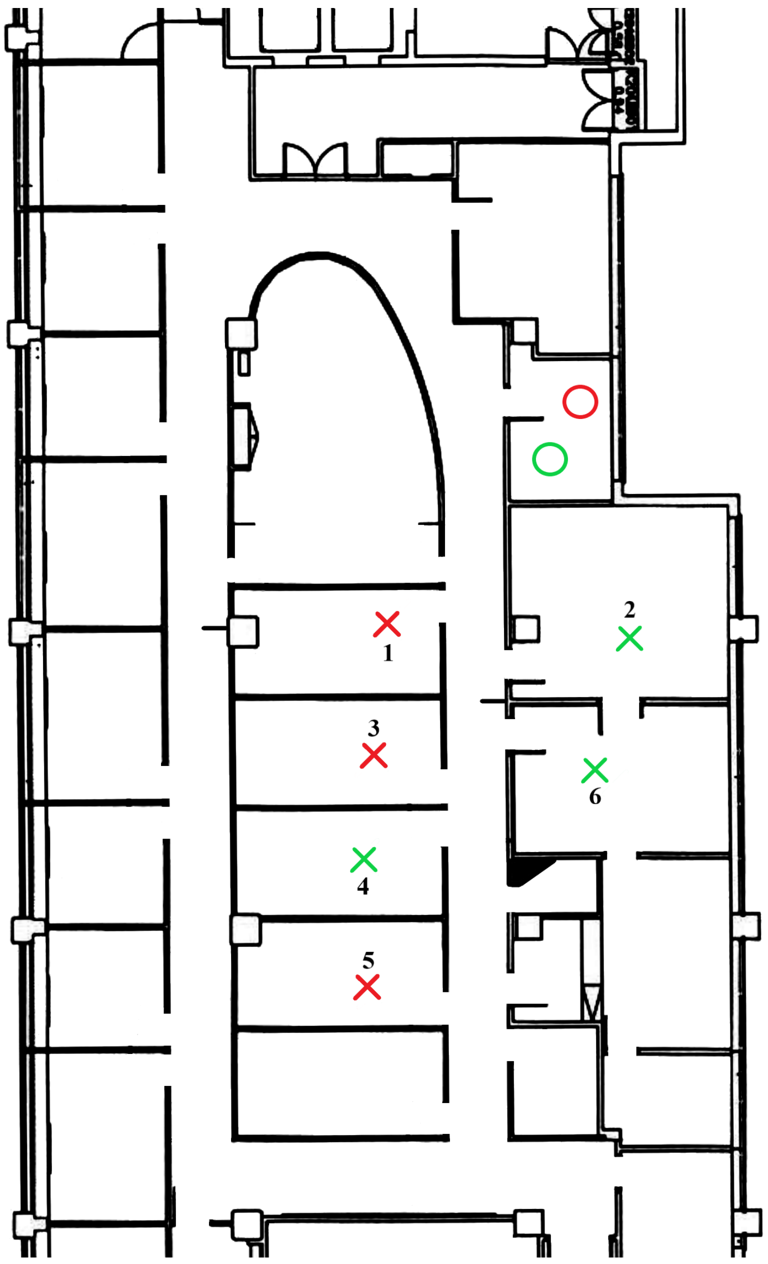


FIGURE 4.1: Office setting for scenario S_{01} (simulated robots). The different coloured circles represent ROBOT_1 (the red circle) and ROBOT_2 (green circle), and the corresponding crosses represent task locations that will be assigned to those robots.

4.2.2 Experiment Statistics

The statistical analysis to be used depends on whether the data are normally distributed or not, and the sample size. The Shapiro-Wilk [97](SW) test is used to check if a data sample is likely to come from a normal distribution,

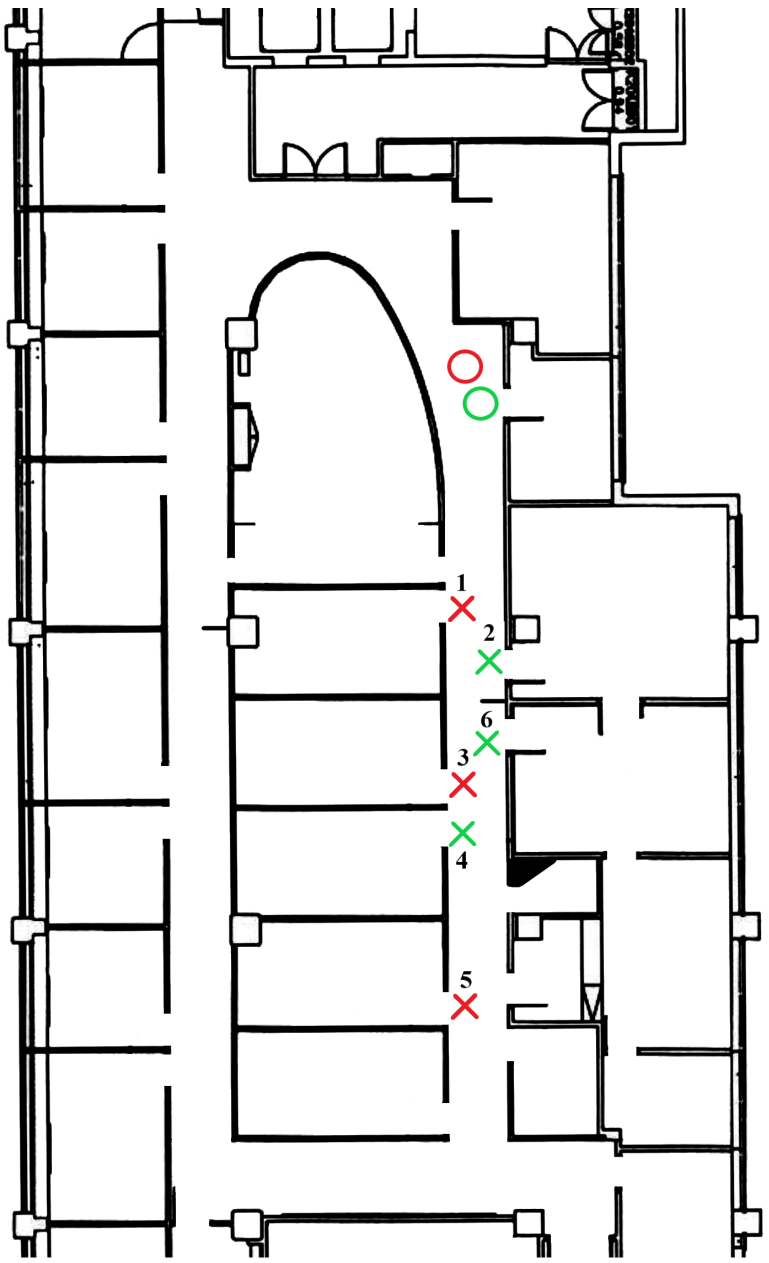


FIGURE 4.2: Office setting for scenario S_{02} (physical robots). The different coloured circles represent ROBOT_1 (the red circle) and ROBOT_2 (green circle), and the corresponding crosses represent task locations that will be assigned to those robots.

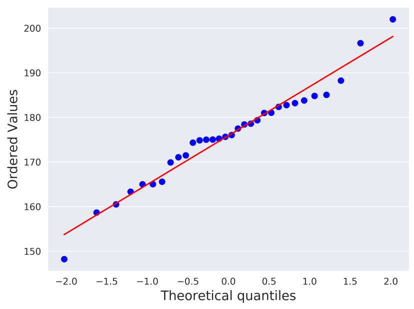
and it is known to work well on small sample sizes of $n \leq 50$, where n indicates the number of data points (observations). However, for small sample sizes, specifically for the physical experiments (n=10), it becomes increasingly difficult to determine the distribution of data samples as the SW test becomes weaker. This raises a number of issues. For example, not all data samples are normally distributed and the ideal network quality (SPL0) is

Experiment Label	Performance Metric	W-value	p-value
WLAN_NB_SPL0	EXECUTION_PHASE_TIME	0.971	0.627
WLAN_NB_SPL25	EXECUTION_PHASE_TIME	0.979	0.564
WLAN_NB_SPL50	EXECUTION_PHASE_TIME	0.983	0.926
WLAN_NB_SPL75	EXECUTION_PHASE_TIME	0.956	0.234

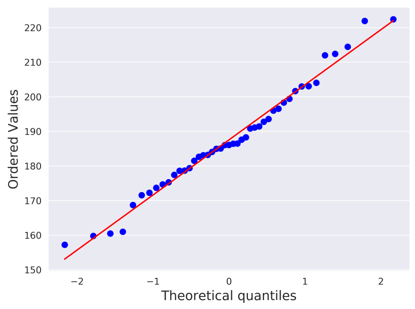
TABLE 4.3: The Shapiro-Wilk value and p-value for the *Execution Phase Time* performance metric.

being compared against increasing network perturbations (SPL25, SPL50, SPL75), which implies that for each performance metric, each experiment configuration data sample needs to be normally distributed. To demonstrate these issues, when performing normality tests for the experiments in Table 4.1, only a single performance metric (*Execution Phase Time*) showed normality across all experiment configurations, Table 4.3. The following conditions need to be satisfied for the null hypothesis (i.e. the data samples come from a normal distribution) to be accepted; for data sample size n=30 and α=0.05 (assuming 95% confidence interval, CI) the W-value $\geq$ 0.927 (Table 6 [97]) and p-value > 0.05. Additionally, quantile-quantile (Q-Q) plots, Figure 4.3, are used as supporting evidence of normal distribution, which is illustrated by the sample data points closely following the line of best fit. However, although the W and p values in Table 4.3 indicate that all the samples are normally distributed, the cooresponding Q-Q plot for schema F_{04}, Figure 4.3d, can be interpreted as either, conforming or not conforming to a normal distribution (which is reliant on the researchers knowledge of the data and statistical requirements). Therefore, using parametric analysis will be less effective, as some data samples may not conform to a normal distribution or the sample size may be too small to infer normality, as such non-parametric analysis is preferred.

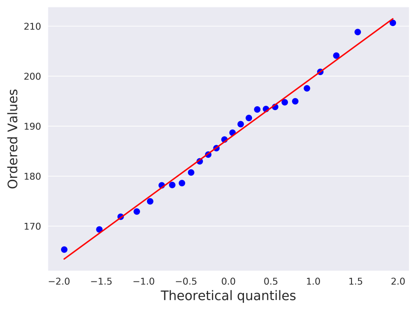
The Mann-Whitney [98] non-parametric test can be used to investigate multiple hypotheses about two independent samples, such as whether the samples come from the same distribution or whether observations in one sample are inclined to be larger or less than in the other. However, there are



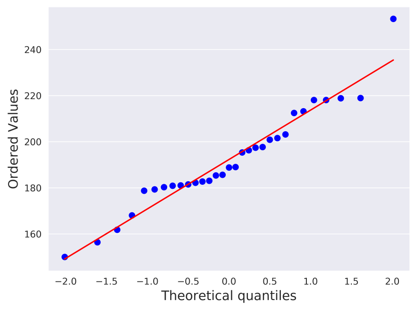
(A) Q-Q plot using schema F_{01} (SPL0).



(B) Q-Q plot using schema F_{02} (SPL25).



(C) Q-Q plot using schema F_{03} (SPL50).



(D) Q-Q plot using schema F_{04} (SPL75).

FIGURE 4.3: Q-Q plots of *Execution Phase Time* and experiment configurations from Table 4.3.

a few assumptions that must be made for the Mann-Whitney U-test statistic to be valid. The assumptions are, all data points (from each sample) are independent of each other and all pairwise calculations are ordinal. The null hypothesis ($\mathbf{H_0}$) being investigated for the Mann-Whitney U-test is whether the first sample of observations are statistically significantly equal to the observations of the second sample. For instance, the probability that a randomly drawn data point from one group is equal to a randomly drawn data point from another. Moreover, the alpha (α) value chosen for the null hypothesis is 0.05 (95% CI). The α value is the probability of rejecting the null hypothesis when it is actually true, i.e. $\alpha \geq 0.05$. The alternative hypothesis ($\mathbf{H_A}$) being investigated is whether the observations from the first data sample are more likely to be less than the observations from the second sample, i.e. $\alpha < 0.05$. The probability (p) value is the chance of accepting either

$\mathbf{H_0}$ or $\mathbf{H_A}$ by observing if $\alpha \geq 0.05$ or $\alpha < 0.05$ respectively. The p-value is one of the major results that is scrutinised for the statistical significance analysis, which is dependent on the U-value. The U-value is a score that is made up of numeric ranks that are assigned to all observations of the two independent samples. An observation from one sample is compared to an observation from another and this is assigned as the rank score. The way pairwise ranks are scored depends on the hypothesis that is being tested, i.e. smaller, larger or difference in population distribution. Rank ties are solved by assigning a rank that is equal to the midpoint of the tied rank. The U-value equation [98] is:

$$U = mn + \frac{m(m+1)}{2} - T \tag{4.1}$$

Where m is the first independent sample size and n is the second independent sample size and T is $\Sigma Rank(x_i)$, where x_i is the pairwise rank value. The U-value is dependent on the sample size and rank-sums between sample points. A large U-value indicates more likely the $\mathbf{H_0}$ hypothesis, whereas a smaller U-value indicates the $\mathbf{H_A}$ hypothesis. The *effect size* will also be analysed. It is used to measure the magnitude of the experiment effect. The larger the effect size the more reliable the accepted hypothesis. The effect size can also be used to help with power analysis and sample size planning. The effect size equation [99] is:

$$r = \frac{Z}{\sqrt{N}} \tag{4.2}$$

$Z = \frac{U - m_U}{\sigma_U}$, where U = equation 4.1, $m_U = \frac{mn}{2}$ and $\sigma_U = \sqrt{\frac{mn(m+n+1)}{12}}$, and N is the sample size being analysed. The book by Cohen [99] interprets statistical power, effective size and sample size. Cohen [99] asserts that an effect size of [0.1-0.3] is considered weak, [0.3-0.5] is considered medium and >0.5

is considered strong. A negative effect size is interpreted in reverse, i.e. <-0.5 is considered strong, and generally indicates that the observations from the initial data sample are less than the observations from the compared data sample. It should be noted that the interpretation of effect size given by Cohen [99] is not be considered universal, but should be interpreted situationally. Finally, for smaller sample sizes, i.e. for the physical experiment results in the thesis, the p-value loses some of its significance. Therefore, the U-value, p-value and effect size will be scrutinized together to better interpret the statistical result.

4.2.3 Results

This section contains the statistical analysis of the *MRTeAm* experimental results and presents plots showing how the four main performance metrics, *Execution Phase Time*, *Total Movement Time*, *Total Distance Travelled* and *Overall Near Collisions*, vary with increasing packet-loss (*SPL*). Each performance metric is represented by a figure, which contains two sub-figures A and B, which are the simulation and physical experiment results, respectively. The aim is to show that there are statistically significant differences in some of the performance metrics with increasing levels of *SPL*. The resulting data samples' observations are analysed using the Mann-Whitney U test [100]. The *Overall Near Collisions* performance metric(Figure 4.7) results are briefly discussed in Section 4.2.4 as the data does not provide much insight and is irregular for scenarios S_{01} and S_{02}.

Execution Phase Time, *Total Movement Time* and *Total Distance Travelled* are verified for statistical significances between data samples. Table 4.4 shows the simulation experiments' U and p values, effect size and the accepted hypotheses, for all but the *Overall Near Collisions* performance metric, when there is no message loss (i.e. *SPL0*), against increasing levels of message loss (i.e. $\geq$ *SPL25*). The *Execution Phase Time* and *Total Movement Time* metric results in Table 4.4 show similar increasing trends and $\mathbf{H_A}$ is accepted for *SPL25* and *SPL50* respectively. Moreover, the effect size is considered either moderate or strong and the plots in Figure 4.4A and Figure 4.5A support this outcome visually. However, *Total Distance Travelled* performance metric shows no statistical change with increasing *SPL*. This can be attributed to the nature of simulation experiments in general and the simplicity of scenario S_{01} , as there is better overall communication and sensor information, enabling the robots to travel the exact same route every time (i.e. minor change in distance).

Table 4.5 shows the *Execution Phase Time*, *Total Movement Time* and *Total Distance Travelled* metrics for the physical robots comparing the ideal case, when there is no message loss (i.e. *SPL0*), against increasing levels of message loss. The results are different to the simulated experiments. The physical experiments' U-values are much lower than those of the simulated experiments (i.e. sample size differences) and are therefore more sensitive to the hypotheses being tested and may be inaccurate. *Execution Phase Time* follows a similar increasing trend to the equivalent simulated experiments metric in Table 4.4 with a greatly increased standard deviation, but the effect size is weak and $\mathbf{H_A}$ is not accepted. A similar but much weaker increasing trend is noticeable for the *Total Movement Time* metric. However, the progressive decrease of the U and p values for both the aforementioned metrics indicate two possibilities. The first, is that the sample size is not large enough to consolidate statistical significance, further supported by the weak effect size, and the second is that increasing *SPL* is negatively affecting the two metric, however not by a significant enough margin. A larger sample size is required to determine statistical significance and the proof-of-concept sample of 10 physical experiment runs is not enough. The effect size is used to consider if the sample size and the chosen hypothesis hold enough statistical power. As calculating the sample size will be different and contradictory for every different performance metric. For example, there are four performance metrics specifically being investigated here and the controlled variable (*SPL*) has four instances, this means for every independent sample size comparison to the ideal case, there will be 16 different calculations for the estimated preferred sample size.

$\mathbf{H_A}$ is accepted for *SPL25* for the *Total Distance Travelled* metric, revealing it is significantly affected by increased message dropping. More noise and message transmission errors exist in the physical environment, therefore path planning is not as reliable and it is more likely that robots do not take identical paths during each experiment, resulting in increased *Total Distance Travelled*. Moreover, the *Total Distance Travelled* result indicates that network design/parameters can greatly affect the performance of robots between identical experiments executed in simulation and physical environments. It indicates that network design needs to be considered more carefully to create a more balanced playing field between the simulated and physical environments. The physical results can be visualised in Figures 4.4B, 4.5B and 4.6B.

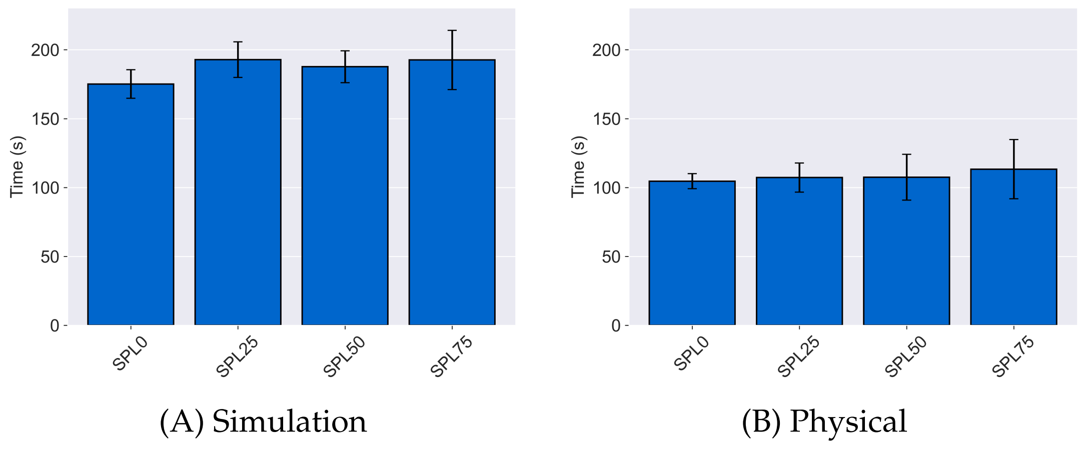


FIGURE 4.4: MRTeAm results for *Execution Phase Time*, with increasing *SPLX*

SPL0 versus Perturbation	**U**	**p**	**effect size**	**Accepted Hypothesis (H_0/H_A)**
Execution Phase Time				
SPL25	108	0.00	-0.92	$\mathbf{H_A}$
SPL50	192	0.00	-0.70	$\mathbf{H_A}$
SPL75	192	0.00	-0.70	$\mathbf{H_A}$
Total Movement Time				
SPL25	471	0.63	0.06	$\mathbf{H_0}$
SPL50	305	0.02	-0.39	$\mathbf{H_A}$
SPL75	200	0.00	-0.67	$\mathbf{H_A}$
Total Distance Travelled				
SPL25	418	0.32	-0.09	$\mathbf{H_0}$
SPL50	487	0.71	0.10	$\mathbf{H_0}$
SPL75	585	0.98	0.36	$\mathbf{H_0}$

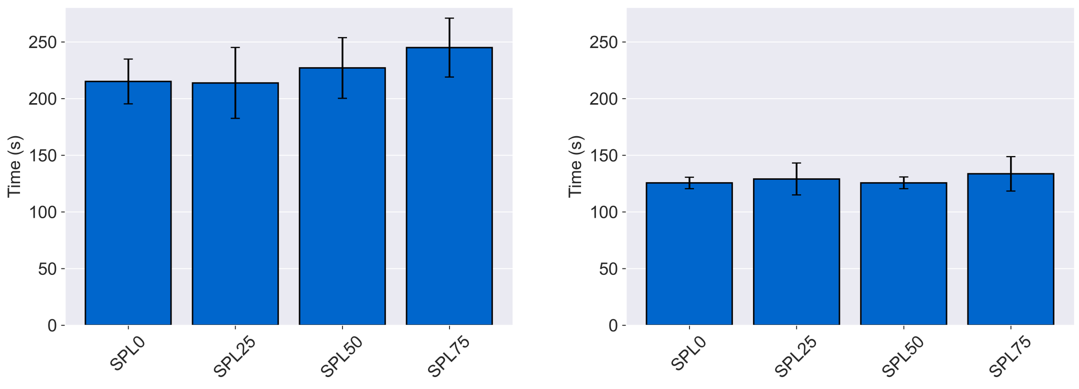
TABLE 4.4: MRTeAm S_{01} simulated robot experiments, showing U-value, p-value, effect size and hypothesis outcome for the ideal case (i.e. *SPL0*) versus all other values (*SPLX*).

SPL0 versus Perturbation	**T**	**p**	**effect size**	**Accepted Hypothesis (H_0/H_A)**
Execution Phase Time				
SPL25	48	0.45	-0.05	$\mathbf{H_0}$
SPL50	57	0.71	0.17	$\mathbf{H_0}$
SPL75	36	0.15	-0.33	$\mathbf{H_0}$
Total Movement Time				
SPL25	100	1.0	0.0	$\mathbf{H_0}$
SPL50	57	0.71	0.17	$\mathbf{H_0}$
SPL75	44	0.34	-0.14	$\mathbf{H_0}$
Total Distance Travelled				
SPL25	10	0.00	-0.96	$\mathbf{H_A}$
SPL50	28	0.05	-0.53	$\mathbf{H_0}$
SPL75	14	0.00	-0.86	$\mathbf{H_A}$

TABLE 4.5: MRTeAm S_{02} physical robot experiments, showing U-value, p-value, effect size and hypothesis outcome for the ideal case (i.e. *SPL0*) versus all other values (*SPLX*).

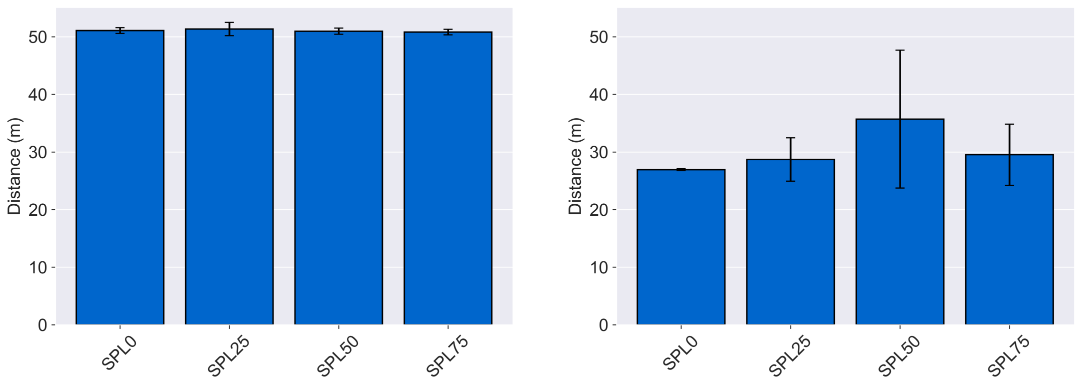
4.2.4 Discussion

The experiment scenarios S_{01} and S_{02} are marginally different and it is clear from the more difficult to navigate tasks in S_{01} that the simulation experiment results take longer time to complete and travel further to reach their



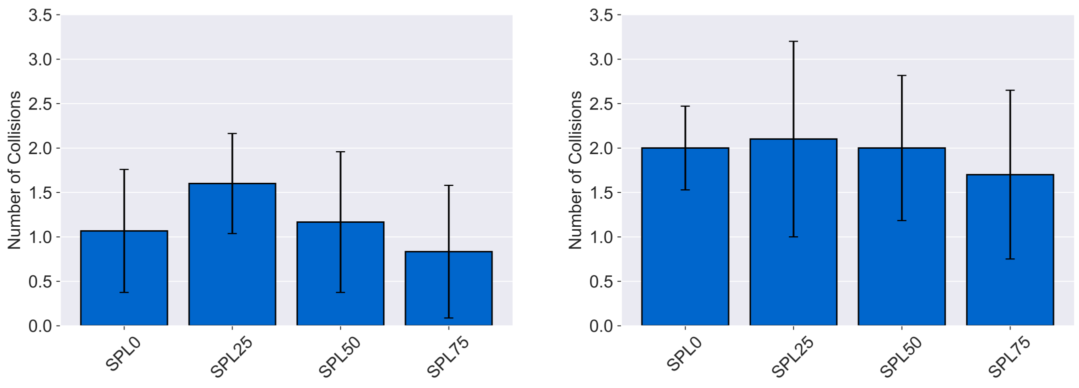
(A) Simulation (B) Physical

FIGURE 4.5: MRTeAm results for *Total Movement Time*, with increasing *SPLX*



(A) Simulation (B) Physical

FIGURE 4.6: MRTeAm results for *Total Distance Travelled*, with increasing *SPLX*



(A) Simulation (B) Physical

FIGURE 4.7: MRTeAm results for *Overall Near Collisions*, with increasing *SPLX*

goals, as seen in Figures 4.4, 4.5 and 4.6. Therefore, the statistical results cannot be directly compared. However, different scenarios' performance metric trends can be compared and statistical analysis can be used to further stimulate the relationship. For instance, *Execution Phase Time* and *Total Movement Time* performance metrics exhibit similar increasing trends, both visually and statistically. Scrutinising the *Total Movement Time* performance metric for the physical experiments more closely, it is noted that $\mathbf{H_A}$ is never accepted for increasing *SPL*, but it is clear that both the U and p values are affected by *SPL*. However, it cannot be said that the *Total Movement Time* result is statistically significant, but it can be visually and statistically shown that it is affected and that the increasing trend is similar for both the simulated and physical experiment scenarios. The *Total Distance Travelled* performance metric displays very different trends between simulated and real robot experiments. This can be attributed to the simulated robot experiments experiencing less sensor noise and other external perturbations, which results in smoother path planning and better avoidance of other robots, as such it is more likely for robots to travel the same route every time (i.e. the same distance). The difference between simulated and physical robots in *Total Distance Travelled* is important. It highlights the divide between network based robot experiments in simulation and physical environments. Finally, it is observed that the standard deviation per metric for the simulated experiments does not generally increase as *SPL* is increased, however it does for the physical experiments. This behaviour is expected, since the simulated environment has predictable and static noise, unlike the physical environment.

The *Overall Near Collisions* metric reveals that on average more collisions may occur during a physical experiment compared to a simulated one, and that the variability is overall quite high. The most interesting aspect of this metric revealed by Figure 4.7, is that collision occurrence marginally increases for *SPL25*, but then gradually decreases as *SPL* increases. No concrete analysis can be made unless a much larger data-set is gathered and more scenarios are designed with collision analysis in mind. Although an assumption for this phenomenon is that a robot cannot register a collision as the messages warning of a collision are being dropped at a faster rate with increasing *SPL*, therefore a collision is not recorded.

4.3 MRComm Empirical Results

The following experiment results are taken from a broader set of experiments, so that they can match the set of experiment configurations in Section 4.2. It is important to note that the two scenarios defined in Section 4.2.1 are different to the single scenario defined in Section 4.3.1. The initially chosen robot start locations and task locations in the two scenarios of Section 4.2.1 were either no longer available or the locations were inconvenient. When physical experiments were conducted in Section 4.2.1, the initial locations used for the task composition were either disturbing the normal office work of colleagues or experiments were being disrupted. Moreover, as mentioned previously, the changes between the simulated and physical environments' task locations were unforeseen and caused differences in the experiment results.

4.3.1 Experiment Configuration

To signify that this is a scenario for experiments using *MRComm* and for consistency, as the same scenario is used in Chapters 5 and 6 as well, there will be no initial zero in front of the scenario number (i.e. S_X, where X denotes the identifying scenario number). For all experiments conducted on *MRComm* scenario S_1 is used. Scenario S_1, shown in Figure 4.8, has the following setup:

- Tasks' composition $\langle ST, SR, IT, EV, SA \rangle$ and task locations are represented by crosses in Figure 4.8;
- Starting composition (S_{COMP}) is *clustered* and starting robot locations are represented by circles in Figure 4.8;
- The map $M = strand\_office$;
- The number of robots $N = 3$;
- The number of tasks $T = 7$.

As before, the same RR algorithm 1 is used to assign tasks. The task agenda of each robot is as follows: ROBOT_1 (r_1) is assigned tasks $t_1 \in 1, 4, 7$, ROBOT_2 (r_2) is assigned tasks $t_2 \in 2, 5$ and ROBOT_3 (r_3) is assigned tasks $t_3 \in 3, 6$. The scenario S_1 remains the same for both simulated and physical robot experiments, and the task composition and starting composition do not change.

The *MRTeAm* experiments helped identify the baseline network parameters for analysing communication quality, which are the *WLAN* network type and *SPL* network perturbation. The same baseline network parameters are integrated and used by *MRComm* to conduct the same experiment configurations, but with an updated scenario. The baseline experiment configurations for *MRComm* are shown in Table 4.6. The simulated experiments are each repeated 30 times and the physical experiments are each repeated 5 times.

Experiment schema
$F_1 = \{S_1\} \times \{WLAN\} \times \{SPL0\} \times \{NB\}$
$F_2 = \{S_1\} \times \{WLAN\} \times \{SPL25\} \times \{NB\}$
$F_3 = \{S_1\} \times \{WLAN\} \times \{SPL50\} \times \{NB\}$
$F_4 = \{S_1\} \times \{WLAN\} \times \{SPL75\} \times \{NB\}$

TABLE 4.6: Experiment configurations for both simulated and physical robots using MRComm and the S_1 scenario.

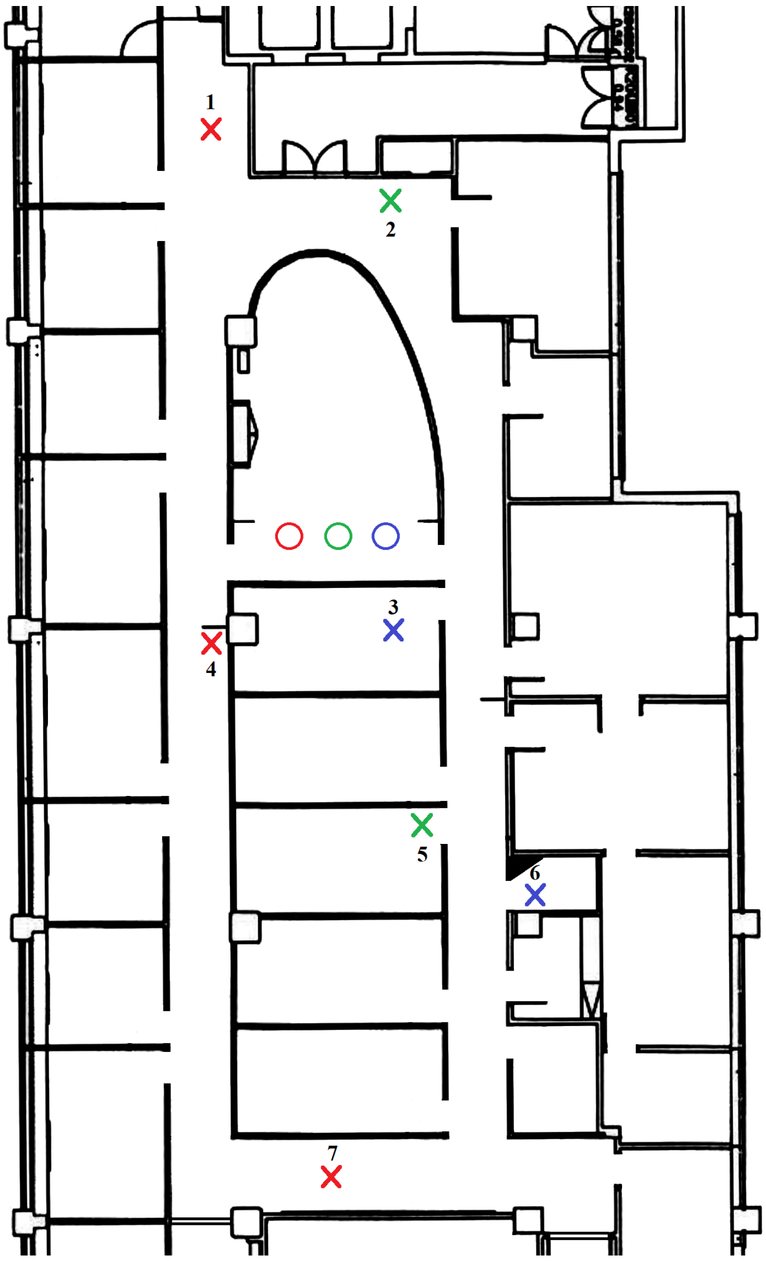


FIGURE 4.8: Office setting for scenario S_1. The different coloured circles represent ROBOT_1 (the red circle), ROBOT_2 (green circle) and ROBOT_3 (blue), and the corresponding crosses represent task locations that will be assigned to those robots.

4.3.2 Extended Performance Metrics

By evaluating the results from Section 4.2, two new performance metrics are defined. The extended performance metrics are specific to communication performance and enable the framework to analyse the success rate of critical communication during a mission. The extended performance metrics are:

- *Average Failed Tasks*: this is the average number of tasks that failed to be communicated per mission, which directly constitutes the overall success/failure of a mission.
- *Successful Missions*: I define successful missions by the advent of successful communication (i.e. no failed tasks). This is achieved when **all** *task status messages* are communicated to all agents during a mission. A mission fails when a *task status message* fails to be received by the collective team, thus at the end of a mission if a particular task(s) is unresolved it is assumed incomplete.

4.3.3 Results

The metrics *Execution Phase Time* (Figure 4.9), *Total Movement Time* (Figure 4.10), *Total Distance Travelled* (Figure 4.11), *Overall Near Collisions* (Figure 4.12), *Average Failed Tasks* (Figure 4.13) and *Successful Missions* (Figure 4.14) are plotted as before, where each figure contains two sub-figures, A and B, representing simulated and physical experiment results, respectively. Furthermore, the Mann-Whitney U test is used to investigate the $\mathbf{H_0}$ and $\mathbf{H_A}$ hypotheses, on whether the observations from the ideal case data sample (*SPL0*) are inclined to be equivalent or less than the other data samples.

The *Execution Phase Time* metric demonstrates some contradictory results between the U-value and p-value, and the p-value and the plots. The highest U-value possible for any data samples is the product of the sample sizes being compared, i.e. $U_{max} = 30 \cdot 30$ for simulation or $U_{max} = 5 \cdot 5$ for physical, and as previously stated a higher U-value suggests that the $\mathbf{H_0}$ is more likely. The p-value simulation results for *Execution Phase Time* show that $\mathbf{H_A}$ is accepted for *SPL50* and the U-value is moderately high. Moreover, the plot in Figure 4.9A does not show any obvious trend and the mean values, rounded up to 1 decimal point, for each sample are 194.8s (*SPL0*), 190.8s (*SPL25*), 194.5s (*SPL50*) and 193.1s (*SPL75*). The averages are all within standard deviation of each other and they highlight a weakness in the Mann-Whitney U test. The standard deviation and positioning of sample observations can have an impact on rank scores of the U-value. If the differences between observations are very minor, but still favour one sample over another, it can skew the results. The U-value and p-value for the physical experiments are clearly unreliable due to the sample size of the data and are used as a proof-of-concept and to have some comparison to the simulated experiments. Therefore, by visual observation and analysing the mean values it can be argued that the *Execution Phase Time* is not affected by increasing message loss.

The *Total Movement Time* metric (Figure 4.10), specifically for simulated experiments, shows a distinct decreasing time pattern between network parameters *SPL0* and *SPL50* or greater. However, the p-value, shown in Tables 4.9 and 4.10, is still within the CI, thus it does not show statistical significance. The *Total Distance Travelled* metric (Figure 4.11) shows the same trends for both the simulated and physical experiments as that of *Total Movement Time*. There is no statistical significance in both simulated and physical experiments.

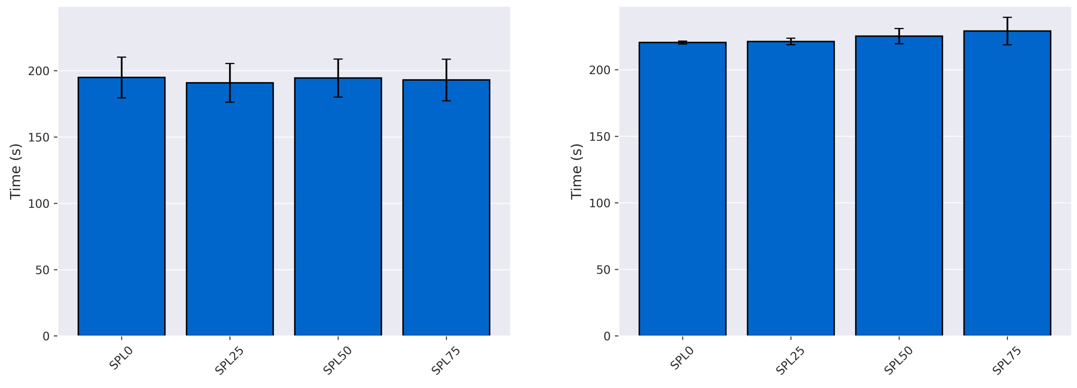
The *Average Failed Tasks* and *Successful Missions* metrics are required to analyse the impact of increasing network perturbation (*SPL*) on communication performance. The *Average Failed Tasks* metric shows the average amount of tasks **not** received (communicated) per robot, which the *assigner* agent flags as failed at the end of a mission. The *Successful Missions* metric is the number of missions where no communication failure of a task occurred for any robot. Two contingency analysis Tables 4.7 and 4.8, and Figures 4.14 and 4.13 are used to illustrate these metrics. Tables 4.7 and 4.8, representing the contingency analysis of the simulated and physical experiments respectively, reveal a drastic drop in *Successful Missions* after *SPL25*. The result from the Tables and plots is powerful enough that further statistic analysis is not necessary, nonetheless the Chi-Squared test is performed on the result using the same Python statistics library [100]. For both Tables a CI of 95% is used and the calculated Chi-Squared statistic and p-values are identical (68.0 and 0.0), clearly indicating that increasing network perturbations heavily impact communication performance within a robot team. This is further supported by the fact that Figure 4.13 for *SPL25* and above, exhibits on average at least one *task status message* failure.

Network Perturbation	SPL0	SPL25	SPL50	SPL75
Successful Missions				
YES	30	6	0	0
NO	0	24	30	30

TABLE 4.7: Contingency table of the simulation experiments showing the correlation between the number of *Successful Missions* and *SPL* network perturbation.

Network Perturbation	SPL0	SPL25	SPL50	SPL75
Successful Missions				
YES	5	0	0	0
NO	0	5	5	5

TABLE 4.8: Contingency table of the physical experiments showing the correlation between the number of *Successful Missions* and *SPL* network perturbation.



(A) Simulation (B) Physical

FIGURE 4.9: MRComm results for *Execution Phase Time*, with increasing *SPLX*.

SPL0 versus Perturbation	**U**	**p**	**effect size**	**Accepted Hypothesis (H_0/H_A)**
Execution Phase Time				
SPL25	538	0.90	0.24	H_0
SPL50	328	0.04	-0.33	H_A
SPL75	304	0.02	-0.39	H_A
Total Movement Time				
SPL25	480	0.67	0.08	H_0
SPL50	520	0.85	0.19	H_0
SPL75	556	0.94	0.29	H_0
Total Distance Travelled				
SPL25	514	0.83	0.17	H_0
SPL50	510	0.81	0.16	H_0
SPL75	531	0.89	0.22	H_0

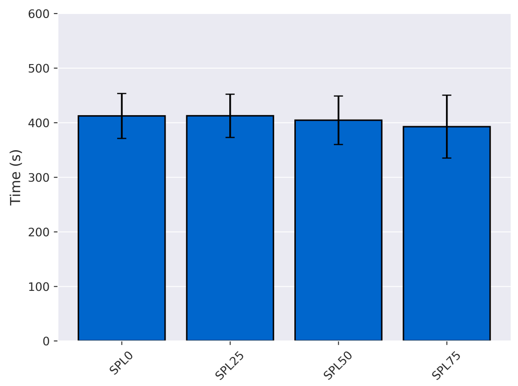
TABLE 4.9: MRComm simulated robot experiments' U-value, p-value, effect size and hypothesis outcome for only the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

SPL0 versus Perturbation	**U**	**p**	**effect size**	**Accepted Hypothesis (H_0/H_A)**
Execution Phase Time				
SPL25	10	0.34	-0.23	H_0
SPL50	5	0.07	-0.70	H_0
SPL75	1	0.01	-1.07	H_A
Total Movement Time				
SPL25	15	0.73	0.23	H_0
SPL50	13	0.58	0.05	H_0
SPL75	8	0.20	-0.42	H_0
Total Distance Travelled				
SPL25	18	0.89	0.51	H_0
SPL50	17	0.85	0.42	H_0
SPL75	12	0.5	-0.05	H_0

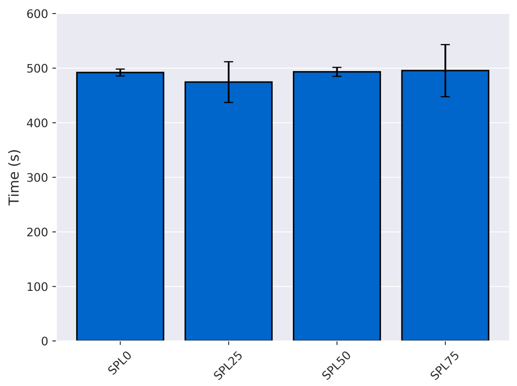
TABLE 4.10: MRComm physical robot experiments' U-value, p-value, effect size and hypothesis outcome for only the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

4.3.4 Discussion

Total Movement Time (Figure 4.10) and *Total Distance Travelled* (Figure 4.11), show similar results and reveal that degrading communication quality (i.e. increase in *SPL*) did not impact MRT performance to the same degree as previously seen by the results obtained using *MRTeAm*. *Execution Phase Time* (Figure 4.9) indicates that there is statistical significance with degrading

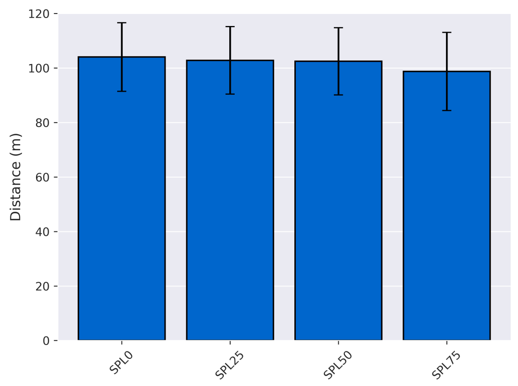


(A) Simulation

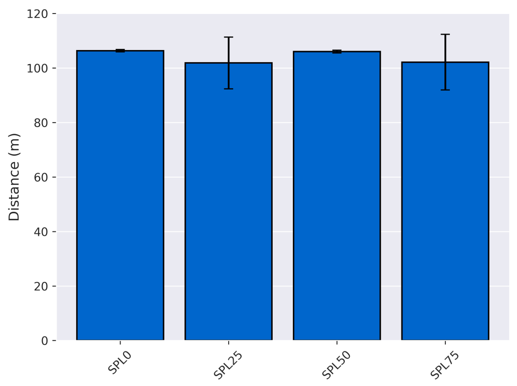


(B) Physical

FIGURE 4.10: MRComm results for *Total Movement Time*, with increasing *SPLX*.

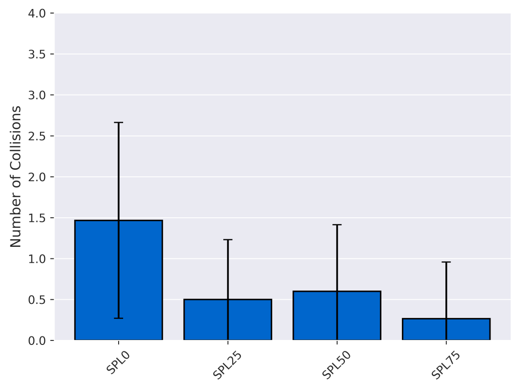


(A) Simulation

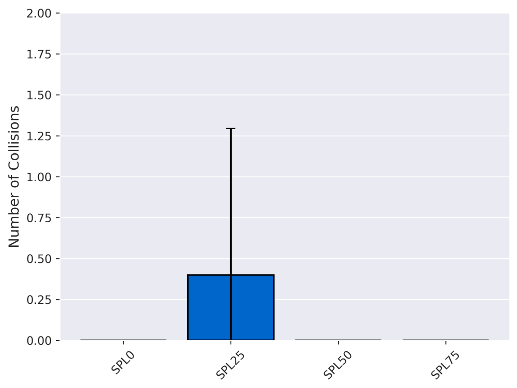


(B) Physical

FIGURE 4.11: MRComm results for *Total Distance Travelled*, with increasing *SPLX*.

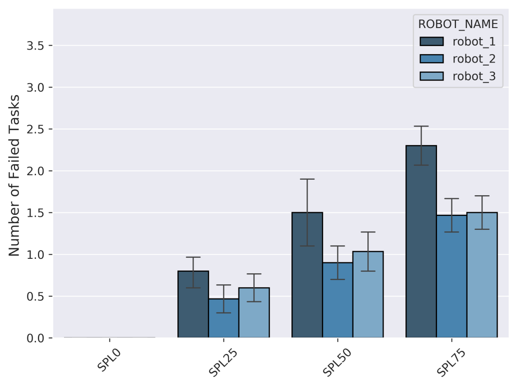


(A) Simulation

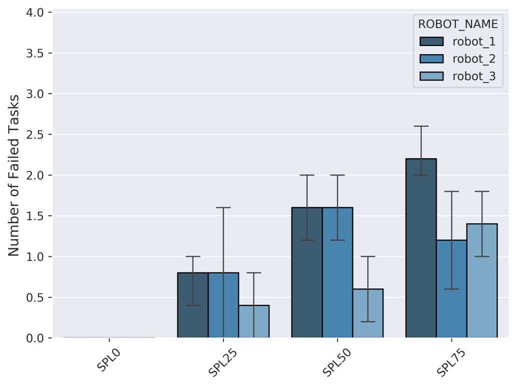


(B) Physical

FIGURE 4.12: MRComm results for *Overall Near Collisions*, with increasing *SPLX*.

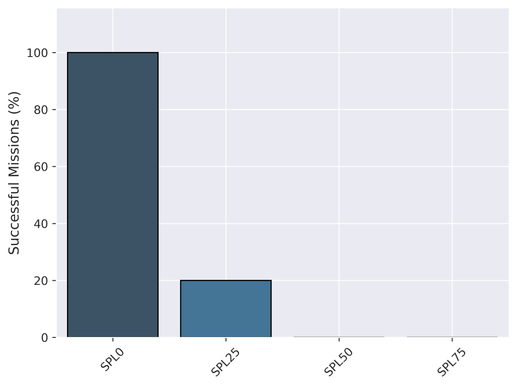


(A) Simulation

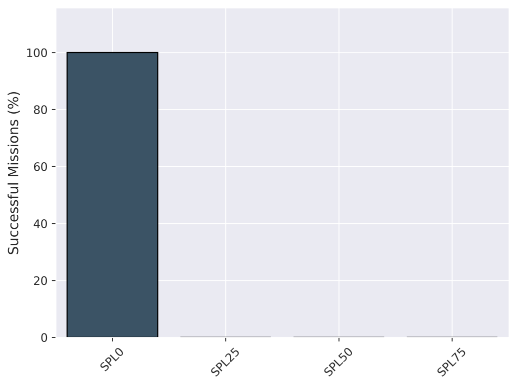


(B) Physical

FIGURE 4.13: MRComm results for the *Number of tasks failed to communicate per robot*, with increasing $SPLX$.



(A) Simulation



(B) Physical

FIGURE 4.14: MRComm results for the percentage of *Successful Missions* completed, with increasing $SPLX$.

communication quality. However, visual inspection, comparing the mean and standard deviations contradicts the Mann-Whitney U test result, inclining toward observations from the compared samples to be of similar magnitude. Overall these three performance metrics demonstrate consistent performance and robust communication behaviour.

The extended performance metrics, *Average Failed Tasks* and *Successful Missions*, show that beneath the "consistent" core performance metrics there remains a critical impact on communication performance on the multi-robot team, which heavily impacts overall mission success rate. As depicted in Section 4.3.1, ROBOT_1 contains three tasks in its agenda and therefore has the highest likelihood (shown in Figure 4.13) to fail to communicate the *task status message* of at least one of its three tasks to the rest of the robot team and the *task assigner* agent.

4.4 General Discussion

Direct comparison and correlation between experiments conducted with *MRTeAm* and *MRComm* can not be made as the scenarios used were varied. However, as the experiment configurations are identical (not considering the scenario changes) comparisons can be made in emerging trends and the results between the two frameworks are paramount. For example, the results obtained using *MRTeAm*, in Section 4.2, demonstrated discrepancy in the core performance metrics when impacted by a network perturbation. The results revealed that communication is an issue in a general MRS framework, but was limited in its feature set and did not reveal how communication was specifically impacted. Contrarily, the results obtained using *MRComm*, in Section 4.3, demonstrated that a framework with better communication middleware support, *FKIE*, can improve the core capabilities of a robot team making them more consistent. For example, increasing network perturbation did not impact performance metrics and there were no emergent trends for the *MRComm* framework. However, the extended performance metrics, introduced in *MRComm*, revealed that communication performance was critically being impaired by *SPL*.

The communication design of *MRTeAm* and general MRTs is not geared toward imperfect communication, therefore it is not surprising that the core performance metrics are affected by network perturbations. *MRTeAm* implements its own *ROS Master Bridge* software for shared communication, which is not as robust to compromised communication. The *ROS Master Bridge* connects multiple ROS masters (robots) using a special queuing server, which is run centrally, for example on a remote server. Each robot has a local *master bridge relay* node that is responsible for publishing both **local** and **received** messages from the master bridge. Apart from the central server required to host the *ROS Master Bridge*, it is the local *master bridge relay* node that is the limiting factor, which reduces *MRTeAm*'s consistency and robustness for multi-robot communication, i.e. shared messages required locally are aslo affected by *SPL*. Upon introducing network perturbations it is noted that the communication design for the MRT using *MRTeAm* is not performing as expected. *MRComm*'s success in improving communication over *MRTeAm* comes from the change in communication design. As previously mentioned, the *MRComm* framework employs the *FKIE* software for multi-node communication, which allows each *pub-sub* messaging topic to be locally accessible to each robot with the *added* functionality to then publish the shared/receiving topics to other robots.

4.5 Summary

It is found from the results in Sections 4.2.3 and 4.3.3 that a standard MRS (*MRTeAm*), which does not focus on communication design, has drawbacks when it comes to consistency of communication and results, compared to a MRS (*MRComm*), which focuses on communication design. *MRComm* fixes the inconsistency of communication present in *MRTeAm* across the core performance metrics. However, the newly introduced communication-specific performance metrics reveal that network perturbations immensely impact the successful operation of a robot team in both the simulated and physical environment.

Chapter 5

MRComm: Simulation Experiments

5.1 Introduction

The overarching aim of the experiments presented here is to analyse how the different *simulated* network parameters (type and perturbation) affect communication performance of a truly decentralised MRT, while the MRT performs task execution and coordination using two different behaviours, *NB* and *LF*. Moreover, possible improvements to communication performance are evaluated for the novel *LF* behaviour over the *NB* behaviour.

The experiments presented here are performed using the same robot software framework (ROS) and simulation environment (Stage), which is described in Chapter 3. The simulated environment imitates noise present in odometry and localisation sensors. However, communication conditions are ideal, apart from the simulated network perturbations I introduce to the environment; there is no additional uncertainty or noise presented from external sources as there would be in a physical environment. Therefore, I expect there to be some minor disparity between the results presented here and those in Chapter 6. However, it is beneficial to use the simulated environment as it is convenient to run experiments continuously and obtain a larger data-set, and to test new features/parameters before bringing them out in the real-world. Larger data-sets can be more accurately validated for mean values, reveal outliers (shortcomings) of the *MRComm* system and improve statistical power of the analysis to support research findings. All experiments conducted in this chapter were performed on simulated robots using the S_1 scenario, defined in Chapter 4. The same scenario is used for experiments presented in Chapter 6, so that comparisons can be made between the simulated and physical environments for corresponding experiment configurations.

In this chapter, two new network perturbations are introduced and described, namely *Simulated Loss Threshold* (*SLT*) and *Simulated Signal Degradation* (*SSD*). All network perturbations impact the same shared message topics as before, *pose messages* and *task status messages*. Moreover, the two network types defined in Chapter 3, *WLAN* and *AH*, are used in the experiment configurations. The general experiment schema used to represent experiment configurations, described in Section 3.10, is adapted to include two network perturbations simultaneously. The performance metrics are extended to accommodate for experiments that use the novel *LF* behaviour. A portion of the experiments and accompanying results shown in this chapter were published in [88].

The rest of this chapter is structured as follows. Sections 5.2 and 5.3 define and describe the *SLT* and *SSD* network perturbations, respectively. Section 5.4 outlines the methodology and setup used for performing the simulated experiments to analyse communication performance of the *NB* and *LF* behaviours. Furthermore, this section presents the extended performance metrics required to evaluate the *LF* behaviour and the alteration made on the experimental schema design. Section 5.5 presents the experimental results and Section 5.6 discusses the results and issues raised in the context of the experiments. Finally, Section 5.7 brings the chapter to a close with a brief conclusion and summary.

5.2 Simulated Loss Threshold (SLT)

SLT is defined as a network perturbation, however it does not perturb the signal strength, it is more of a threshold that enables robots to "sense" signal strength. It is specifically designed for simulated experiments to test robot communication capabilities when using the *AH* network type. Furthermore, it represents the effect that signal strength can have on a MRT.

The *SLT* employs a threshold *warning* and a threshold *limit* based on the distance between robots, which are defined by the analysed signal strength in an outdoor environment between two laptops (robots), described in Chapter 3. The results are re-plotted here for convenience, in Figure 5.1, and to highlight the *SLT* thresholds. The *SLT* threshold *warning* is an internal *MRComm* function, which simply warns the robots that signal strength is becoming weaker between the team members and a disconnect is imminent. The *SLT* threshold *limit* is the final internal *MRComm* function that tells the robots that shared communication will cease, until the robots are within range again. The *SLT*'s threshold *warning* is defined to be 4.0 m, as depicted by the observed drop in signal strength, Figure 5.1. The *SLT*'s threshold *limit* is in fact the *AH* network's hypothetical communication limit described in Section 3.8.1, which is 8.0 m.

The *SLT* thresholds are static and do not change depending on the environment or network type that is used. This carries both positive and negative connotations. Advantages of this design is that it is simple and allows *SLT* to be used interchangeably for simulated and physical robot experiments and any network type. The simplistic design is also its greatest weakness, as it limits the accuracy of how true signal strength behaves under different network conditions.

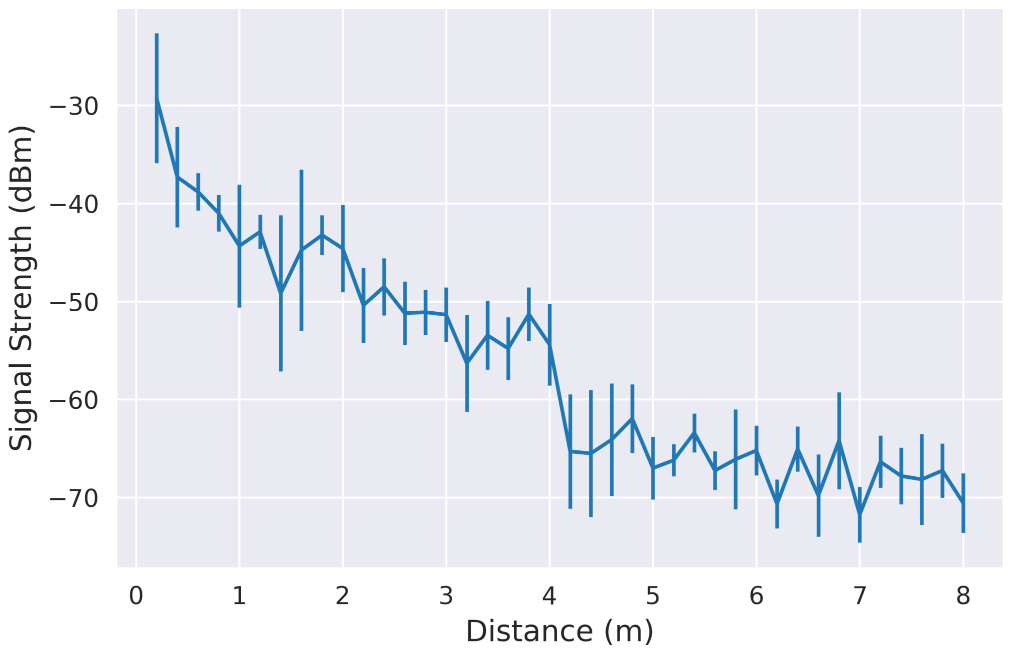


FIGURE 5.1: Average signal strength (30 repeat readings per data point) vs. distance in an outdoor environment. Re-plotted from Chapter 3

5.3 Simulated Signal Degradation (SSD)

The main difference between *SSD* and *SLT* is that *SLT* is a simple static threshold function based on estimating signal strength with change in distance between robots. *SLT* is not accurate when it comes to predicting actual signal strength degradation and lacks the dynamics with which to demonstrate the effects that the environment (simulated or physical) has on signal strength. The *SSD* network perturbation is developed to improve these shortcomings by using Support Vector Regression (SVR) models to generate simulated signal strength based on the distance between robots.

Using the assumptions made on the network types in Chapter 3, *SSD* can only be used with experiments that employ the *AH* network type and not the *WLAN* network type. Furthermore, the signal strength results obtained for the *SSD* perturbation are modelled in the operational (in-door) environment where multi-robot experiments are conducted, unlike how the

SLT thresholds are modelled on signal strength in an outdoor environment. The *SSD* thresholds are modelled using two separately trained SVR models with Radial Basis Function kernels (RBF): the first model predicts the signal strength between robots in direct line-of-sight of each other, and the second predicts the signal strength for obstructed line-of-sight, as seen in Figures 5.3a and 5.3b. The RBF kernel equation used is:

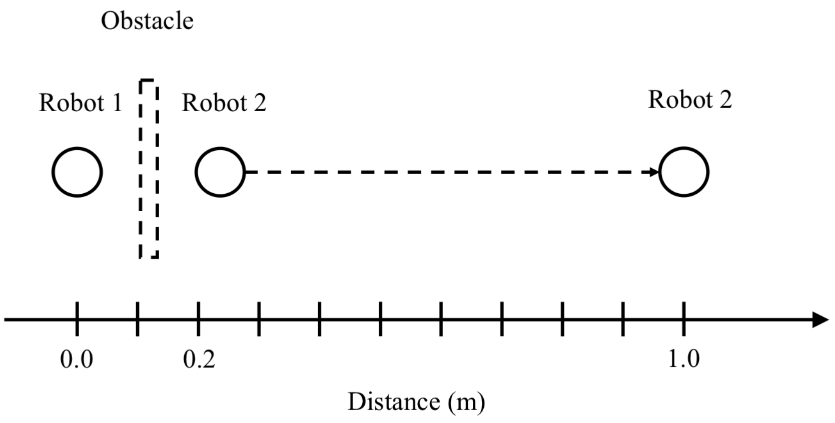
$$k(\mathbf{x}, \mathbf{c}) = \exp\left(-\gamma||\mathbf{x} - \mathbf{c}||^2\right) \tag{5.1}$$

where $\mathbf{c}$ is the function's centre, $||\mathbf{x} - \mathbf{c}||$ represents Euclidean distance and $\gamma = 1/2\sigma^2$ is set to 0.1 using the *scikit-learn* [100] library[1] to train the SVR models. Furthermore, this package contains a constant C, which denotes a penalty parameter for the error term in the SVR used, which is set to 100. This parameter is introduced to trade correct classification with maximising the decision function's margin. A larger value for C minimises the margin, thereby improving accuracy for the model at the cost of increasing the computational complexity.

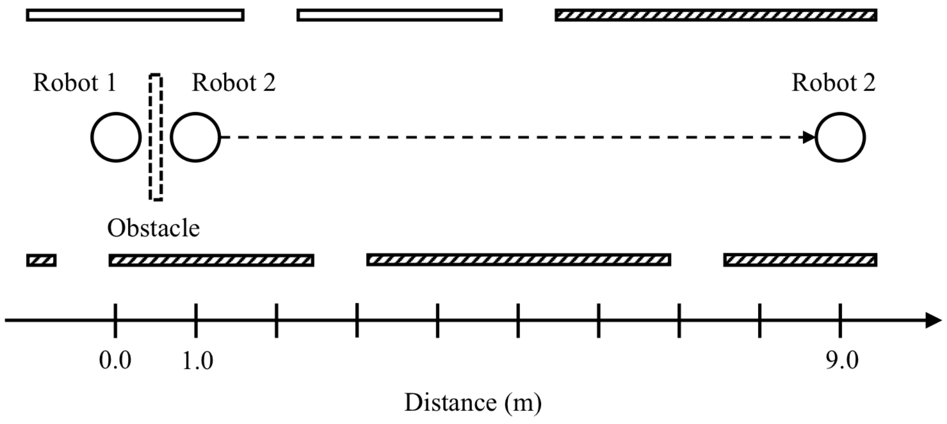
In total, four tests were performed taking 20 repeat readings at each distance point, which generated four data-sets. The tests are initially divided into two levels of granularity (0.1 m and 1 m steps), since signal strength accuracy proportionally decreases with increasing distance, as mentioned in [101]. Each of the granularity tests are repeated, once with the robots in direct line-of-sight and once while the robots' line-of-sight is obstructed. The 0.1 m increment test illustrated by Figure 5.2a picks up accurate and fine changes in the signal strength, and overall improves the SVR models. The 1 m increment test illustrated by Figure 5.2b shows the changes over the span of the imposed *AH* limit (i.e., $\approx$ 8 m). Furthermore, it was noted that the metal window blinds in the office environment (see Figure 5.2b) seem to cause constructive interference at $\approx$ 3 m and $\approx$ 6 m shown by the average

[1]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

signal strength (red markers) in Figure 5.3a. The two granularity tests for direct/obstructed line-of-sight are combined to represent two final data-sets one for each type of line-of-sight. The two data-sets are each used to train an SVR to predict dynamic signal strength during a multi-robot experiment. The *SSD warning* threshold remains the same throughout an experiment, however signal strength is dynamically predicted depending on the nature of the environment, rather than the distance of the robots. The *SSD warning* threshold value is defined in Table 5.1.

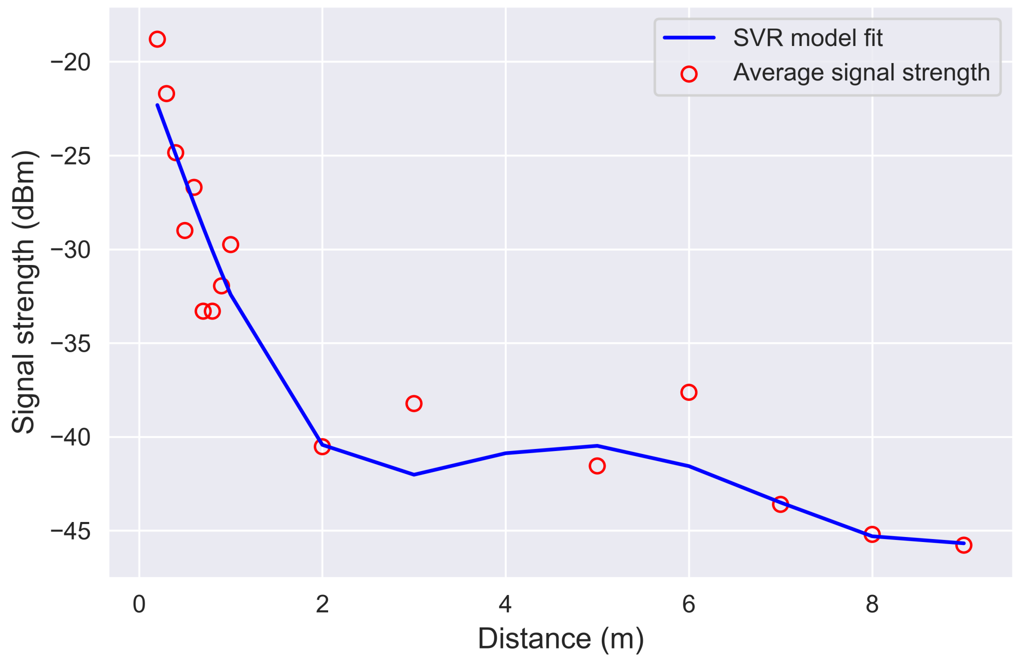


(a) Diagram showing fine grained signal strength analysis with distance range from 0.2 m to 1.0 m. The dotted obstacle was removed when performing direct line-of-sight analysis.

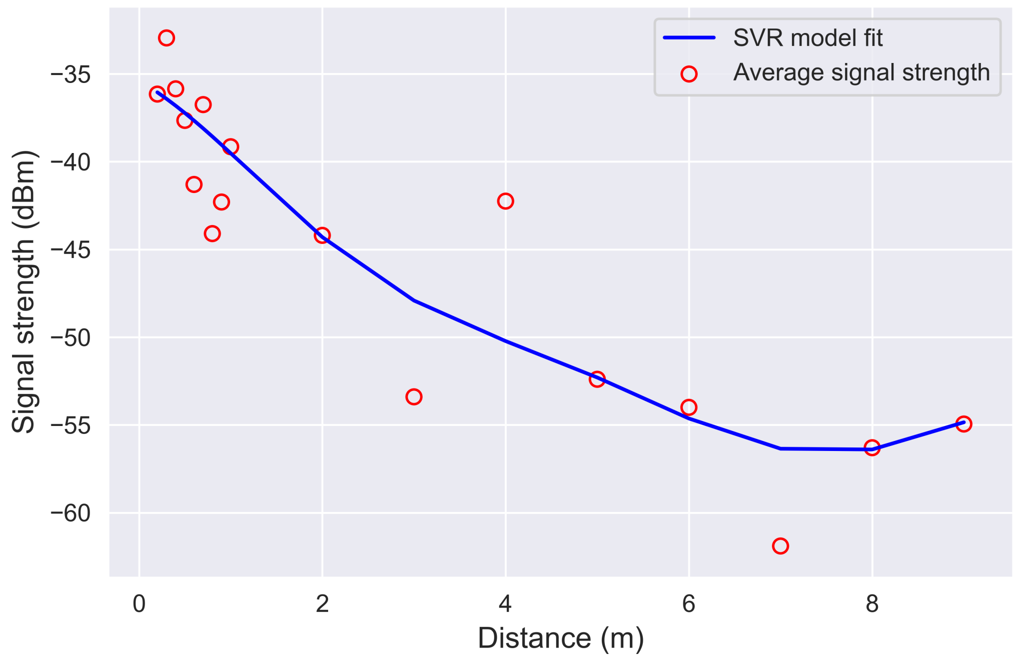


(b) Diagram showing extended signal strength analysis with distance range from 1 m to 9 m. The dotted obstacle was removed when performing direct line-of-sight analysis. The bars on each side of the robots represent walls. Diagonally striped bars represent metal window blinds.

FIGURE 5.2: Signal strength evaluation in operational environment for *SSD*.



(a) SVR model of signal strength with direct line-of-sight.
Red markers represent the average signal strength at the respective distance.



(b) SVR model of signal strength with obstructed line-of-sight.
Red markers represent the average signal strength at the respective distance.

FIGURE 5.3: SVR models.

5.4 Experiments

Missions are executed using *Stage* simulator on a powerful workstation computer, which simulates the MRT and *task assigner* agent. The workstation computer consists of an i7-6700 CPU, 16 GB of RAM and two lower-end AMD FirePro W4100 GPUs. The GPUs were not fully utilised for the simulation as they were used for rendering the GUI, simulated environment and robots, which was a considerably lighter load compared to the system software, complex navigation and simulated sensor signals running simultaneously using CPU resources.

Task execution agents are assigned tasks, by the *task assigner* agent, to complete. Each task definition includes a location where the robot performs actions, such as *sensor-sweep* (e.g. collecting a series of images). At the start of a mission, in order to coordinate team activity, a *task assigner* agent determines which robots should perform which tasks and thereafter mission execution begins. At the end of an experiment, the *task assigner* agent records all shared messages and performance metrics, and concludes by determining which robot(s) failed to transmit "SUCCESSFUL" completion of *task status messages*.

As previously mentioned, the network parameters are split into two categories, namely network types and network perturbations. The network types explored in this chapter are "simulations" of *WLAN* and *AH*. The network perturbations tested in these experiments are:

- Simulated Packet-Loss (*SPLX*, where $X \in \{0, 25, 50, 75\}$), see Section 3.8.3;
- Simulated Loss Threshold (*SLT*), see Section 5.2;
- Simulated Signal Degradation (*SSD*), see Section 5.3.

5.4.1 Network Perturbations and Thresholds

The *SLT* network perturbation is static and remains the same for all experiment configurations that use it. However, the *SSD warning* threshold is set to a signal strength value, which is predicted using SVR models. Therefore, the distance at which the *warning* threshold is triggered is dynamic, this is further highlighted in Table 5.1. It is important to note that the *AH* network limit of 8.0 m is used to define the *limit* threshold for both *SLT* and *SSD*, unless otherwise specified.

As defined in Section 5.2, the *SLT* perturbation is a baseline perturbation with a *warning* threshold of 4.0 m, shown in Table 5.1. The focus of the robot team employing the *LF* behaviour is to monitor the *SLT warning* threshold, which is half that of the *limit* threshold. The *SLT warning* threshold is designed with the purpose of alerting a MRT to stay within communication range of the *AH* network.

The *SSD* perturbation alters how robots using the *LF* behaviour react compared to using *SLT*. As mentioned, the *SSD warning* threshold is static, however the simulated signal strength changes depending on the dynamic changes of the environment. This enables it to predict realistic signal strength values, unlike the static *warning* threshold of *SLT* that is based only on distance between robots. The *SSD warning* threshold for the experiments presented here is set to -42 dBm, shown in Table 5.1. If robots using *LF* and *SSD* reach this threshold signal strength they will be warned that a disconnect is imminent.

Network Perturbation Warning Thresholds			
Network Perturbation	Warning Threshold	**Distance (approx.)**	
		Line-of-sight signal	Obstructed signal
SLT	4.0 m	$d > 4.0$ m	$d > 4.0$ m
SSD	-42.0 dBm	2.6 m $< d <$ 3.6 m and $d >$ 6.1 m	$d >$1.6 m

TABLE 5.1: Demonstrates the *warning* thresholds and shows approximate distance when the system is warned of disconnecting (LF behaviour only).

5.4.2 Extended Performance Metrics (for LF)

The extended metrics in this section are required to fully analyse the performance of a MRT that employs the *LF* behaviour. As defined in Section 3.9, *Initialisation Time* (light brown/dark orange) and *Task Execution Time* (yellow), are part of the metrics that are recorded, but are not individually analysed.

- *Initialisation Time* (Assigner Time): time metric defined as the time from when the *task assigner* sends an experiment start message, a robot behaviour is initialised and the robot has began to move. Represented by the light brown/dark orange colour in Figure 5.4.

- *NA Movement Time*: not assigned movement time is the time the robot spends moving toward its task while not having a role. If a behaviour is assigned at the start of an experiment, NA movement time varies according to how the assigned roles are defined by the behaviour (i.e. how the environment affects communication). Otherwise, if no-behaviour (*NB*) is assigned, then NA movement time varies as a result of the network perturbation chosen and dynamic events taking place in the environment (e.g. obstacles, collisions). This is represented by the light green colour in Figure 5.4.

- *Leader Movement Time*: this metric is defined as the amount of time a robot has been moving while assigned the role of "Leader" in accordance with the chosen behaviour. Represented by the dark green colour Figure 5.4.

- *Follower Movement Time*: this metric is defined as the amount of time a robot has been moving while assigned the role of "Follower" in accordance with the chosen behaviour. Represented by the turquise colour in Figure 5.4.

- *Leader Disconnect Time*: the leader disconnect time occurs when a robot, assigned as leader, is considered too far from the furthest follower robot(s) and is required to stop and wait for them to be within acceptable distance before beginning to move again. The leader disconnect distance value is based on the type of network threshold chosen (i.e. *SLT*, *SSD*). Represented by the orange colour in Figure 5.4.

- *Follower Waiting Time*: follower waiting time occurs when two conditions are true for the follower robot(s). The first condition is that the follower robot needs to have reached the last pose received from the leader robot. The second condition is that any one of the followers should be out of connection range of the leader robot. Therefore, if the first condition is met by the follower closer to the leader and the second condition is true then the closer robot will enter the waiting state. Once the leader has reconnected to all followers and begins moving, the follower waiting time ends. Represented by the red colour in Figure 5.4.

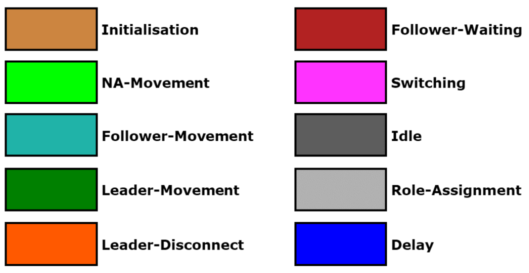
- *Switching Time*: once a robot with an assigned role has successfully completed the role's constraints, it is reassigned to use the general behaviour (NA). The time it takes to switch from an assigned role back to NA behaviour is the switching time. Represented by the light purple colour in Figure 5.4.

- *Role Assignment Time*: the time taken for a robot to calculate its utility function, send the result, and compare the team's results and get assigned a role. Represented by the light grey colour in Figure 5.4.

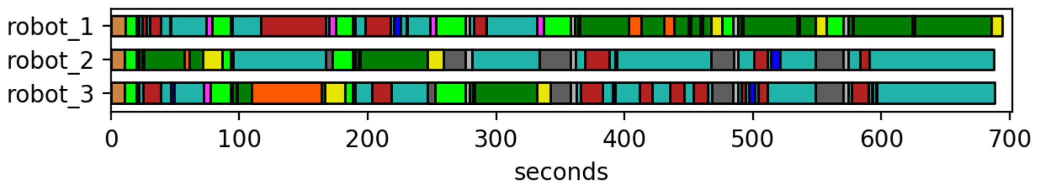
- *Overall Idle Time* (redefined for LF): once a robot has completed all its tasks and waits for the remainder of the team to complete their tasks, if an assigned leader disconnects from the idle robot it forces it to be reassigned as a follower, thus ending its idle time. Represented by the dark grey colour in Figure 5.4.

- *Minimum Separation*: the minimum distance recorded to the nearest robot throughout the duration of the mission. This is illustrated by a light blue arrow on Figure 5.5.

- *Maximum Separation*: the maximum distance recorded to the furthest robot throughout the duration of the mission. This is illustrated by a red arrow on Figure 5.5.



(a) Timeline legend



(b) A timeline presenting time-based performance metrics and how they are calculated during an experiment (example configuration F_{25} from Table 5.3).

FIGURE 5.4: Timeline of an experiment using the extended performance metrics.

5.4.3 Experiment Setup

Scenario S_1, see Section 4.3.1, is used for the simulated experiments here. Scenario S_1's task locations are purposefully positioned in difficult to reach and narrow spaces and starting locations for the robot team are suboptimal. The starting location space has only two opened doors, which allow for only

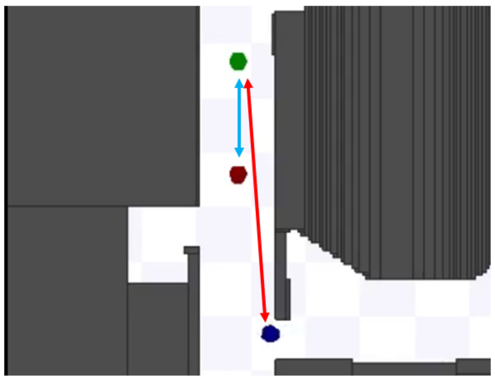


FIGURE 5.5: Simulation of robot team, presenting how the minimum and maximum separation metrics are calculated.

one robot to enter/exit at a time. The starting location space is a major disadvantage to the robots. If the robots are using the *LF* behaviour and a robot disconnects very soon in the starting location space, there is a high probability of it stopping in front of a doorway, which will delay the robot team. If the robots are using the *NB* behaviour and a robot simply blocks a doorway, it will cause the robot(s) trapped inside the starting location to fail in executing their planned path, therefore it will either force that robot to re-plan a new path or the robot will be "stuck" in a planning loop until the blocking robot(s) continues moving/becomes unstuck. However, the starting location is practical from the point of view of a hypothetical disaster scenario. For example, a central starting location for a decentralised MRT performing search and rescue tasks, will enable robots to have a minimum distance to any far reaching corner of the operational environment. As before, tasks T_R are assigned sequentially using RR 1 assignment to each robot R, and the assignments are fixed for all experiments. *robot*_1 is assigned tasks T_1 = {1,4,7}, *robot*_2 is assigned tasks T_2 = {2,5} and finally *robot*_3 is assigned with tasks T_3 = {3,6}, as previously illustrated in Figure 4.8.

The reason another robot was added to the team was to investigate how *MRComm* could cope with more team members and to test more communication points (i.e. more robots) with the network parameters in this chapter and Chapter 6. The rationale for choosing an odd number of tasks, i.e. 7, with task locations that are very distant from each other was to initially make sure that the RR algorithm was functioning as expected, but there are two additional reasons. Firstly, to make sure that for the *LF* behaviour the leader robot being picked was a different robot each time, i.e. robots are taking turns completing their agenda, and to analyse how the team would perform when one robot had more tasks (specifically if there was any impact on communication). Secondly, the distance of the task locations was strategic, as it made robots obstruct each other and this would require more shared communications to occur, which could be analysed to see how communication was affected for the novel *LF* behaviour.

For the experiments conducted here and Chapter 6, a modified experiment schema is used. It is more realistic to experience multiple different network perturbations during an MRT mission. Therefore, the new experiment design takes this into account and adds network thresholds (i.e. *SLT* and *SSD*) as the second network perturbation. The original schema in equation 3.10 is changed here to equation 5.4.3. For equation 5.4.3 the initial network perturbation remains the level of message loss (*SPL*) and the second network perturbation analyses degrading signal strength (*SLT*, *SSD*).

$$\begin{aligned} F_y = \{S_i\} \times \{NET\_TYPE\} \times \{NET\_PERTURBATION_1\} \\ \times \{NET\_PERTURBATION_2\} \times \{ROBOT\_BEHAVIOUR\} \end{aligned}$$

where S_i is the scenario, NET_TYPE is the *WLAN* or *AH* network type, $NET\_PERTURBATION_1$ is the first network perturbation (i.e. *SPL*), $NET\_PERTURBATION_2$ is the second network perturbation (i.e. *SLT*, *SSD*) and $ROBOT\_BEHAVIOUR$ is the specified MRT behaviour, either *NB* or

LF.

The simulated experiments are each repeated 30 times. For clarity, the experiments conducted in this chapter are listed in Tables 5.2 and 5.3 representing experiments for *WLAN* and *AH* network types, respectively. Table 5.2 consists of 8 configurations each performed 30 times, therefore 240 experiment runs in total. Tables 5.3 consists of 16 configurations each performed 30 times, therefore 480 experiment runs in total.

Experiment schema
$F_5 = \{S_1\} \times \{WLAN\} \times \{SPL0\} \times \{SLT\} \times \{NB\}$
$F_6 = \{S_1\} \times \{WLAN\} \times \{SPL25\} \times \{SLT\} \times \{NB\}$
$F_7 = \{S_1\} \times \{WLAN\} \times \{SPL50\} \times \{SLT\} \times \{NB\}$
$F_8 = \{S_1\} \times \{WLAN\} \times \{SPL75\} \times \{SLT\} \times \{NB\}$
$F_9 = \{S_1\} \times \{WLAN\} \times \{SPL0\} \times \{SLT\} \times \{LF\}$
$F_{10} = \{S_1\} \times \{WLAN\} \times \{SPL25\} \times \{SLT\} \times \{LF\}$
$F_{11} = \{S_1\} \times \{WLAN\} \times \{SPL50\} \times \{SLT\} \times \{LF\}$
$F_{12} = \{S_1\} \times \{WLAN\} \times \{SPL75\} \times \{SLT\} \times \{LF\}$

TABLE 5.2: Experiment configurations for WLAN network type.

Experiment schema
$F_{13} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{SLT\} \times \{NB\}$
$F_{14} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{SLT\} \times \{NB\}$
$F_{15} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{SLT\} \times \{NB\}$
$F_{16} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{SLT\} \times \{NB\}$
$F_{17} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{SSD\} \times \{NB\}$
$F_{18} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{SSD\} \times \{NB\}$
$F_{19} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{SSD\} \times \{NB\}$
$F_{20} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{SSD\} \times \{NB\}$
$F_{21} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{SLT\} \times \{LF\}$
$F_{22} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{SLT\} \times \{LF\}$
$F_{23} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{SLT\} \times \{LF\}$
$F_{24} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{SLT\} \times \{LF\}$
$F_{25} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{SSD\} \times \{LF\}$
$F_{26} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{SSD\} \times \{LF\}$
$F_{27} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{SSD\} \times \{LF\}$
$F_{28} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{SSD\} \times \{LF\}$

TABLE 5.3: Experiment configurations for AH network type.

5.5 Results

The performance metrics presented here are *Total Movement Time* (Figure 5.6), *Leader Disconnect Time* (Figure 5.7), *Follower Waiting Time* (Figure 5.8), *Overall Near Collisions* (Figure 5.9), *Overall Delay Time* (Figure 5.10), *Overall Idle Time* (Figure 5.11), *Total Distance Travelled* (Figure 5.12), *Average Failed Tasks* (Figure 5.13), *Successful Communication* (Figure 5.14) and *Minimum and Maximum Separation* (Figure 5.15). Each figure representing a performance metric is split into three sub-figures representing experiments performed with: first the *WLAN* network type and the *SLT* network perturbation; second the *AH* network type and the *SLT* network perturbation; third the *AH* network type with the *SSD* network perturbation. The error bars in the figures reflect the standard deviation between runs.

The Mann-Whitney U test is performed as before, investigating the same hypotheses, where $\mathbf{H_0}$ checks if the pairwise observations from independent samples are equivalent or, $\mathbf{H_A}$ checks if the observations from the ideal case (i.e. *SPL0*) are less than the other (i.e. *SPLX*) observations. Tables 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10 are used to show the Mann-Whitney U analysis, including the effect size. However, it should be noted that if a performance metric (i.e. *Follower Waiting Time*) or a certain experiment label is **missing** from the tables it is because the analysis did not reveal any significant trend and the U and p values were comparable for increasing message loss.

The *Execution Phase Time* metric is not analysed here as the *Total Movement Time* is deemed to have similar enough trend and pattern, albeit over a longer duration of time considering it is the movement time of all robots. It is important to note there is a correlation in *Total Movement Time* specifically between the simulation results in Section 4.3.3, for experiments conducted with *MRComm* using the configuration of *NB*, *WLAN* and *SPLX* compared to the simulation experiments here that use the additional *SLT* network perturbation. This interrelation is expected and demonstrates that, although the experiments presented here enable the robots to estimate signal strength degradation using the *SLT* network perturbation, the robots are not equipped with the behaviour to react to communication changes. Experiments using *LF* are approximately three times the magnitude of *NB*. Moreover, as *LF* is communication aware it takes longer to recover from message loss, therefore it is negatively affected by increasing *SPL* as Table 5.4 reveals, $\mathbf{H_A}$ is accepted each time at *SPL50*.

The *Total Distance Travelled* in Table 5.8 showed insignificant changes, with *LF* experiencing minor linear decrease in U and p value. Table 5.8, for *AH* and *SLT* shows insignificant linear decrease in U and p value for *NB*, but a significant decrease in U and p value for *LF* and *SPL25* and *SPL75* is noted. For *AH* and *SSD*, *Total Distance Travelled* shows no changes for *NB*, but again a significant linear decrease in U and p value for *LF* is recorded, and the $\mathbf{H_A}$ is accepted at *SPL75*.

The *Overall Idle Time* metric in Table 5.7, for *WLAN* shows that both *NB* and *LF* experienced decreasing U and p values, with the latter being significantly affected. For *NB*, *Overall Idle Time* has insignificant changes with increasing *SPL*, therefore it is not included in Table 5.7. Whereas, for *AH* and *LF* the *Overall Idle Time* in Table 5.7 is significantly impacted by increasing *SPL*. The table reveals very low U-values and a very high effect size from the first comparison, *SPL25*.

The *Leader Disconnect Time* is *LF* specific and is shown in Table 5.5, although it shows a decreasing U and p values for most of the experiment configurations the trend is not statistically significant. The same is true for the *Follower Waiting Time*, but the metric is visually represented in Figure 5.8 and not shown in the Tables, as the results show no trend.

The *Minimum Separation* metric shows no trend for both *LF* and *NB*, hence it is omitted and is not shown in a Table, but shown visually in Figure 5.15, along with the *Maximum Separation*. The *Maximum Separation* shows a statistically significant result with decreasing U and p values in Table 5.10, specifically for the communication aware *LF*, as message loss affects it greatly.

There is one metric that is an exception to the original hypothesis analysis, and that is *Overall Near Collisions*. The results for this metric, showing a decreasing trend (i.e. near collisions decrease with the increase of message loss), are the reverse of previous performance metrics, which is unexpected (further discussed in Section 5.6). More importantly, the alternative hypothesis tested for this metric, is whether the observations from the ideal case (i.e. *SPL0*) are "greater" from the other observations (SPLX). Furthermore, the Mann-Whitney U-value should be analysed in reverse from previously, such that a large U-value strongly supports $\mathbf{H_A}$. The p-value and effect size are still read in the same way. Table 5.6 shows the U-value and the results of the new $\mathbf{H_A}$ being tested for the *Overall Near Collisions* metric. Figure 5.9 shows that for *LF* near collisions are significantly reduced as message loss is increased, and Table 5.6 reaffirms this with increasing U-value. Although the results and trend are similar for *NB*, they were not as prominent.

Figure 5.13 represents on average how many shared *tasks status messages* failed to be communicated to the *task assigner* agent. The difference between *LF* and *NB* is not only obvious, but substantial. Robots performing experiments with *LF* experience no failure in communicating *task status messages*. Contrarily, *NB* has increasing failure with increase in message loss as can be seen in Figure 5.13 and Table 5.9. Table 5.9 does not contain any experiments with *LF* as there are no communication failures. The only unique result is that for experiments with *NB WLAN*, there is no communication failure for *NB SPL0*, as there is infrastructure and no limit to communication. Unlike for experiments with *AH*, where there is communication failure for *NB SPL0* experiments. The reason that Figure 5.13 shows that some experiments have a fraction of a task failed, is because the result is the mean value from the 30 experimental runs.

The *Successful Missions* performance metric (Figure 5.14), is a percentage of successful communication based on the number of experiments, therefore it does not have a standard deviation. A successful mission is one where no communication failure has occurred. For *LF* every single experiment had successful communication, i.e. 100% mission success, whereas communication for experiments with *NB* failed almost every time. There is one except for *NB*, which is when the network type was *WLAN* and *SPL25*, Figure 5.14a, where 20% mission success was observed.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	470	0.62	0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	240	0.00	-0.57	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	121	0.00	-0.89	$\mathbf{H_A}$
AH					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	350	0.62	0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	332	0.00	-0.57	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	169	0.00	-0.89	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	329	0.04	-0.33	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	189	0.00	-0.70	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	124	0.00	-0.88	$\mathbf{H_A}$

TABLE 5.4: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for *Total Movement Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis ($\mathbf{H_0/H_A}$)**
WLAN					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	549	0.93	0.27	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	452	0.51	0.01	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	359	0.09	-0.25	$\mathbf{H_0}$
AH					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	454	0.53	0.01	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	412	0.29	-0.01	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	344	0.06	-0.29	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	417	0.32	-0.09	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	370	0.12	-0.22	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	398	0.22	-0.14	$\mathbf{H_0}$

TABLE 5.5: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for *Leader Disconnect Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
		WLAN			
{NB, SPL0, SLT}	{NB, SPL25, SLT}	662	0.00	0.57	$\mathbf{H_A}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	639	0.00	0.51	$\mathbf{H_A}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	718	0.00	0.72	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	774	0.00	0.88	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	883	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	894	0.00	1.19	$\mathbf{H_A}$
		AH			
{NB, SPL0, SLT}	{NB, SPL25, SLT}	472	0.37	0.06	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	529	0.11	0.21	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	655	0.00	0.55	$\mathbf{H_A}$
{NB, SPL0, SSD}	{NB, SPL25, SSD}	476	0.34	0.07	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	531	0.09	0.21	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	580	0.01	0.35	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	807	0.00	0.96	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	877	0.00	1.15	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	866	0.00	1.12	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	624	0.00	0.47	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	760	0.00	0.84	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	801	0.00	0.95	$\mathbf{H_A}$

TABLE 5.6: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for the *Overall Near Collision*, testing the alternative hypothesis that observations from the ideal case (*SPL0*) are **greater** than those of other *SPLX* observations.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
		WLAN			
{NB, SPL0, SLT}	{NB, SPL25, SLT}	510	0.81	0.16	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	401	0.24	-0.13	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	340	0.05	-0.30	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	356	0.08	-0.25	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	182	0.00	-0.72	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	136	0.00	-0.85	$\mathbf{H_A}$
		AH			
{LF, SPL0, SLT}	{LF, SPL25, SLT}	320	0.03	-0.35	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	270	0.00	-0.49	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	187	0.00	-0.71	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	197	0.00	-0.68	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	49	0.00	-1.08	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	29	0.00	-1.14	$\mathbf{H_A}$

TABLE 5.7: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for *Overall Idle Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	520	0.85	0.19	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	465	0.59	0.04	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	447	0.49	0.00	$\mathbf{H_0}$
AH					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	483	0.69	0.09	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	432	0.40	-0.05	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	385	0.17	-0.18	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	306	0.02	-0.39	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	432	0.40	-0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	323	0.03	-0.34	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	457	0.54	0.02	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	405	0.26	-0.12	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	320	0.03	-0.35	$\mathbf{H_A}$

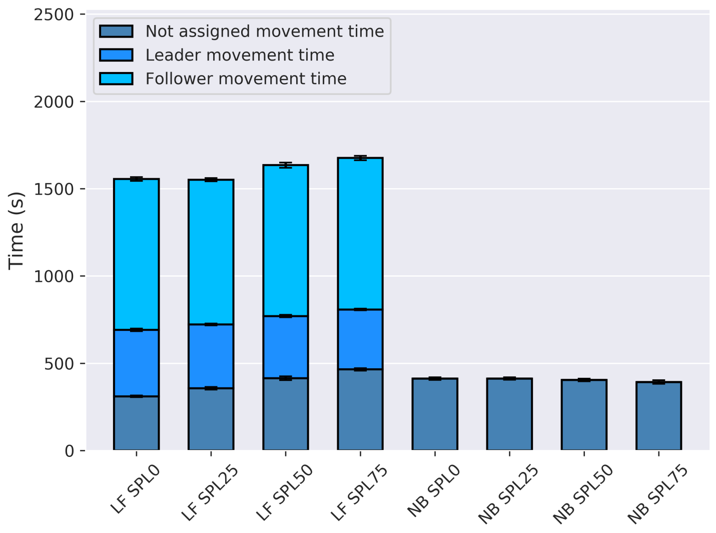
TABLE 5.8: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for *Total Distance Travelled*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis ($\mathbf{H_0/H_A}$)**
WLAN					
{WLAN, NB, SPL0, SLT}	{WLAN, NB, SPL25, SLT}	90	0.00	-0.97	$\mathbf{H_A}$
{WLAN, NB, SPL0, SLT}	{WLAN, NB, SPL50, SLT}	0	0.00	-1.21	$\mathbf{H_A}$
{WLAN, NB, SPL0, SLT}	{WLAN, NB, SPL75, SLT}	0	0.00	-1.21	$\mathbf{H_A}$
AH					
{AH, NB, SPL0, SLT}	{AH, NB, SPL25, SLT}	300	0.00	-0.40	$\mathbf{H_A}$
{AH, NB, SPL0, SLT}	{AH, NB, SPL50, SLT}	105	0.00	-0.93	$\mathbf{H_A}$
{AH, NB, SPL0, SLT}	{AH, NB, SPL75, SLT}	45	0.00	-1.09	$\mathbf{H_A}$
{AH, NB, SPL0, SSD}	{AH, NB, SPL25, SSD}	187	0.00	-0.71	$\mathbf{H_A}$
{AH, NB, SPL0, SSD}	{AH, NB, SPL50, SSD}	112	0.00	-0.91	$\mathbf{H_A}$
{AH, NB, SPL0, SSD}	{AH, NB, SPL75, SSD}	95	0.00	-0.96	$\mathbf{H_A}$

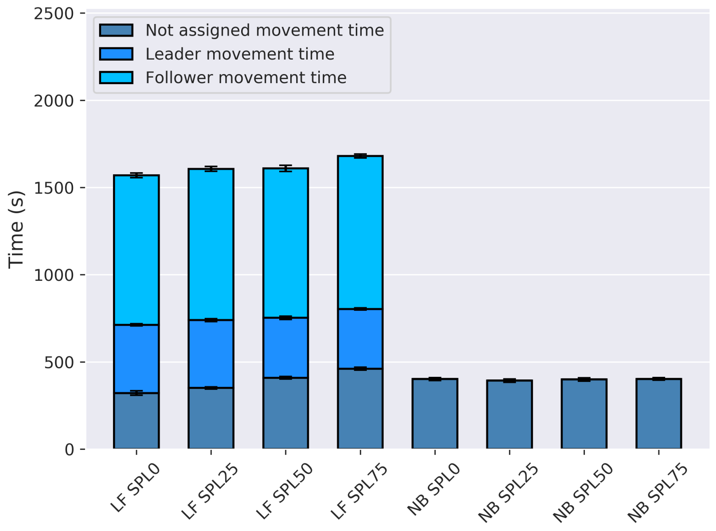
TABLE 5.9: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for the *Average Failed Tasks*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
		WLAN			
{LF, SPL0, SLT}	{LF, SPL25, SLT}	170	0.00	-0.76	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	64	0.00	-1.04	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	82	0.00	-0.99	$\mathbf{H_A}$
		AH			
{NB, SPL0, SSD}	{NB, SPL25, SSD}	387	0.18	- 0.17	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	374	0.13	-0.21	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	326	0.03	-0.33	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	132	0.00	-0.86	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	92	0.00	-0.97	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	23	0.00	-1.15	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	271	0.00	-0.48	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	45	0.00	-1.09	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	55	0.00	-1.07	$\mathbf{H_A}$

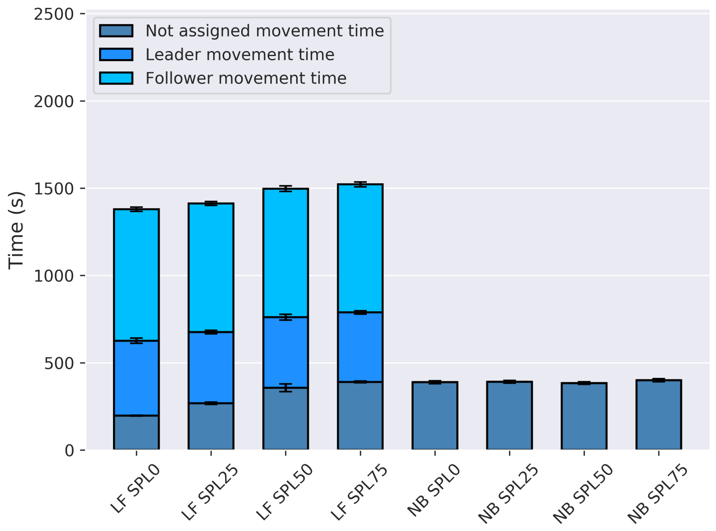
TABLE 5.10: U-value, p-value, effect size and hypothesis outcome of *MRComm* simulated robot experiments for *Maximum Separation*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.



(a) Total Movement Time for WLAN with SLT

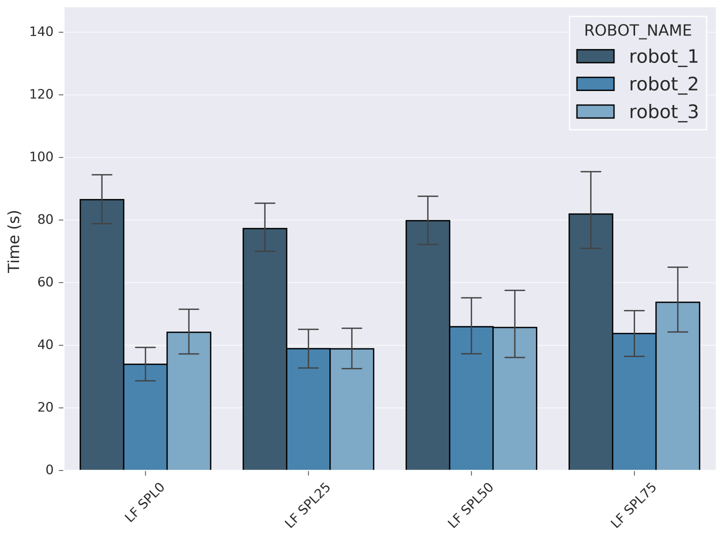


(b) Total Movement Time for AH with SLT

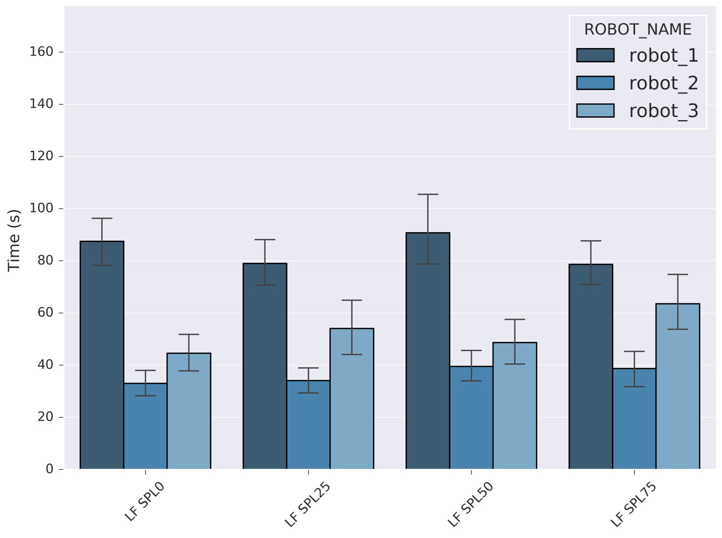


(c) Total Movement Time for AH with SSD

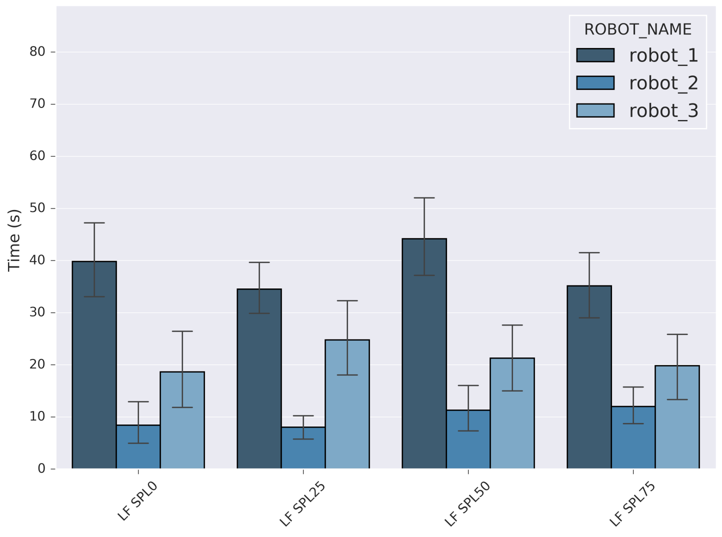
FIGURE 5.6: Total Movement Time stacked for all experiment configurations.



(a) Leader Disconnect Time for WLAN with SLT

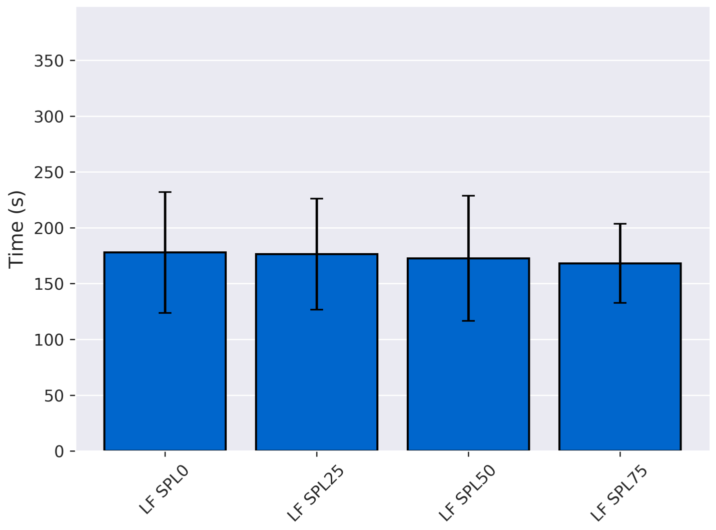


(b) Leader Disconnect Time for AH with SLT

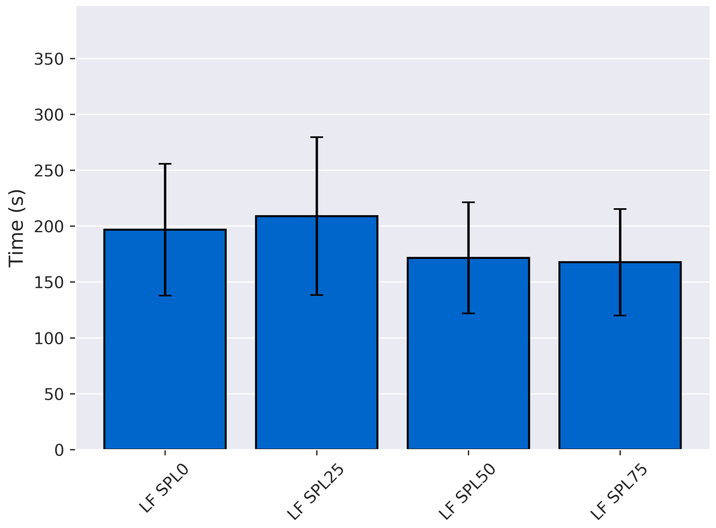


(c) Leader Disconnect Time for AH with SSD

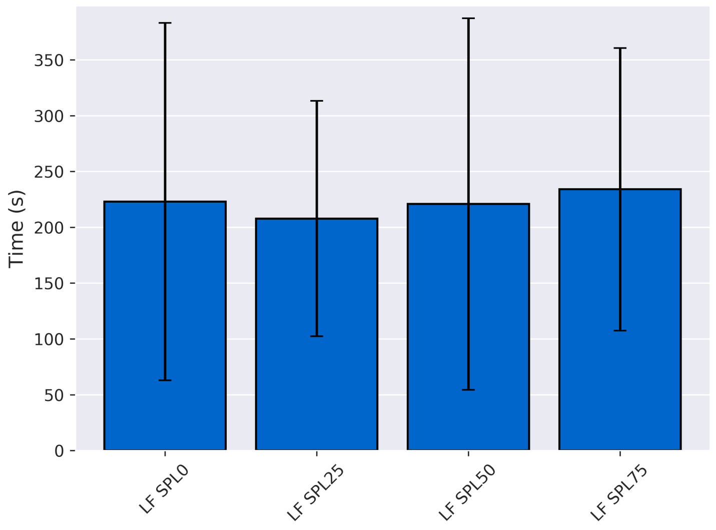
FIGURE 5.7: Leader Disconnect Time per robot for all experiment configurations.



(a) Follower Waiting Time for WLAN with SLT

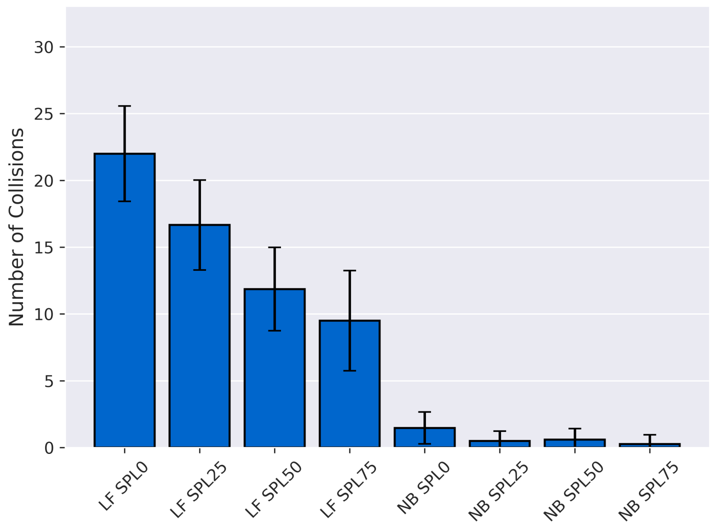


(b) Follower Waiting Time for AH with SLT

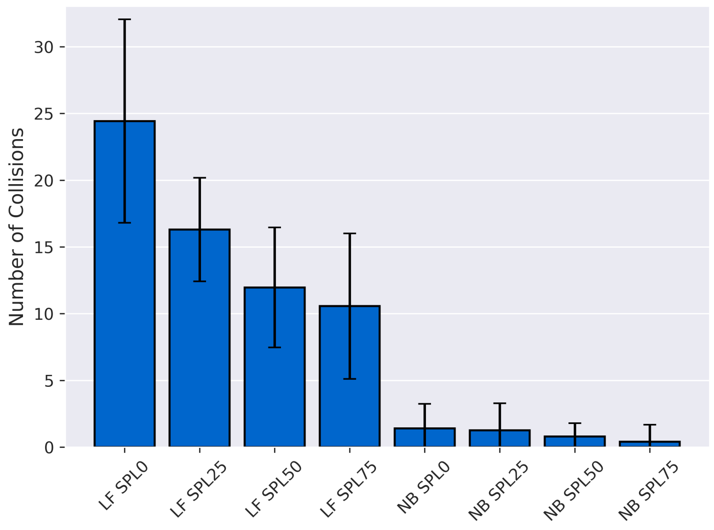


(c) Follower Waiting Time for AH with SSD

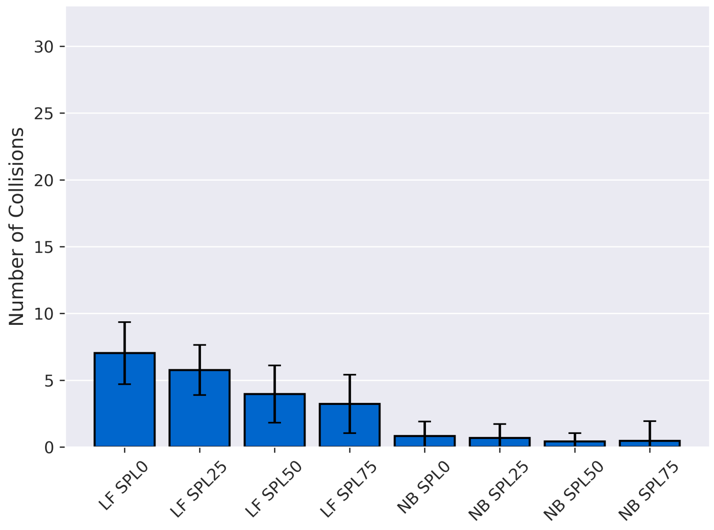
FIGURE 5.8: Follower Waiting Time for all experiment configurations.



(a) Overall Near Collisions for WLAN with SLT

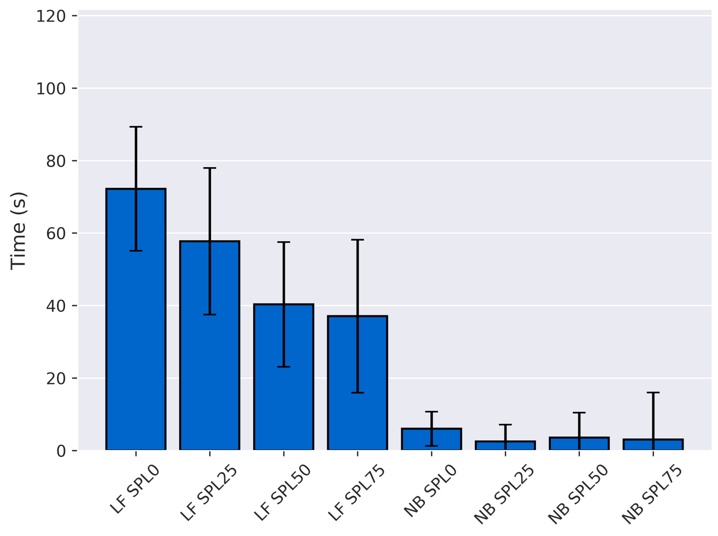


(b) Overall Near Collisions for AH with SLT

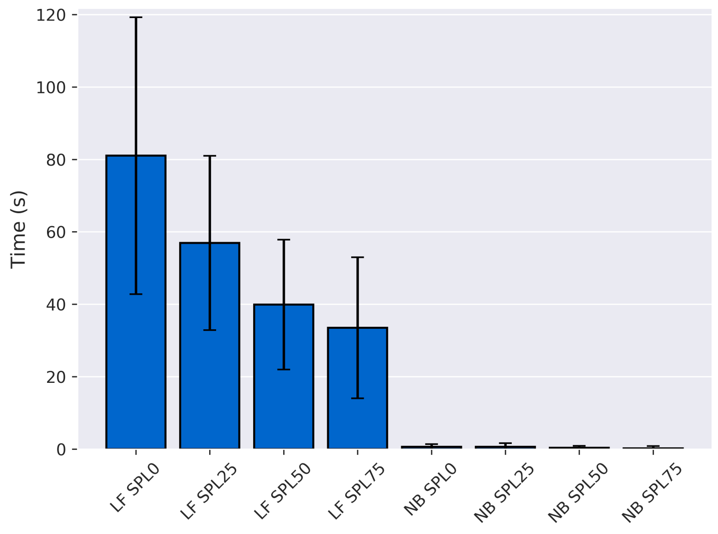


(c) Overall Near Collisions for AH with SSD

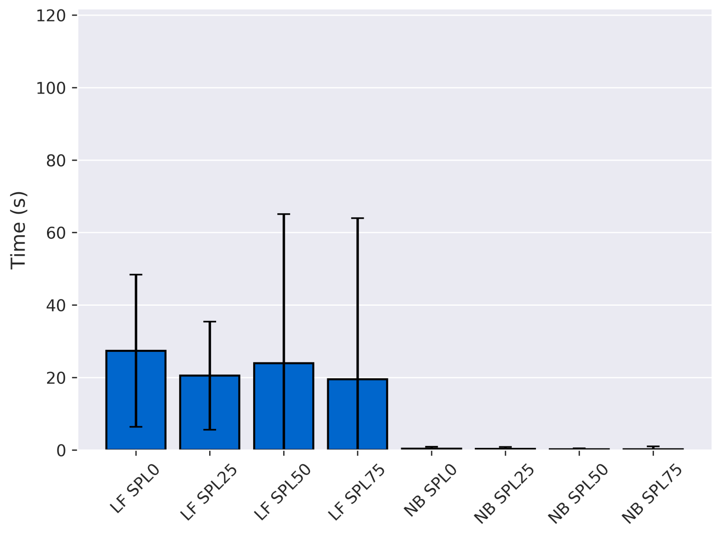
FIGURE 5.9: Overall Near Collision for all experiment configurations.



(a) Overall Delay Time for WLAN with SLT

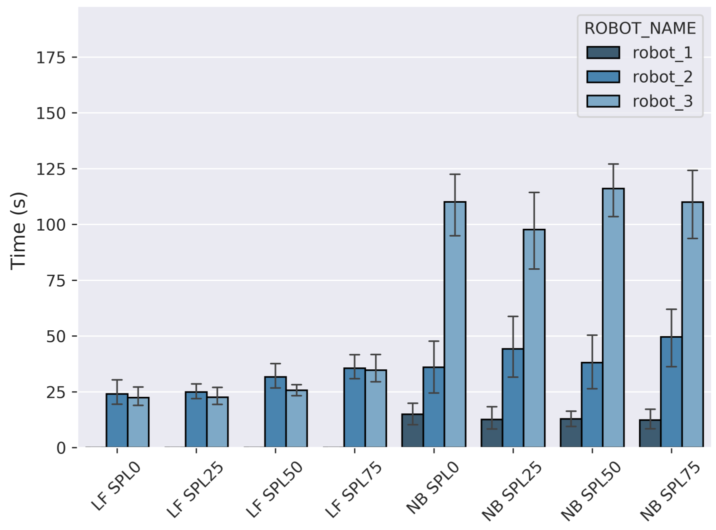


(b) Overall Delay Time for AH with SLT

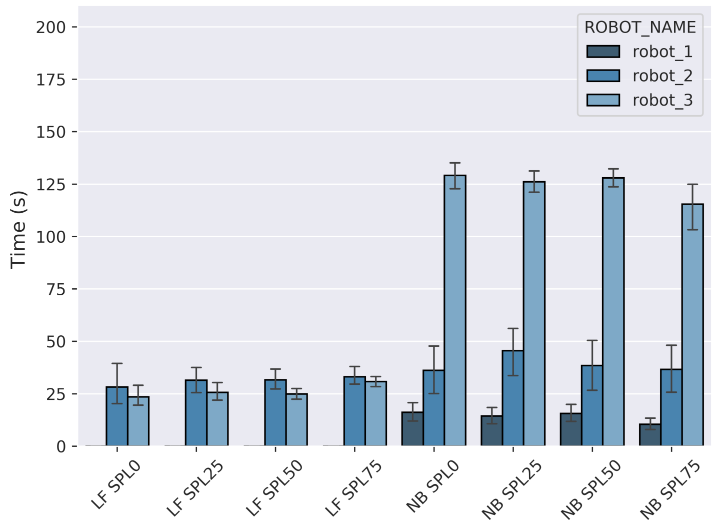


(c) Overall Delay Time for AH with SSD

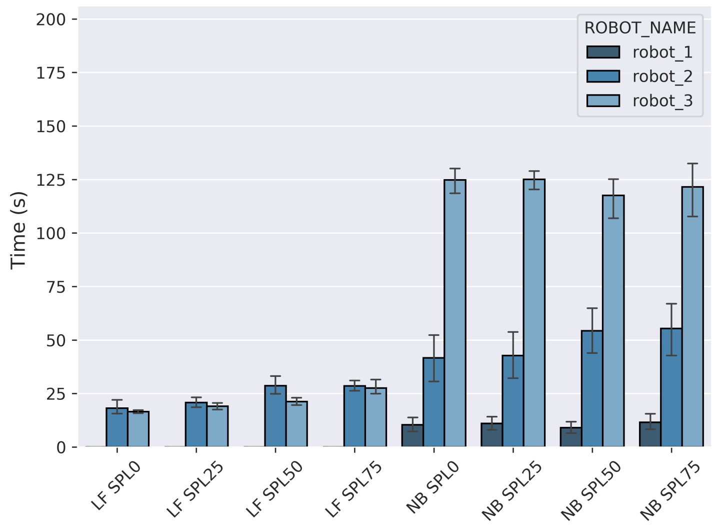
FIGURE 5.10: Overall Delay Time for all experiment configurations.



(a) Overall Idle Time for WLAN with SLT

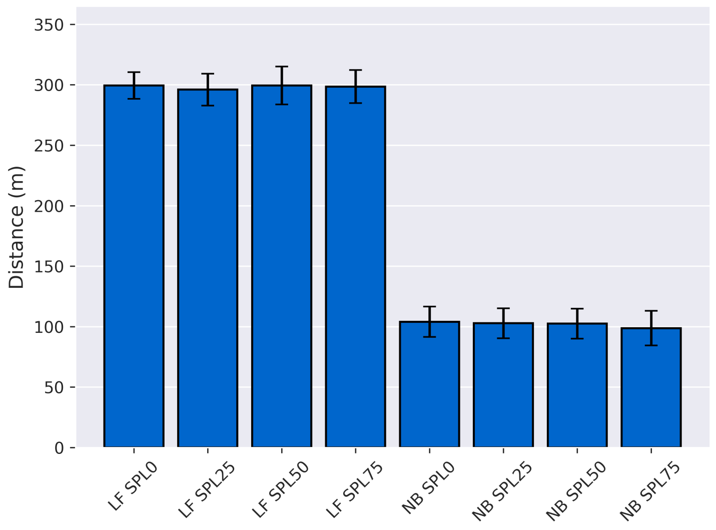


(b) Overall Idle Time for AH with SLT

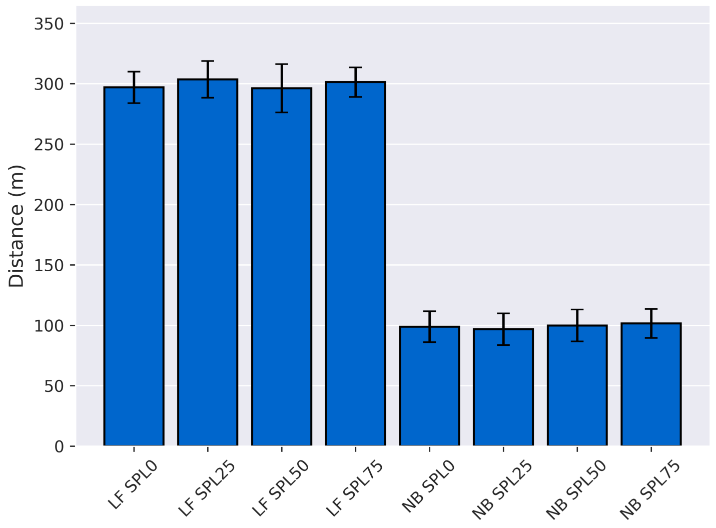


(c) Overall Idle Time for AH with SSD

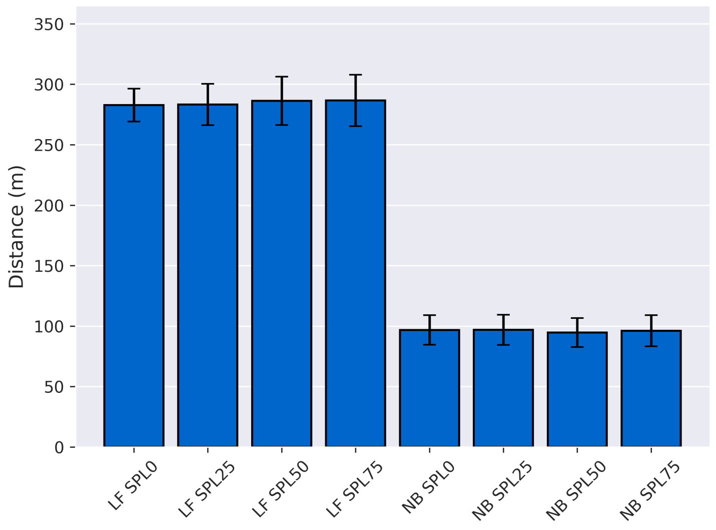
FIGURE 5.11: Overall Idle Time per robot for all experiment configurations.



(a) Total Distance Travelled for WLAN with SLT

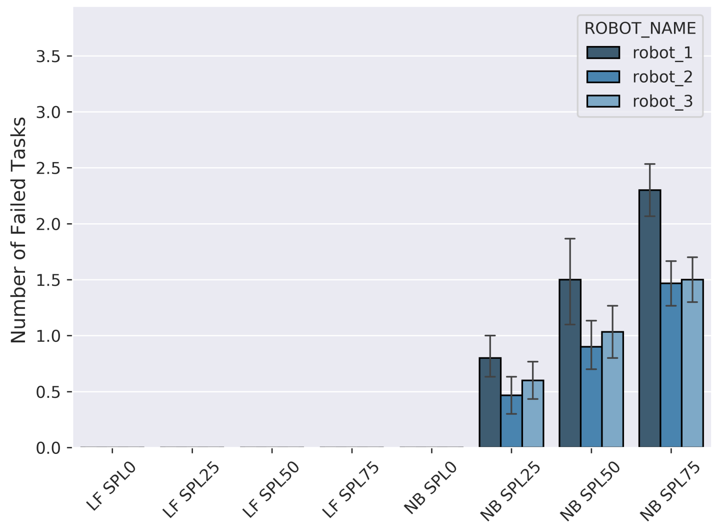


(b) Total Distance Travelled for AH with SLT

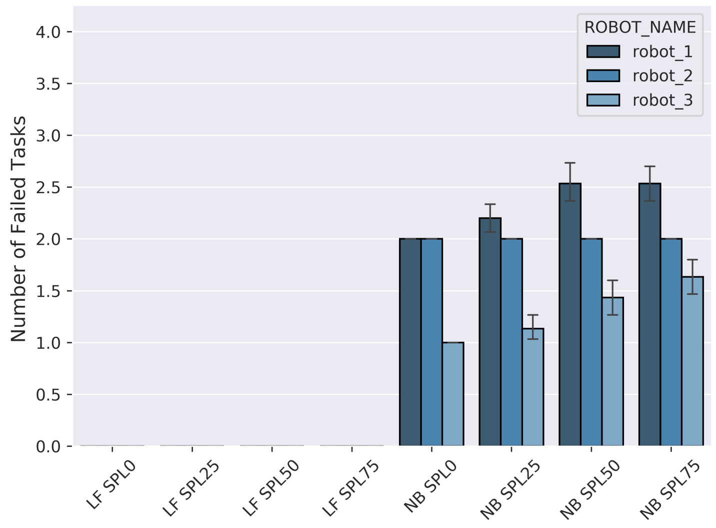


(c) Total Distance Travelled for AH with SSD

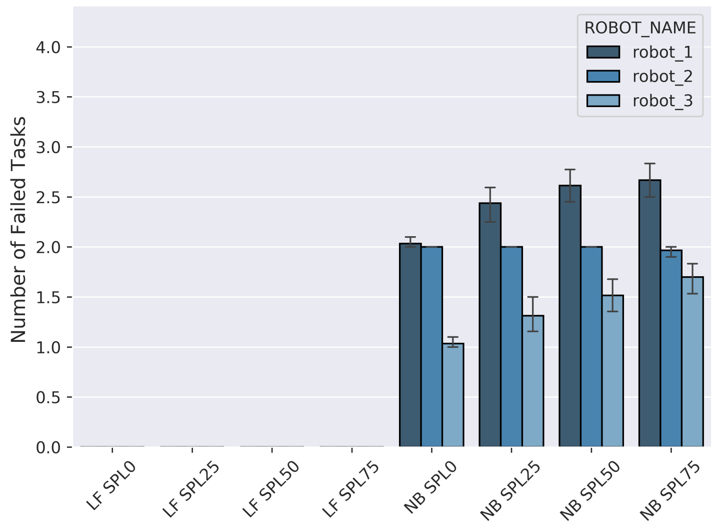
FIGURE 5.12: Total Distance Travelled for all experiment configurations.



(a) Average Failed Tasks for WLAN with SLT

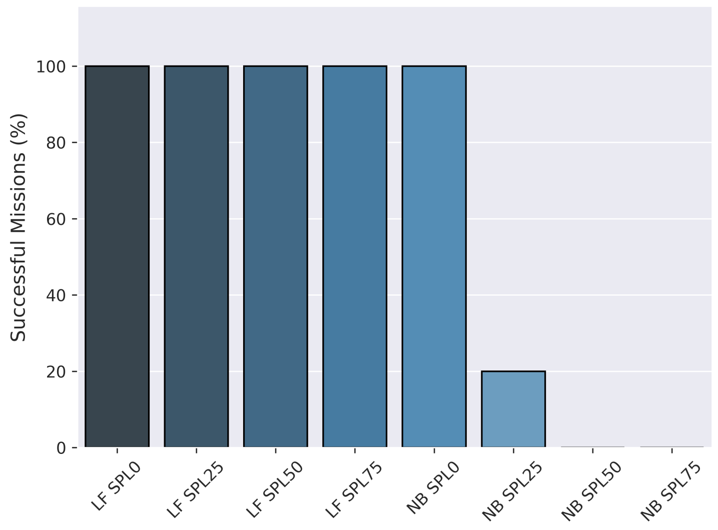


(b) Average Failed Tasks for AH with SLT

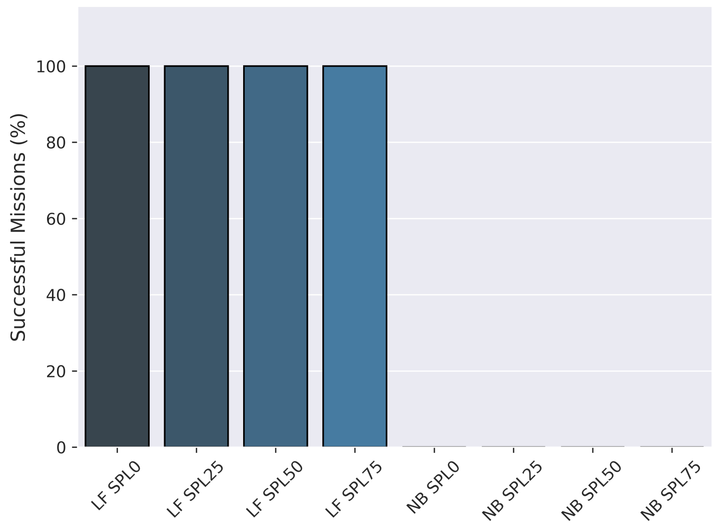


(c) Average Failed Tasks for AH with SSD

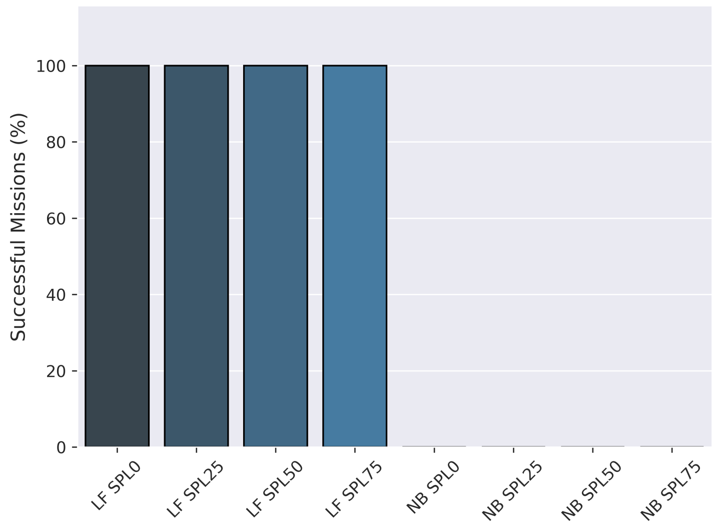
FIGURE 5.13: Average Failed Tasks per robot for all experiment configurations.



(a) Successful Communication in Missions for WLAN with SLT

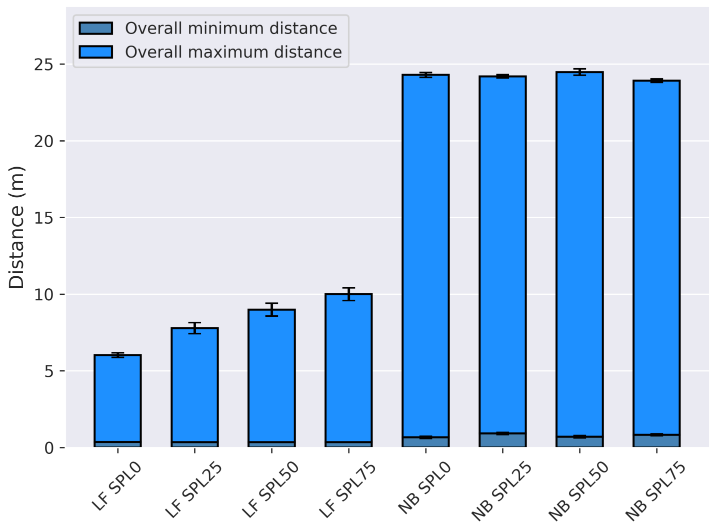


(b) Successful Communication in Missions for AH with SLT

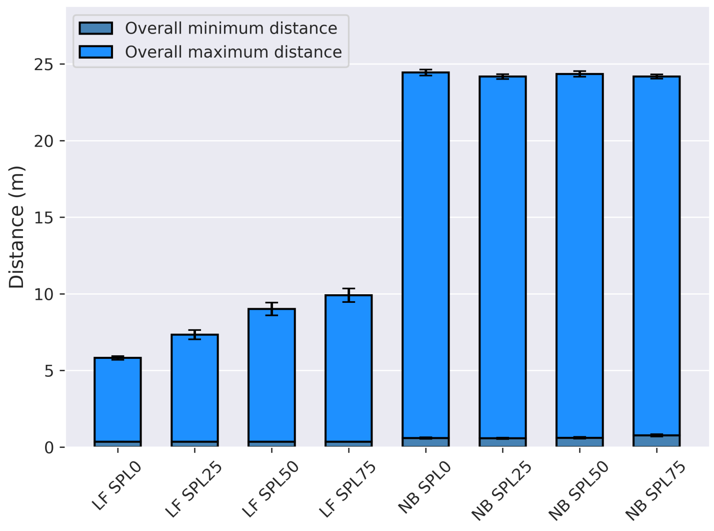


(c) Successful Communication in Missions for AH with SSD

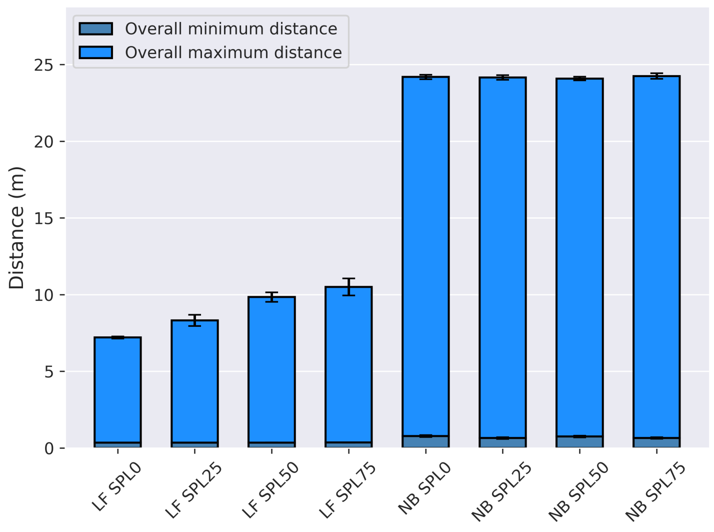
FIGURE 5.14: Percentage of Successful (Communication) Missions for all experiment configurations.



(a) Minimum and Maximum separation for WLAN with SLT



(b) Minimum and Maximum separation for AH with SLT



(c) Minimum and Maximum separation for AH with SSD

FIGURE 5.15: Minimum and Maximum separation for all experiment configurations.

5.6 Discussion

The results reveal that the more complex and communication aware behaviour, *LF*, with suboptimal parameters is capable of achieving perfect mission success rate. Unfortunately, there are negative performance effects to using *LF*, for instance *Total Distance Travelled* and *Total Movement Time* have on estimate, a three-fold increase over that of *NB*. However, due to the complexity of *LF* and the addition of roles and actions (i.e.*Leader Disconnect Time*, *Follower Waiting Time*, etc.), the *Total Movement Time* and *Total Distance Travelled* results show the magnitude that other roles add to those metrics by having communication awareness and maintenance. Moreover, the statistical analysis Tables in the Results section, acknowledge that *LF* is more likely to be significantly affected by increasing message loss than *NB*. This is expected from *NB* as it does not react to changes in communication quality. As aforementioned, suboptimal parameters are used for *LF*, therefore some performance metrics can be improved in future work. Finally, Figure 5.16 is used to show the hierarchical and complexity difference between the *NB* FSM, that is encapsulated by a black box with dashed lines, and the *LF* FSM, that is encapsulated by a red box with dashed lines. The smaller green and blue boxes with dashed lines are themselves FSMs of the Leader and Follower roles. Figure 5.16, is an abstraction of the FSMs from Chapter 3, Sections 3.7.1 and 3.7.2.

One of the main reasons that *LF* is capable of vastly improving and maintaining communication over *NB*, is not only because it can detect signal strength or mitigate message loss, but because it forces the MRT to form a close proximity group. This is observed by the results in Figure 5.15, which shows the minimum and maximum distance between the agents at any given point in time during a mission. *NB* has over double the *Maximum Separation* compared to *LF*, and increasing message loss does not cause a significant increase/decrease for this metric. *LF* is warned of an impending

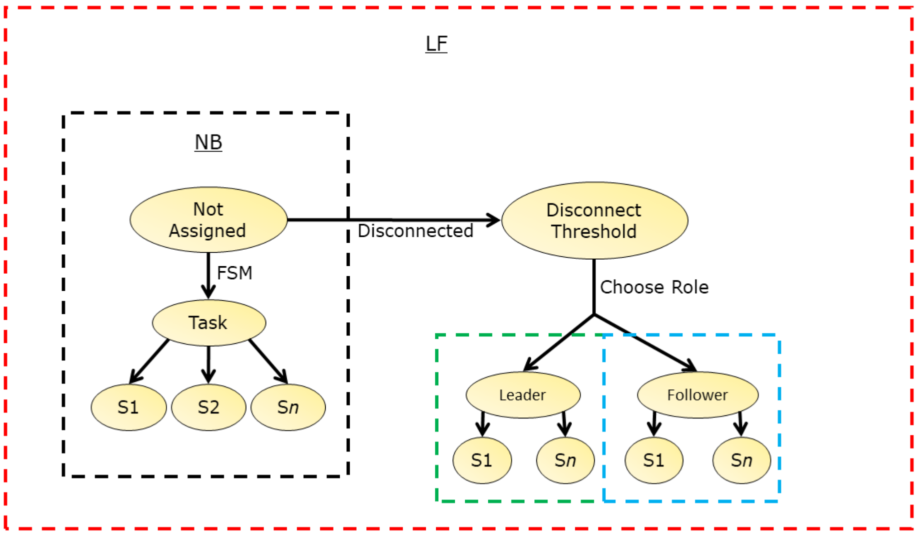


FIGURE 5.16: An abstraction of *LF*, showing how it encapsulates *NB* within the FSM hierarchy

disconnect at a distance of ≈ 4.0 m (*SLT*) or at a distance that causes signal strength of -42 dBm (*SSD*) or less. Contrarily, the results in Figure 5.15 show a maximum separation of just over ≈ 10.0 m for *LF* and *SPL75*. This metric demonstrates an unexpected but excellent result that the *LF* behaviour displays, which is referred to in MAS and MRS as emergent behaviour. For example, if a robot (using *LF*) drifts past the *warning* threshold limit of either *SLT* or *SSD* then, even if it stops communicating with the other agents (i.e. out of *AH* communication range), it has stored the last known position of either the leader or not assigned robot(s). Thereafter, the robot that has separated will use the most current information of the last known position of the leader or not assigned robot to get back to that location, while the other robots will have paused to wait for the separated robot to reconnect. Therefore, even if the *warning* and *limit* thresholds are passed (i.e. the large maximum separation observed in Figure 5.15) a robot team using *LF* will not cause disruption to communication or to the success of the mission. In contrast, the separation distance for *NB* is well over ≈ 20.0 m, approaching the maximum measured distance between the furthest set of tasks in the operational environment.

Although *NB* is not generally affected by message loss for some of the core performance metrics, the extended communication specific metrics show significant impact. Both *NB* and *LF* show similar trends for *Overall Near Collisions* and *NB* shows a significantly increasing trend of *Average Failed Tasks*. The overarching goal of the experiments presented here is to demonstrate that continuous (successful) communication is achievable when using the novel *LF* behaviour. However, there is a cost cost to improving communication performance , such as the negative impact to other performance metrics.

5.7 Summary

The experiments conducted here expand the *MRComm* framework to include all the network parameters for the simulated environment and additionally to prepare the framework toward the truly distributed experiment design in Chapter 6.

I expand the performance metrics to include communication and behaviour specific metrics. The experiment design is updated to include a second network perturbation, which is a signal strength threshold. Two threshold network perturbations are introduced to enable the MRT to predict and emulate signal strength degradation, specifically to analyse the *AH* network type and present how it impacts communication. A simple distance-based network threshold is introduced as a foundation, denoted *SLT*, which can be used for any experiment configuration. A more complex and novel network threshold that predicts signal strength (denoted *SSD*) is used, which bases its predictions on SVR models generated by indoor signal strength maps.

The novel dynamic *LF* behaviour is presented in this chapter and, as demonstrated by the results, it achieves perfect communication with the designed set of network perturbations, i.e. increasing message loss and reacting to degrading signal strength. There are some caveats to keep in mind.

The *LF* behaviour achieves perfect communication performance at the cost of reduced performance for some of the other collected metrics, the most impacted of which are the *Total Movement Time* and *Total Distance Travelled*. The baseline *NB* behaviour, which enables robots to use standard navigation and collision avoidance, shows poor communication results in comparison. However, as expected from the simple implementation of *NB*, the *Total Movement Time* and *Total Distance Travelled* are less impacted, at the cost of very poor or no communication at all.

Chapter 6

MRComm: Physical Experiments

6.1 Introduction

The contributions presented here are as follows. The experiments conducted in the physical environment on a truly distributed MRT, defined in Chapter 2 Section 2.3.4, which builds on from the investigation in Chapter 5. The introduction of a new network perturbation based on actual signal strength from robots is introduced and tested in this chapter, which is denoted *Effective Signal Degradation* (*ESD*). The proof-of-concept that *MRComm* can be adapted to physical robots using the novel *LF* behaviour, which enables the robots to self-maintain an *AH* communication while performing their function, and achieve 100% communication even in adverse conditions.

The physical experiments are carried out using the same scenario as in Chapter 5 (S_1). However, unlike the simulated experiments, the true Wi-Fi based *WLAN* and *AH* network type are used for communication in *MRComm*, which is made possible using *FKIE* the extension introduced to the communication capabilities in Section 3.2.1. The network type limitations mentioned in Section 3.8.1 are applicable in this chapter.

An overarching aim of the simulated experiments and the physical experiments, conducted here, is to investigate two hypotheses. The initial hypothesis states that adverse network conditions do impact the performance or abilities of a physical MRT, regardless of the behaviour used. The second hypothesis states that adverse network conditions do not impact the performance or abilities of a physical MRT, regardless of the behaviour used. Although the results here have minor differences to those in Chapter 5 and are not as statistically strong, the overall conclusion remains unchanged. Therein, either of the hypotheses are supported, depending on which individual metric is being investigated. However, combining the analyses of the different metrics shows that the first hypothesis more accurately depicts the overall findings. Moreover, to prove the soundness of the experiments conducted with the synthetically designed *SSD*, I compare the *SSD* and *ESD* perturbations, in Section 6.3, to show that the signal strength degradation of both adhere to a similar trend.

The rest of this chapter is structured as follows. Section 6.2 introduces the *ESD* network perturbation that is used in the physical experiments. Section 6.3 compares the signal strength results between the new network perturbation (*ESD*) and the *SSD* perturbation. Section 6.4 outlines the methodology and how the physical experiments are setup and executed, and defines the *warning* thresholds of the network perturbations. Section 6.5 presents the results, which demonstrate successful communication and mitigation of common network issues (perturbations), whilst using the novel *LF* behaviour and *AON* message function. Section 6.6 discusses the results and issues raised in the context of the physical experiments. Moreover, the physical experiment results are compared to the corresponding simulation experiments presented in Chapter 5, to better understand environmental discrepancy. Finally, Section 6.7 brings the chapter to a close with a brief conclusion and summary.

6.2 Effective Signal Degradation (ESD)

ESD is the network perturbation implemented for physical experiments in the *MRComm* framework. The *ESD* perturbation uses a software module, which is referred to as *ESD Pulse*. *ESD Pulse* is executed simultaneously with the instantiation of a physical (robot) *task execution* agent. Once the *ESD Pulse* has been initiated on a robot, it will continually keep polling the resulting signal strength from each other robot connected to the local *AH* network at a rate of 2 Hz. However, the *task execution* agent will only request the signal strength periodically or when it is required. An internal call to the *MRComm* system executes the *ESD Pulse* process, but the *ESD Pulse* is managed externally and concurrently with other system processes. Executing the *ESD Pulse* concurrently with the robot agents and ROS packages provides three main benefits. Firstly, if a robot experiences hardware malfunction and loses most sensor functionality, assuming the network device is still functional, it will continue pulsing a signal. Secondly, if a robot experiences a software crash caused by an internal issue as a result of the *MRComm* framework, or external issue as a result of ROS, it will continue pulsing a signal. The above mentioned benefits are a boon for multi-robot exploration and task navigation where the goal is to identify dangerous locations, making it possible for example to relocate broken down robots at the end of a mission and flag dangerous locations. Finally, the *ESD Pulse* can run on almost any type of device, even without the ROS middleware, which would enable said device to send and receive signal strength updates to any robot, thus increasing its utility. Figure 6.1 illustrates the sysem design of communication of shared messages between robot agents when using the *ESD Pulse* module. The *ESD warning* threshold is defined in Section 6.4.2.

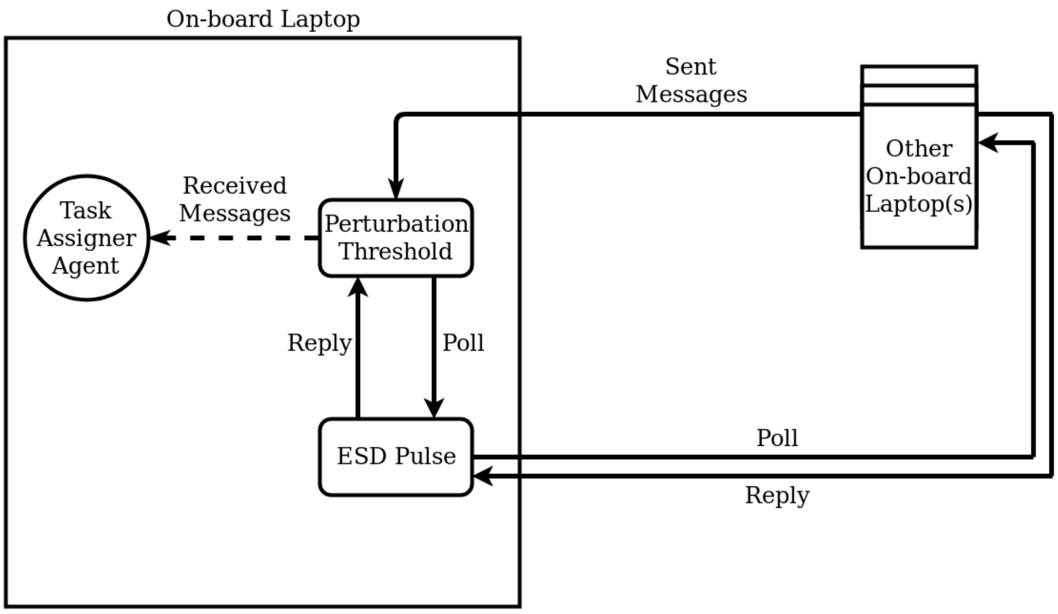


FIGURE 6.1: Communication of shared messages between a receiving on-board laptop (robot) and transmitting on-board laptop, using the *ESD* network perturbation.

6.3 ESD versus SSD

The two network perturbations exhibit a similar trend and overlapping standard deviations across both simulated (*SSD*) and physical (*ESD*) signal strength results, as illustrated in Figure 6.2. Figure 6.2 plots the mean from a sequence of samples of signal strength, received from ROBOT_2 as observed by ROBOT_1's point of view. The samples of data are taken from twenty experimental runs and an initial time range t measured in seconds from the start of an experiment at $t_0 = 0$ up to $t \approx 150$ s. The conditions for the chosen samples were carefully selected, such as using experiments with the *NB* behaviour only, as there is less variance in path routing, and a specified time slice t, which helps to form more consistent signal strength results.

Figure 6.2 shows that both perturbations have a similar trend, however they do not yield similar signal strength results at corresponding distances. To confirm if the two sample distributions are in fact from the same distribution or from separate distribution, the non-parametric Mann-Whitney U test is used. The Mann-Whitney U tests for the ("two-sided" hypothesis) null hypothesis, that the two samples are from the same population distribution, and the alternative hypothesis, that samples are from two separate population distributions. As described in Section 5.3, the *SSD* perturbation is modelled using two separate SVRs, which can predict a limited range of signal strength values, i.e. after a certain distance the simulated signal strength hits a maximum threshold (-60 dBm) where the SVRs can no longer represent accurate predictions of the signal strength. The above limit is represented by a dashed horizontal red line on Figure 6.2, which reveals that *SSD* reaches the limit near the 110 second mark. Therefore, the data sample used for *SSD* is based on the signal strength results only up to the first value that hits the threshold limit of -60 dBm. The U-test score for *SSD*, $n = 110$, versus *ESD*, $n = 151$, is $U = 4937$, $p = 1.22 \times 10^{-8}$ and the $U_{max} = 16,1610$ (i.e. very large). Although the U-value may seem high, the p-value is essentially zero and U_{max} supports this result. Moreover, this indicates that the alternative hypothesis is accepted here (i.e. the two independent samples come from significantly different sample distributions). This means *SSD* and *ESD* are statistically significantly different, however they observe very similar trends and the standard deviation overlaps in most cases. Incidentally, as the signal strength decreases (the robots move further away from each other), the standard deviation increases by approximately double the amount after 25 seconds and a large overlap occurs. As mentioned in [102], it is a common occurrence that signal strength becomes more unreliable as distance increases. Overall the results exhibit a satisfactory outcome, as *SSD* could easily be tweaked to more closely resemble *ESD*, this is briefly discussed in Section 6.7. Further statistical data are listed in Table 6.1.

The initial results portray that *SSD*, which predicts signal strength, can achieve a similar behaviour to the actual signal strength *ESD* in the physical environment, without having to adopt complex machine learning methods or other complex forms of representing signal strength. Moreover, this means that using *SSD* in a simulated environment can give a plausible comparison of how signal strength degradation can affect an MRT. For example, if there is limited access to the physical environment and physical experiments cannot be conducted.

	N	Time Range *(min, max)*	Signal Strength *mean (stdev)*
ESD	151	(0.0, 150.0)	-35.00 (2.8)
SSD	110	(0.05, 110.03)	-35.00 (2.8)

TABLE 6.1: Statistics comparing *ESD* and *SSD* data samples.

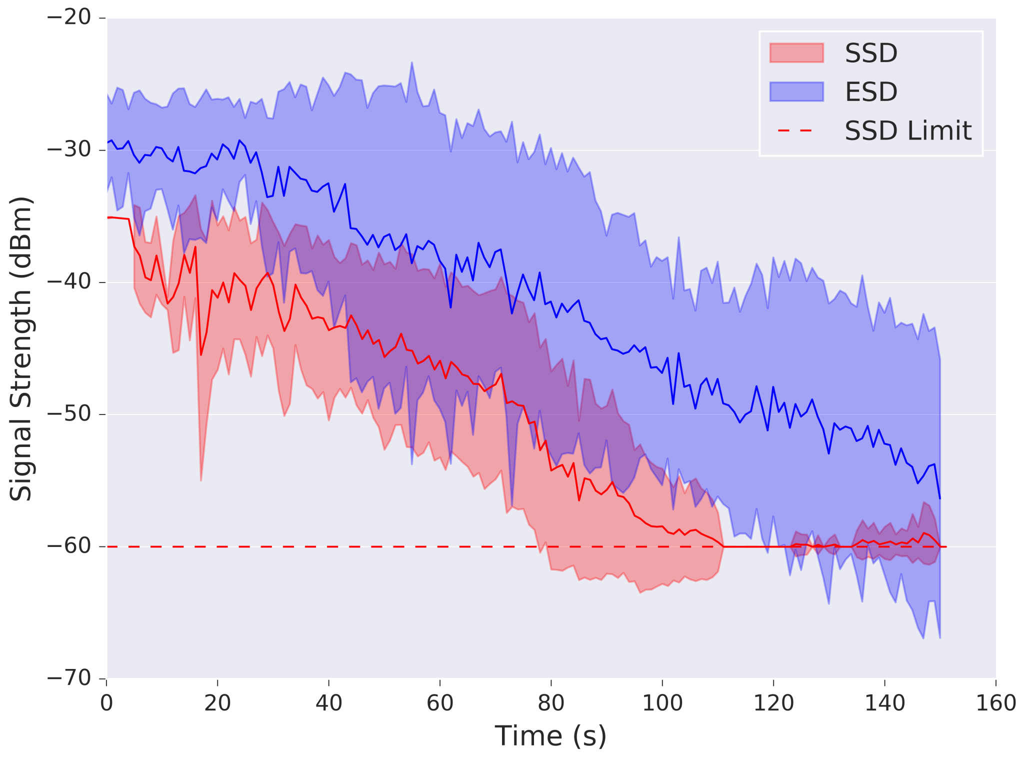


FIGURE 6.2: Samples of signal strength received from ROBOT_2 by ROBOT_1, when using the *ESD* and *SSD* network perturbation. The measurement is taken over the initial 150 seconds of physical MRT experiments.

6.4 Experiments

This section is comprised of three parts. The first part is the Methodology 6.4.1, covering the hardware, experiment parameters and how experiments will be conducted. The second part covers the Network Perturbations 6.4.2, which specifically defines the *warning* thresholds of *SSD* and *ESD*. The final part briefly describes the Experiment Setup 6.4.3, which is similar to the setup in Chapter 5.

6.4.1 Methodology

The robot platform remains the same as before (Section 3.5.1), which comprises of an on-board laptop (the Acer Travelmate B117 running Ubuntu 14.04), the Turtlebot 2 base and an RGB-depth camera (the Asus Xtion Pro). The main difference is that many more communication specific functionalities are used in the experiments conducted here, described below.

The *AH* network type requires the creation of peer-to-peer communication and once connection is established it is self-maintained. This essentially requires either an extra robot to host the *task assigner* agent or that one robot in the team needs to host both a *task assigner* and *task execution* agent. There are no immediate benefits of implementing the initial case and it is not ideal as it will require a redefinition of the *task assigner* and further architectural changes within the *MRComm* framework. Therefore, the second option is used, where one robot hosts a *task execution* and *task assigner* agent.

The method for performing physical experiments in this chapter is different compared to the method used for the baseline physical experiments presented in Chapter 4. The design methodology for the experiments in this chapter is as follows. Each on-board laptop (robot) that is on the Turtlebot2 base is supplied with the same version of the *MRComm* framework. At the start of an experiment, a *task execution* agent is instantiated on each robot. Moreover, the *task assigner* agent is required to initialise the mission, assign tasks and record received messages from the *task execution* agents. However, the distributed nature of the multi-robot system and *AH* network requirements demand that the *task assigner* agent operates on **one** of the on-board laptops concurrently with a *task execution* agent. As such, for physical robot experiments, each *task execution* agent records their own performance metrics. This is part of the truly distributed design mentality described in Section 2.3.5. For the experiments conducted here, ROBOT_2 is chosen to host both a *task execution* agent and a *task assigner* agent. Once the *task assigner* agent is initialised it identifies the robots in the team and assigns them their tasks. Thereafter, the robots begin to navigate towards their assigned tasks. Each task definition includes a location where the robot performs actions, such as *sensor-sweep* (e.g. collecting a series of images). At the end of an experiment, the *task assigner* agent records the shared messages and experiment messages, and concludes by determining which robot(s) failed to communicate *task status messages* .

The network perturbations impact the same shared message topics as in the previous chapter, *pose messages* and *task status messages*. The network perturbations evaluated in the physical experiments are:

- Simulated Packet-Loss (*SPLX* where $X \in \{0, 25, 50, 75\}$) 3.8.3.
- Simulated Loss Threshold (*SLT*) 5.2.
- Simulated Signal Degradation (*SSD*) 5.3.
- Effective Signal Degradation (*ESD*) 6.2.

6.4.2 Network Perturbations and Thresholds

Network perturbations influence the communication medium and act as constraints on the multi-robot system to allow communication quality issues to be tenable in certain cases where either the *WLAN* or *AH* networks are used. The *warning* thresholds impact only *LF* and not *NB*, as the latter behaviour lacks the awareness to react to these communication warnings.

The *SLT* perturbation remains with the same limit of 4.0 m, as defined in Section 5.2. The *warning* threshold and its effect on the *LF* behaviour is displayed in Table 6.2.

The *SSD warning* threshold is based on the result from using the same two separately trained SVR models, as described in Section 5.3. However, it has been adjusted for the physical experiments presented in this chapter, therefore the *warning* threshold limit is set to -41 dBm. Testing the physical experiments with *LF* revealed that detecting the threshold sooner was more beneficial for the MRT and resembled more closely the behaviour observed in Chapter 5's corresponding simulated experiments. The *warning* threshold and its effects on the *LF* behaviour are shown in Table 6.2. It can be seen in Table 6.2 that line-of-sight distance d will signal *warning* threshold at two different estimated distances, between 2.2-3.9 m and greater than 5.7 m.

ESD is the network perturbation introduced in this chapter and it uses the *ESD Pulse*, an external process that is always running in the background, to retrieve and update the stored signal strength for each other robot in the MRT. The *ESD warning* threshold is set to -41 dBm, the same as for *SSD*, and its effects on the *LF* behaviour are shown in Table 6.2. As for the *SSD* case, it too has two different estimated distance triggers for the *warning* threshold in line-of-sight, which are between 3.5-4.2 m and greater than 8.0 m.

Network Perturbation Warning Thresholds			
Network Perturbation	Warning Threshold	**LF Distance Alert (approx.)**	
		Line-of-sight signal	Obstructed signal
SLT	4.0 m	$d > 4.0$ m	$d > 4.0$ m
SSD	-41 dBm	2.2 m $< d <$ 3.9 m or $d >$ 5.7 m	$d > 1.6$ m
ESD	-41 dBm	3.5 m $< d <$ 4.2 m or $d >$ 8.0 m	$d > 2.0$ m

TABLE 6.2: The *warning* thresholds are demonstrated and the approximate distance is shown when the system is warned of disconnecting (LF behaviour only).

6.4.3 Experiment Setup

Scenario S_1 is used for the physical experiments. Each robot needs to be manually setup before each mission, and on rare occasion the navigation system unexpectedly, partially or completely, fails during this process. In the case of a partial failure, inaccurate results may be recorded by the affected robot, but generally a mission is completed. In the case of complete failure, an affected robot may experience either a time-out (i.e. waiting too long for an action during either start-up or execution) or an internal error that shuts-down the system, and a mission is never completed. However, if an experiment is affected by the above issues it is invalidated, and for experiments that experience partial failure, those results are identified and removed from the final data gathered.

As before, tasks T_R are assigned sequentially to each robot R and the assignments are fixed for all experiments in this work. *robot*_1 is assigned tasks T_1 = {1,4,7}, *robot*_2 is assigned tasks T_2 = {2,5} and finally *robot*_3 is assigned with tasks T_3 = {3,6}, which is illustrated in Figure 4.8 of Chapter 4. The physical experiments are repeated 5 times per experimental configuration.

The experiments conducted in this chapter use the same experiment configuration as those of the simulated experiments conducted in Chapter 5, apart from experiments that use the newly introduced *ESD* perturbation. Therefore, Tables 5.2 and 5.3 from Chapter 5 are used to represent the physical experiments conducted using the *WLAN* and *AH* network type, respectively. Additionally, Table 6.3 shows the experiments conducted with the newly introduced *ESD* perturbation. Table 5.2 consists of 8 configurations and for each configuration 5 experiments are performed, therefore 40 experiment runs in total. Tables 5.3 and 6.3 consist of 24 configurations and for each configuration 5 experiments are performed, therefore 120 experiment runs in total. There are 160 physical experiments conducted combining all configurations.

Experiment schema
$F_{29} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{ESD\} \times \{NB\}$
$F_{30} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{ESD\} \times \{NB\}$
$F_{31} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{ESD\} \times \{NB\}$
$F_{32} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{ESD\} \times \{NB\}$
$F_{33} = \{S_1\} \times \{AH\} \times \{SPL0\} \times \{ESD\} \times \{LF\}$
$F_{34} = \{S_1\} \times \{AH\} \times \{SPL25\} \times \{ESD\} \times \{LF\}$
$F_{35} = \{S_1\} \times \{AH\} \times \{SPL50\} \times \{ESD\} \times \{LF\}$
$F_{36} = \{S_1\} \times \{AH\} \times \{SPL75\} \times \{ESD\} \times \{LF\}$

TABLE 6.3: Experiment configurations for the *AH* network type and the *ESD* network perturbation.

6.5 Results

The same performance metrics are investigated as in Chapter 5, *Total Movement Time* (Figure 6.3, Table 6.4), *Leader Disconnect Time* (Figure 6.4, Table 6.5), *Follower Waiting Time* (Figure 6.5, Table 6.6), *Overall Near Collisions* (Figure 6.6, Table 6.7), *Overall Delay Time* (Figure 6.7), *Overall Idle Time* (Figure 6.8, Table 6.8), *Total Distance Travelled* (Figure 6.9, Table 6.9), *Average Failed Tasks* (Figure 6.10, Table 6.10), *Successful Communication* (Figure 6.11) and *Minimum and Maximum Separation* (Figure 6.12). The Figures represent the same performance metrics as those displayed in the previous chapter. However, as another network perturbation is introduced (*ESD*) each Figure is now split into four sub-figures instead of three; the first represents experiments performed using *WLAN* and *SLT*; the second represents *AH* and *SLT*; the third *AH* and *SSD*; the final one represents *AH* and the newly introduced *ESD*. The error bars in the Figures reflect the standard deviation between runs, however as explained in the previous chapter, *Mission Success* does not have a standard deviation. Because, the physical results are conducted on smaller sample-sizes the statistical analysis is considerably weak. Therefore, if a statistical analysis shows significance in the physical results shown here, it is compared to the corresponding simulation results in Chapter 5, and if the results and trends agree, then deductions can be made and it will be plausible to accept them.

The physical experiment results have a different impact on some aspects of the MRT compared to the simulated experiment results. However, there are also similarities in the negative and positive trends for some of the physical results compared to the results in Chapter 5, which are further discussed below. Moreover, **no** experiments successfully communicated all their *task status messages* for *NB*.

The Mann-Whitney U test is used for statistical analysis, as before. However, due to the small sample size of the data, the analyses and hypotheses are carefully scrutinised. Moreover, it is clear that the results achieved here are by no means statistically extensive, but are good enough as a proof-of-concept and show clear benefits of the proposed framework. To improve the statistical soundness of the hypotheses and to draw extensive conclusions from the results a larger number of experiments, i.e. $n \geq 30$, will need to be conducted.

Total Movement Time follows a decreasing trend, as revealed by Table 6.4, dissimilar to the equivalent simulation results in Chapter 5. It cannot be said with certainty that the performance metric is different between the simulated and physical experiments. However, experiments with *LF* have approximately a three-fold increase in *Total Movement Time* compared to *NB*. This is very similar to the simulated results in Chapter 5, and it shows confidence in the physical results of *Total Movement Time* between *LF* and *NB*.

The *Leader Disconnect Time* is *LF* specific, and shows no real changes in trend for experiments conducted with *WLAN* but shows a decreasing trend for experiments conducted with *AH*. However, due to the weak statistical significance of the physical results and the weak trend results obtained in the simulation experiments, in Chapter 5, for this metric, the trend displayed by the *AH* results is not significant enough and can be determined statistically insignificant. The *Follower Waiting Time* is also *LF* specific, and the comparative simulation results have very uniform trend and a table was not made for it. However, for the physical results the *Follower Waiting Time* has a noticeably decreasing trend. Because, the visual and statistical analysis results for the simulated experiments were insignificant for *Follower Waiting Time*, it cannot be said confidently that the physical results here are statistically significant.

The *Overall Near Collision* metric experiences similar trends to the corresponding simulated results in Chapter 5. Therefore, as previously concluded another alternative hypothesis is investigated, which is whether the observations from the ideal case (i.e. *SPL0*) are "greater" from the other observations (SPLX). It should be noted that a large Mann-Whitney U-value now strongly supports the $\mathbf{H_A}$, while the p-value and effect size remain unchanged. What is immediately noticed about the results in Figure 6.6 and Table 6.7 is that $\mathbf{H_A}$ is accepted in the majority of the time and the trend either increases or remains with extremely high U-value in comparison to the idle case of *SPL0*. The physical results here support the simulated results, therefore there is certainty in the statistical significance.

The *Overall Idle Time* results have generally a uniform trend and no significant result for experiments using *WLAN* or *NB*. There is some interesting overlap of significant results for *AH* when either *SSD* and *ESD* are used. The significance indicates that both simulated and physical results show that increasing *SPL* has an impact on *Overall Idle Time* for the *AH* network type and *LF* behaviour, which is a reasonable conclusion and expected.

Total Distance Travelled indicates a reversed alternative hypothesis, similar to that of *Overall Near Collision*. However, judging the erratic trends observed in Table 6.9 and that the simulated results mostly have uniform trends, the physical results are decidedly inconclusive.

The trends examined in Table 6.10, for the *Average Failed Tasks* performance metric show very strong likeness to those of the simulated results. The trends display minimal U-value or show a large decrease in U-value for experiments conducted with the *NB* behaviour, and $\mathbf{H_A}$ is accepted in half the pair-wise experiment comparisons.

Finally, *Successful Missions* performance metric (Figure 6.11) does not have a statistics table. However, visually observing the corresponding successful missions and failed missions, it can be concluded that the simulated and physical results have very strong affinity. For *LF* every single experiment had successful communication, i.e. 100% mission success, whereas communication for experiments with *NB* failed every time.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	15	0.73	0.23	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	13	0.58	0.05	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	8	0.20	-0.42	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	15	0.73	0.23	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	5	0.07	-0.70	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	1	0.01	-1.07	$\mathbf{H_A}$
AH					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	8	0.36	-0.22	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	15	0.73	0.23	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	14	0.86	0.44	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL25, SSD}	0	0.03	-1.06	$\mathbf{H_A}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	6	0.33	-0.29	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	4	0.16	-0.58	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL25, ESD}	5	0.07	-0.70	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL50, ESD}	12	0.50	-0.05	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL75, ESD}	12	0.50	-0.05	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	25	1.0	1.17	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	9	0.27	-0.33	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	5	0.07	-0.70	$\mathbf{H_0}$

TABLE 6.4: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Total Movement Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
AH					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	14	0.66	0.14	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	3	0.03	-0.89	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	5	0.07	-0.70	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	13	0.58	0.05	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	4	0.05	-0.79	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	2	0.02	-0.98	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL25, ESD}	23	0.99	0.98	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL50, ESD}	15	0.73	0.23	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL75, ESD}	8	0.20	-0.42	$\mathbf{H_0}$

TABLE 6.5: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Leader Disconnect Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	Experiment Label 2	U	p	effect size	Hypothesis (H_0/H_A)
WLAN					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	18	0.89	0.51	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	12	0.50	-0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	1	0.01	-1.07	$\mathbf{H_A}$
AH					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	6	0.11	-0.61	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	8	0.20	-0.42	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	1	0.01	-1.07	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	13	0.58	0.05	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	12	0.05	-0.05	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	5	0.07	-0.70	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL25, ESD}	11	0.42	-0.14	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL50, ESD}	13	0.58	0.05	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL75, ESD}	5	0.07	-0.70	$\mathbf{H_0}$

TABLE 6.6: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Follower Waiting Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	20	0.07	0.70	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	25	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	25	0.00	1.17	$\mathbf{H_A}$
AH					
{LF, SPL0, SLT}	{LF, SPL25, SLT}	20	0.07	0.70	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	19	0.10	0.61	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	25	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL25, SSD}	25	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	25	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	23	0.01	1.03	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL25, ESD}	24	0.01	1.07	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL50, ESD}	25	0.00	1.17	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL75, ESD}	23	0.01	1.17	$\mathbf{H_A}$

TABLE 6.7: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Overall Near Collision*, testing the alternative hypothesis that observations from the ideal case (*SPL0*) are **greater** than those of other *SPLX* observations.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
AH					
{LF, SPL0, SSD}	{LF, SPL25, SSD}	9	0.27	-0.33	$\mathbf{H_0}$
{LF, SPL0, SSD}	{LF, SPL50, SSD}	0	0.00	-1.17	$\mathbf{H_A}$
{LF, SPL0, SSD}	{LF, SPL75, SSD}	0	0.01	-1.17	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL25, ESD}	4	0.04	-0.78	$\mathbf{H_A}$
{LF, SPL0, ESD}	{LF, SPL50, ESD}	5	0.07	-0.70	$\mathbf{H_0}$
{LF, SPL0, ESD}	{LF, SPL75, ESD}	5	0.07	-0.70	$\mathbf{H_0}$

TABLE 6.8: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Overall Idle Time*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	18	0.89	0.51	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	17	0.85	0.42	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	12	0.50	-0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL25, SLT}	12	0.50	-0.05	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL50, SLT}	9	0.27	-0.33	$\mathbf{H_0}$
{LF, SPL0, SLT}	{LF, SPL75, SLT}	9	0.27	-0.33	$\mathbf{H_0}$
AH					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	25	1.0	1.17	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	12	0.50	-0.05	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	13	0.80	0.33	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL25, SSD}	0	0.03	-1.06	$\mathbf{H_A}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	3	0.10	-0.72	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	4	0.16	-0.58	$\mathbf{H_0}$

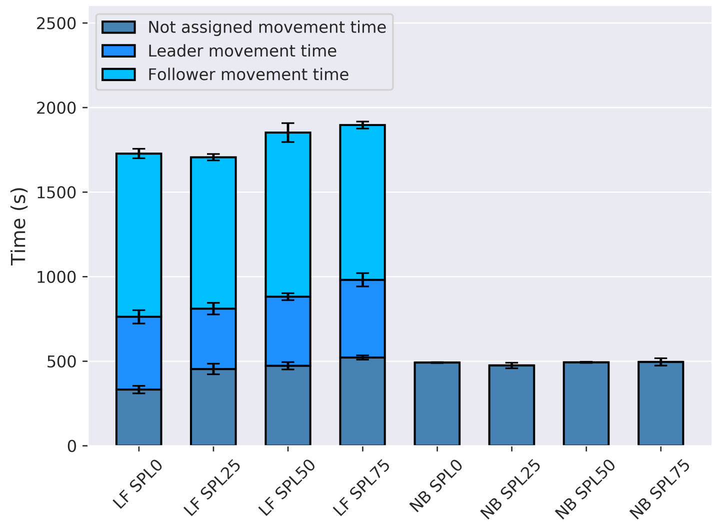
TABLE 6.9: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Total Distance Travelled*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
WLAN					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	0	0.00	-1.17	$\mathbf{H_A}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	0	0.00	-1.17	$\mathbf{H_A}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	0	0.00	-1.17	$\mathbf{H_A}$
AH					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	7	0.19	-0.27	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	10	0.21	-0.23	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	2	0.02	-0.82	$\mathbf{H_A}$
{NB, SPL0, SSD}	{NB, SPL25, SSD}	3	0.14	-0.53	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	3	0.07	-0.72	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	1	0.03	-0.94	$\mathbf{H_A}$
{NB, SPL0, ESD}	{NB, SPL25, ESD}	25	1.0	1.17	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL50, ESD}	2	0.01	-0.93	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL75, ESD}	0	0.00	-1.17	$\mathbf{H_A}$

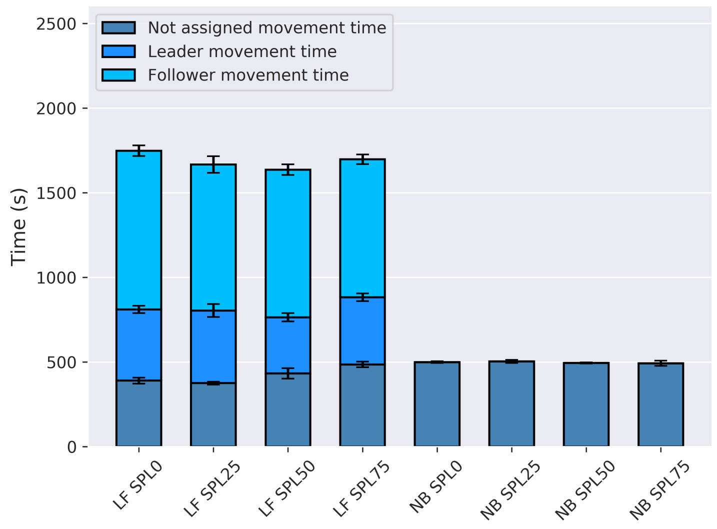
TABLE 6.10: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for the *Average Failed Tasks*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.

Experiment Label 1	**Experiment Label 2**	**U**	**p**	**effect size**	**Hypothesis (H_0/H_A)**
AH					
{NB, SPL0, SLT}	{NB, SPL25, SLT}	25	1.0	1.17	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL50, SLT}	11	0.42	-0.14	$\mathbf{H_0}$
{NB, SPL0, SLT}	{NB, SPL75, SLT}	4	0.08	-0.66	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL25, SSD}	0	0.02	-1.06	$\mathbf{H_A}$
{NB, SPL0, SSD}	{NB, SPL50, SSD}	25	1.0	1.17	$\mathbf{H_0}$
{NB, SPL0, SSD}	{NB, SPL75, SSD}	1	0.03	-1.01	$\mathbf{H_A}$
{NB, SPL0, ESD}	{NB, SPL25, ESD}	19	0.93	0.61	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL50, ESD}	11	0.42	-0.14	$\mathbf{H_0}$
{NB, SPL0, ESD}	{NB, SPL75, ESD}	7	0.15	-0.51	$\mathbf{H_0}$

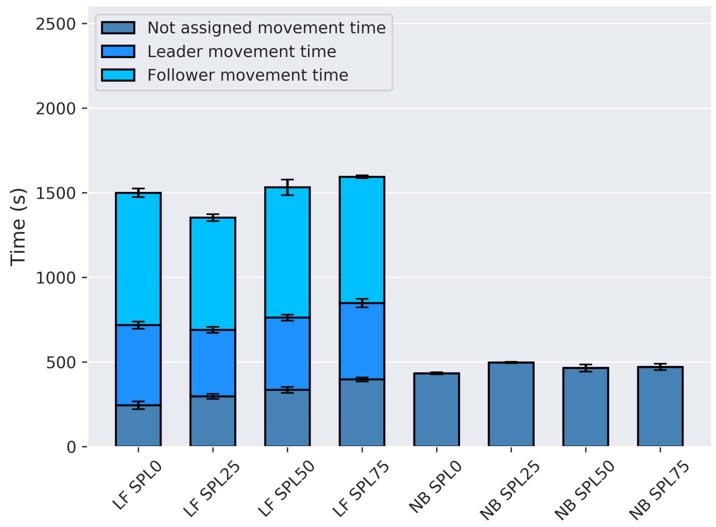
TABLE 6.11: U-value, p-value, effect size and hypothesis outcome of *MRComm* physical robot experiments for *Maximum Separation*, comparing the ideal case (i.e. *SPL0*) versus all other *SPLX* values.



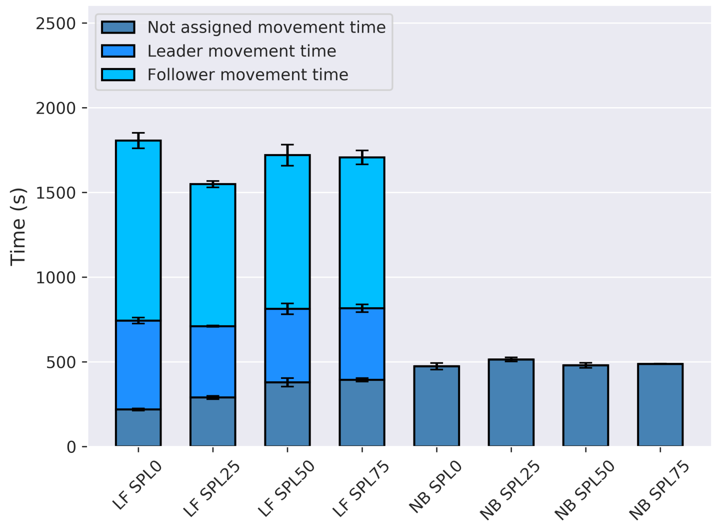
(a) Total Movement Time for *WLAN* with *SLT*



(b) Total Movement Time for *AH* with *SLT*

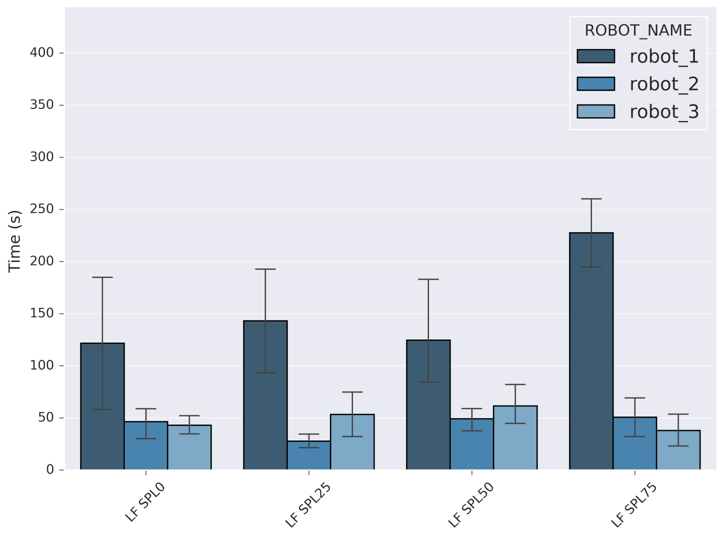


(c) Total Movement Time for *AH* with *SSD*

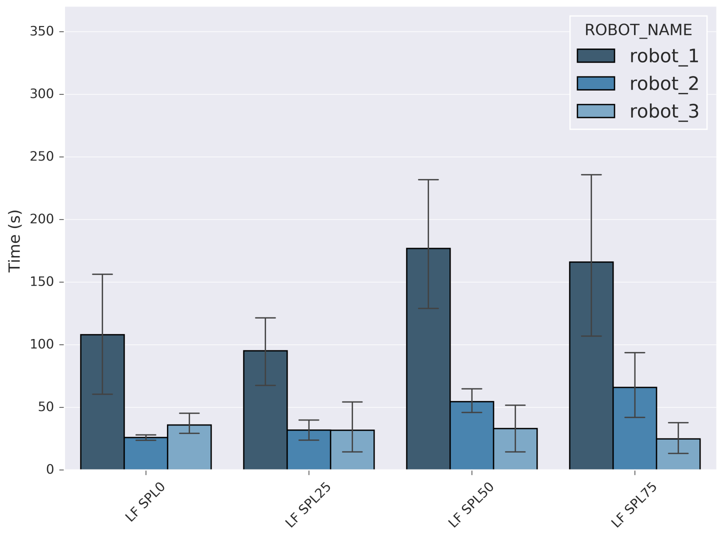


(d) Total Movement Time for *AH* with *ESD*

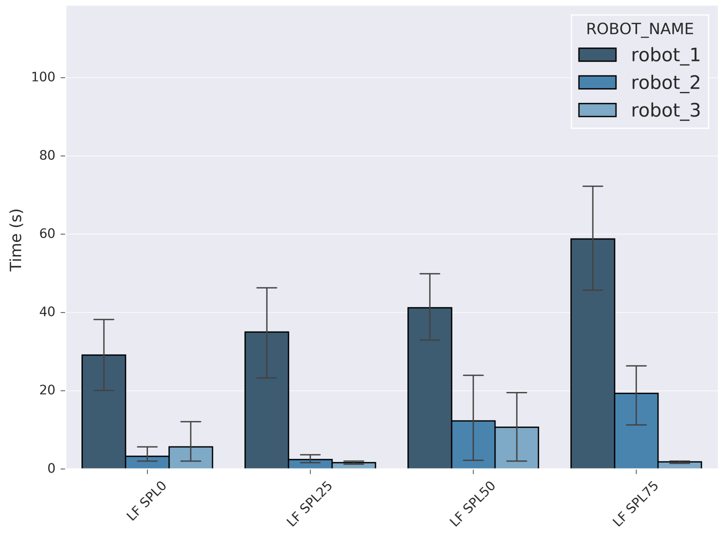
FIGURE 6.3: Total Movement Time stacked for all experiment configurations.



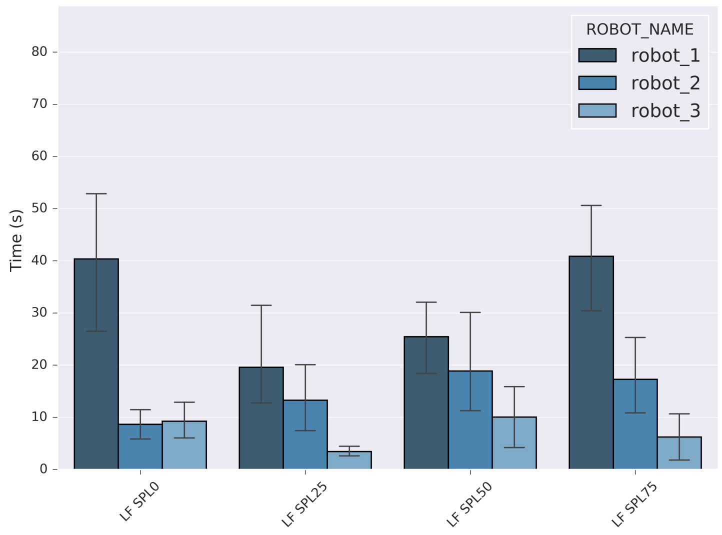
(a) Leader Disconnect Time for *WLAN* with *SLT*



(b) Leader Disconnect Time for *AH* with *SLT*

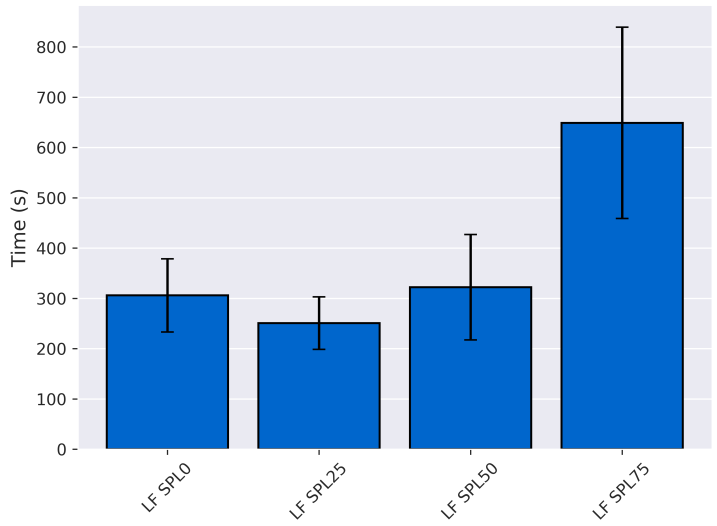


(c) Leader Disconnect Time for *AH* with *SSD*

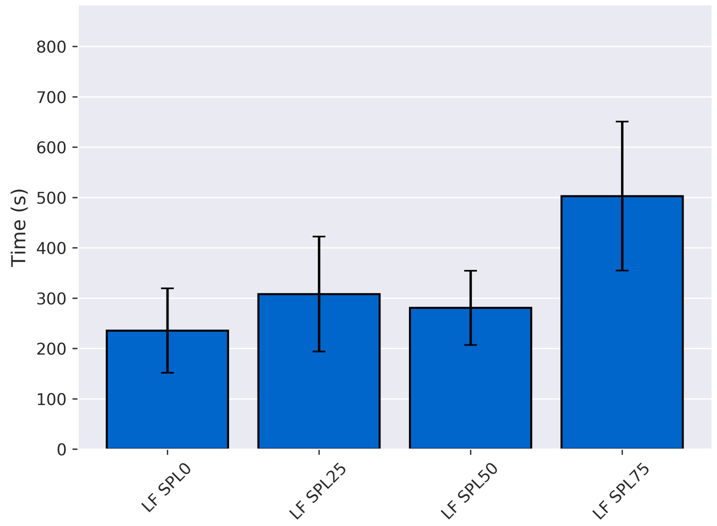


(d) Leader Disconnect Time for *AH* with *ESD*

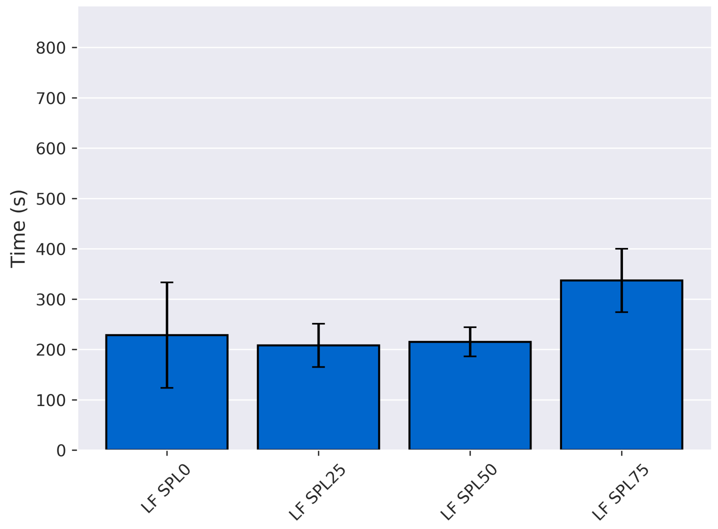
FIGURE 6.4: Leader Disconnect Time per robot for all experiment configurations.



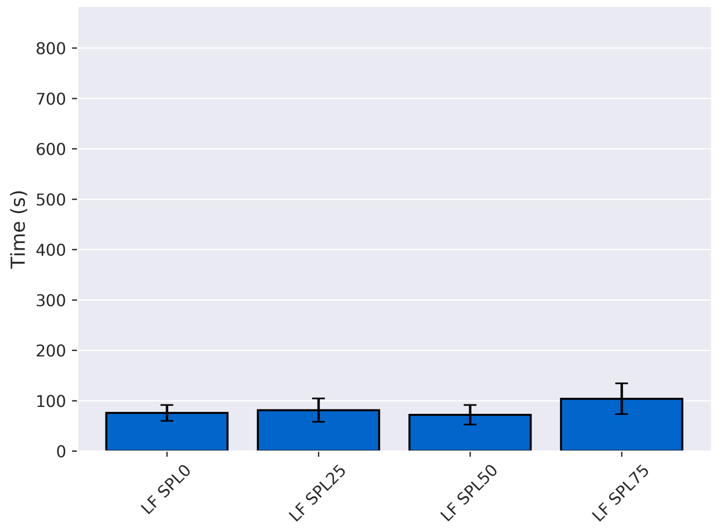
(a) Follower Waiting Time for *WLAN* with *SLT*



(b) Follower Waiting Time for *AH* with *SLT*

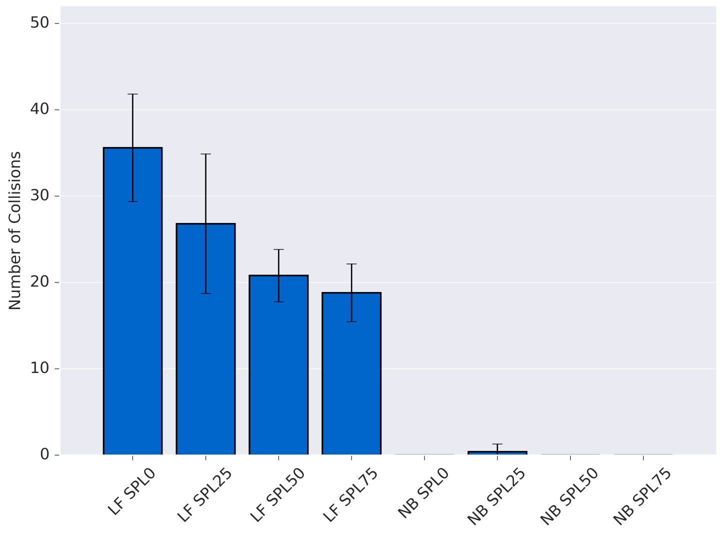


(c) Follower Waiting Time for *AH* with *SSD*

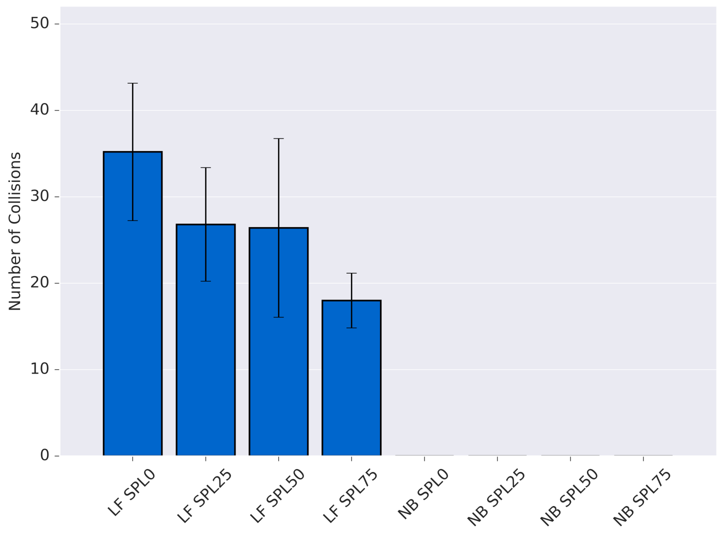


(d) Follower Waiting Time for *AH* with *ESD*

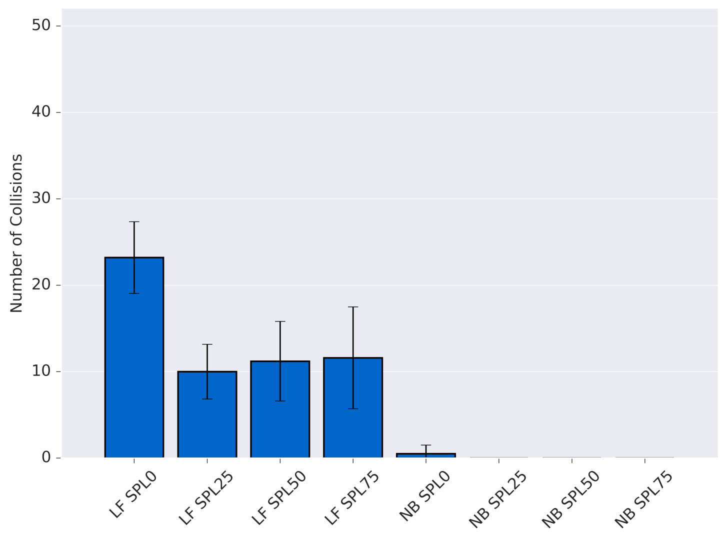
FIGURE 6.5: Follower Waiting Time for all experiment configurations.



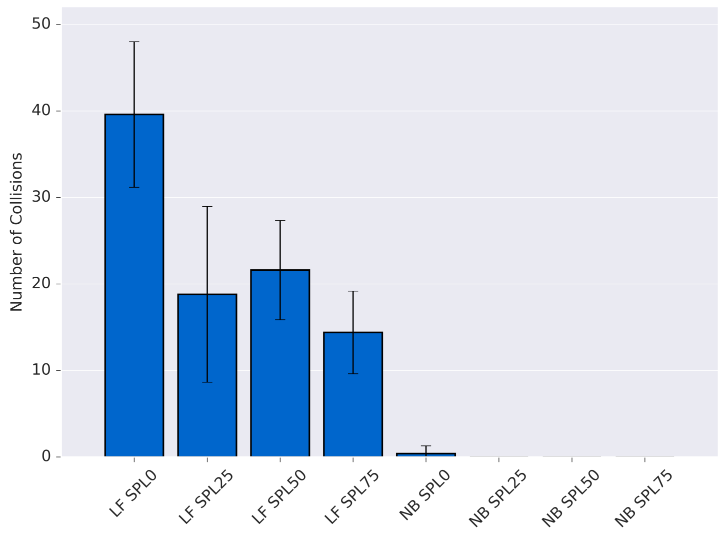
(a) Overall Near Collisions for *WLAN* with *SLT*



(b) Overall Near Collisions for *AH* with *SLT*

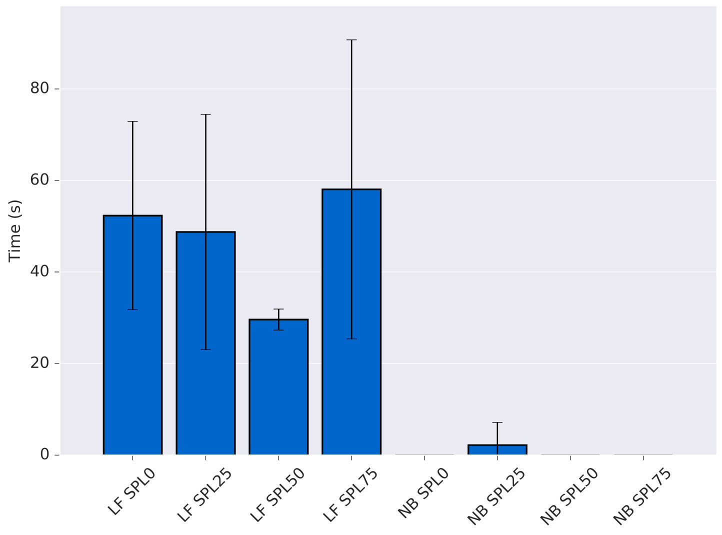


(c) Overall Near Collisions for *AH* with *SSD*

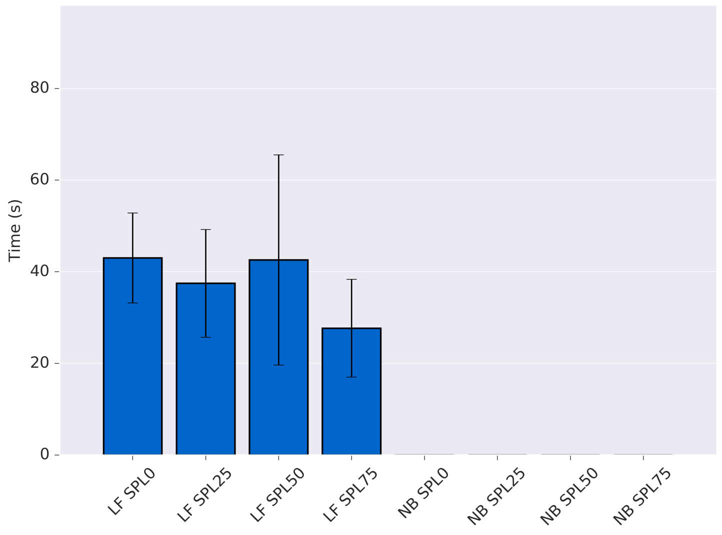


(d) Overall Near Collisions for *AH* with *ESD*

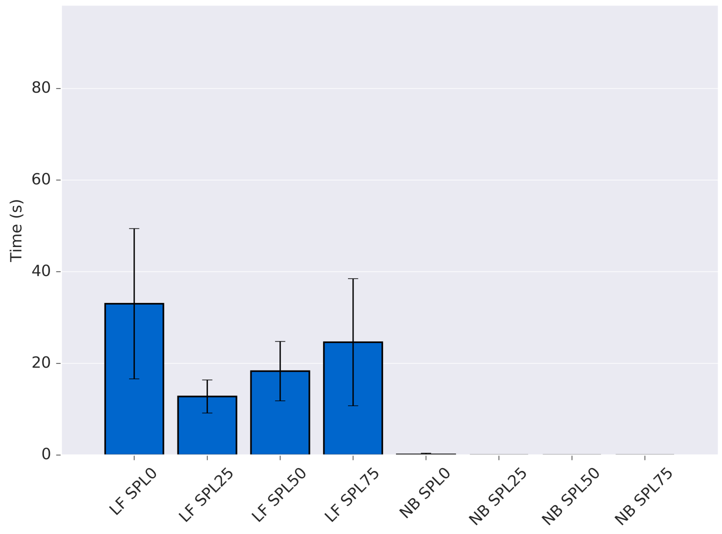
FIGURE 6.6: Overall Near Collision for all experiment configurations.



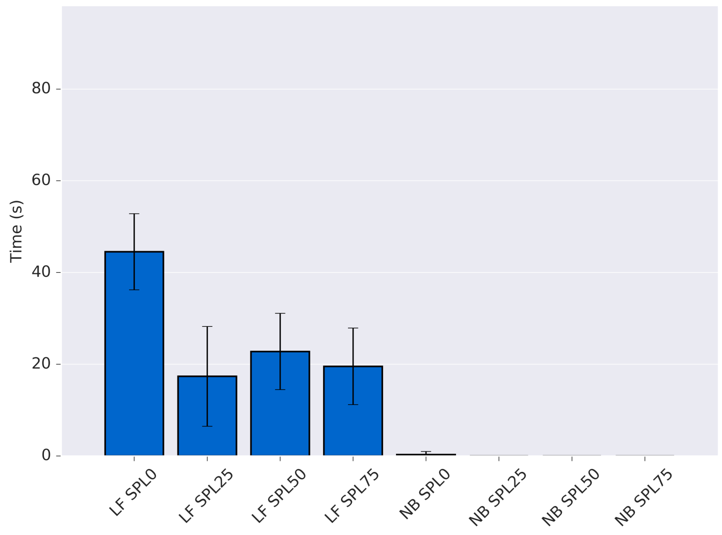
(a) Overall Delay Time for *WLAN* with *SLT*



(b) Overall Delay Time for *AH* with *SLT*

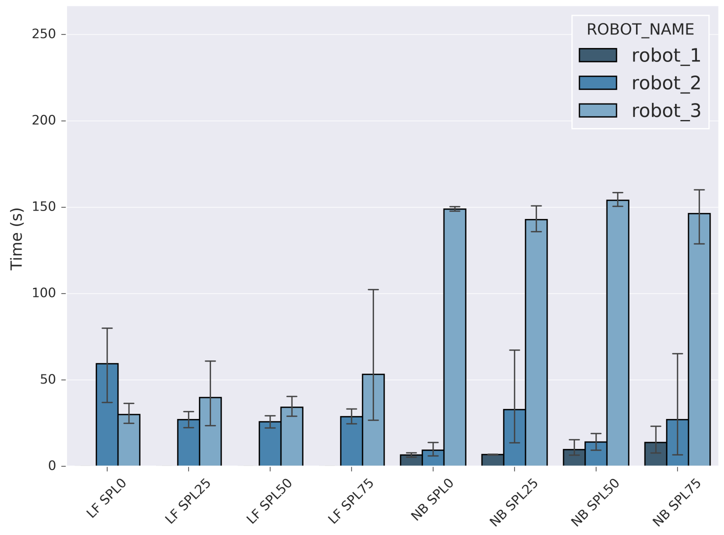


(c) Overall Delay Time for *AH* with *SSD*

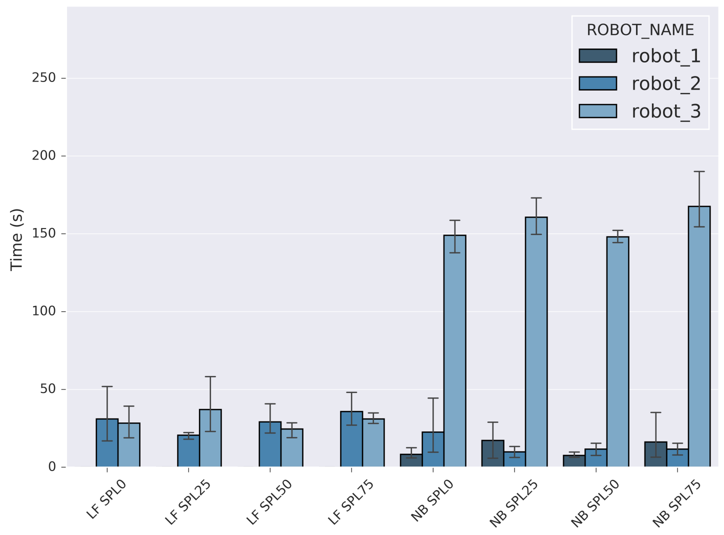


(d) Overall Delay Time for *AH* with *ESD*

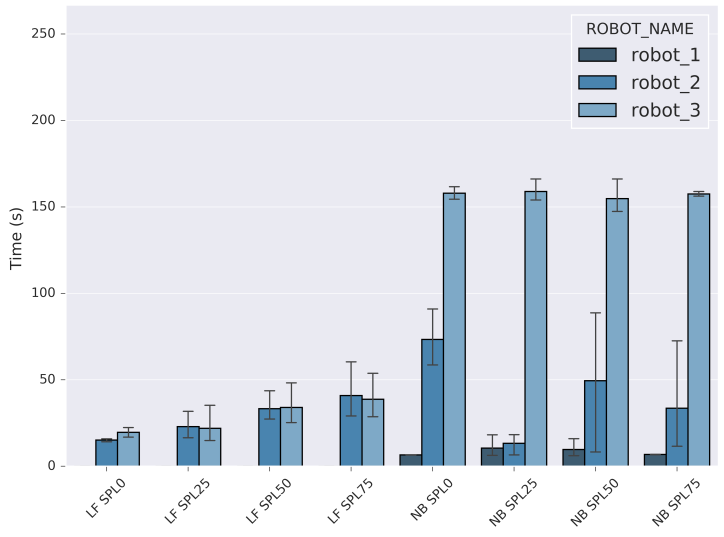
FIGURE 6.7: Overall Delay Time for all experiment configurations.



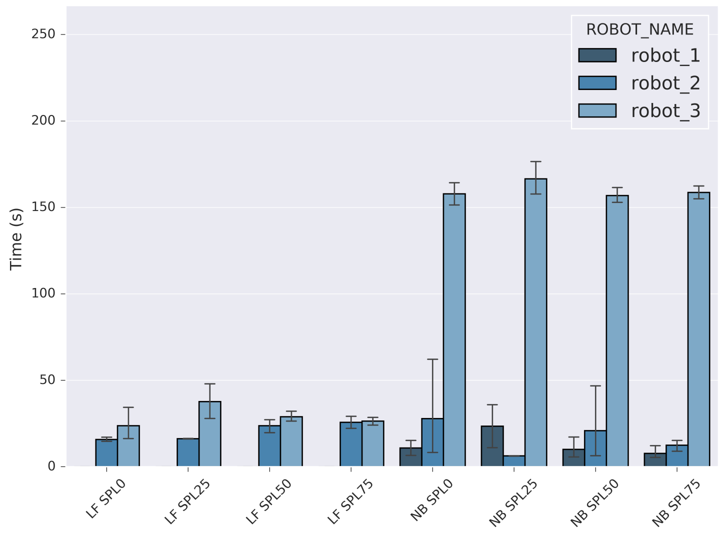
(a) Overall Idle Time for *WLAN* with *SLT*



(b) Overall Idle Time for *AH* with *SLT*

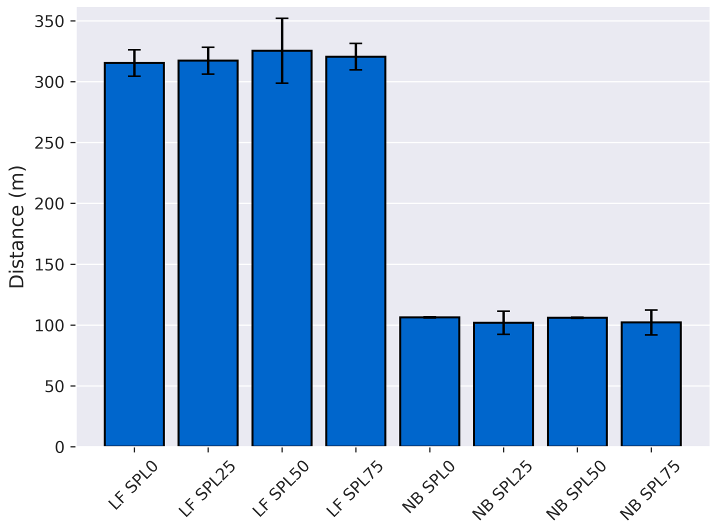


(c) Overall Idle Time for *AH* with *SSD*

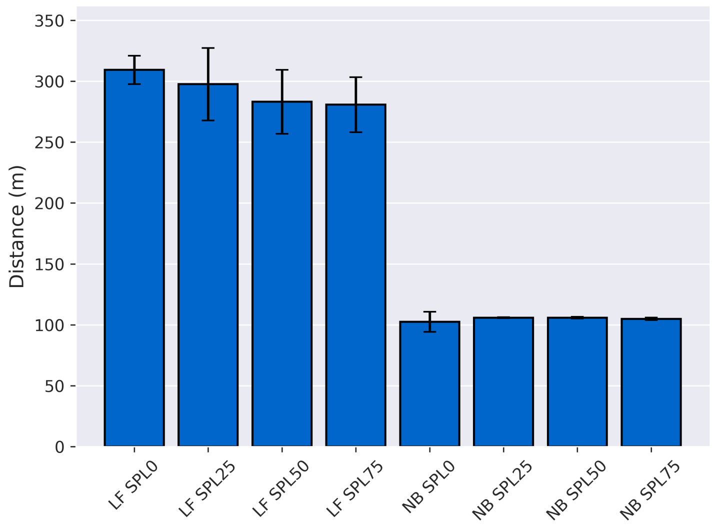


(d) Overall Idle Time for *AH* with *ESD*

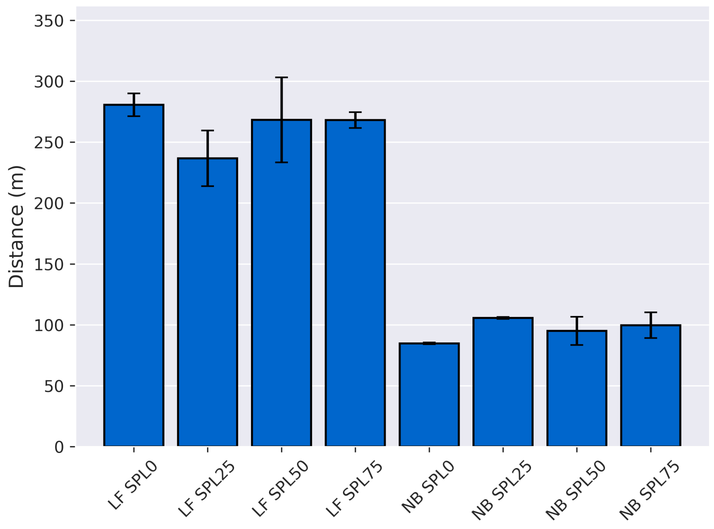
FIGURE 6.8: Overall Idle Time per robot for all experiment configurations.



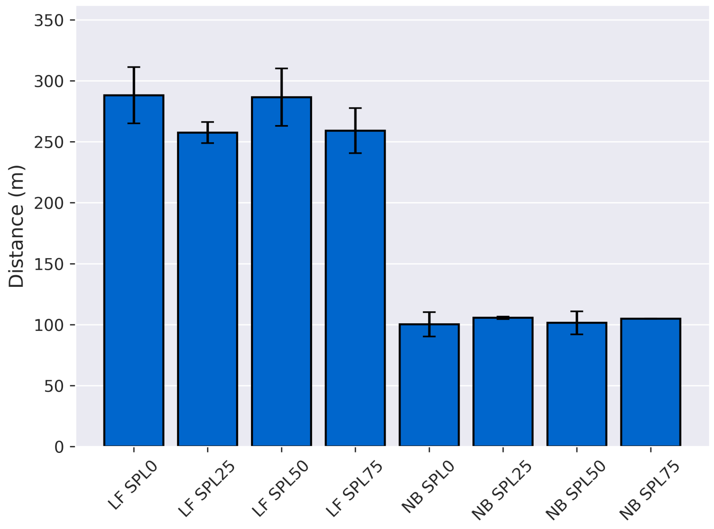
(a) Total Distance Travelled for *WLAN* with *SLT*



(b) Total Distance Travelled for *AH* with *SLT*

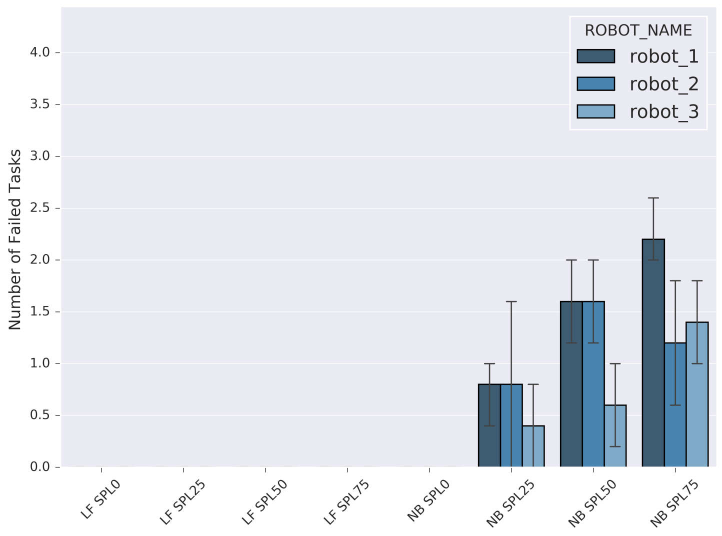


(c) Total Distance Travelled for *AH* with *SSD*

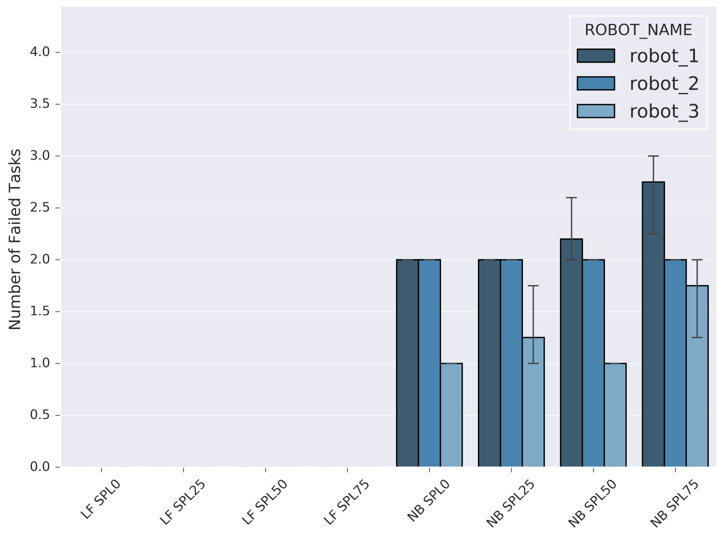


(d) Total Distance Travelled for *AH* with *ESD*

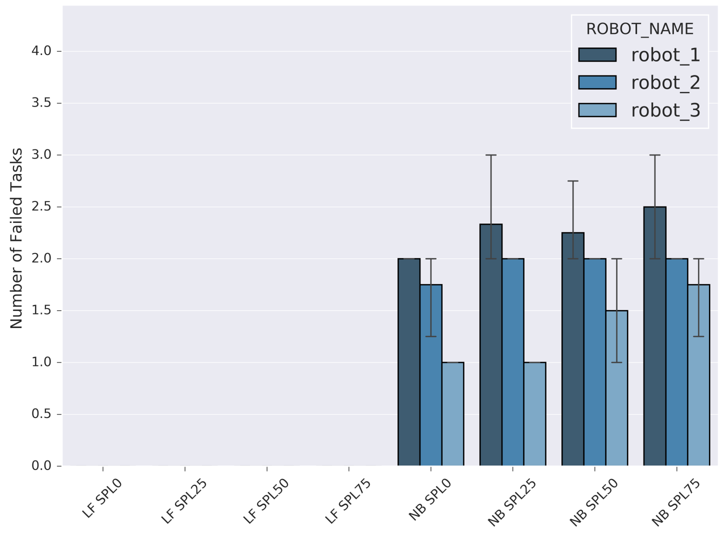
FIGURE 6.9: Total Distance Travelled for all experiment configurations.



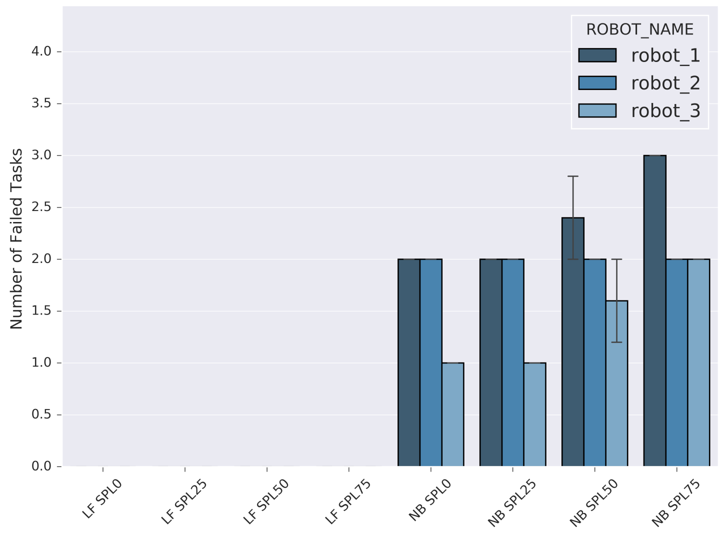
(a) Average Failed Tasks for *WLAN* with *SLT*



(b) Average Failed Tasks for *AH* with *SLT*

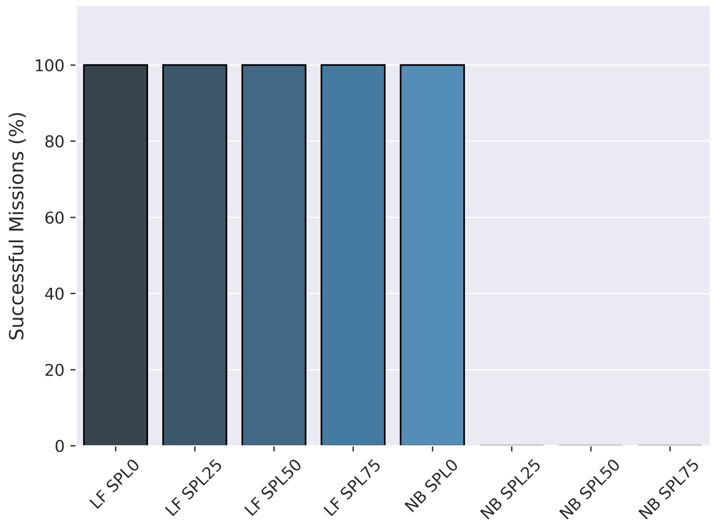


(c) Average Failed Tasks for *AH* with *SSD*

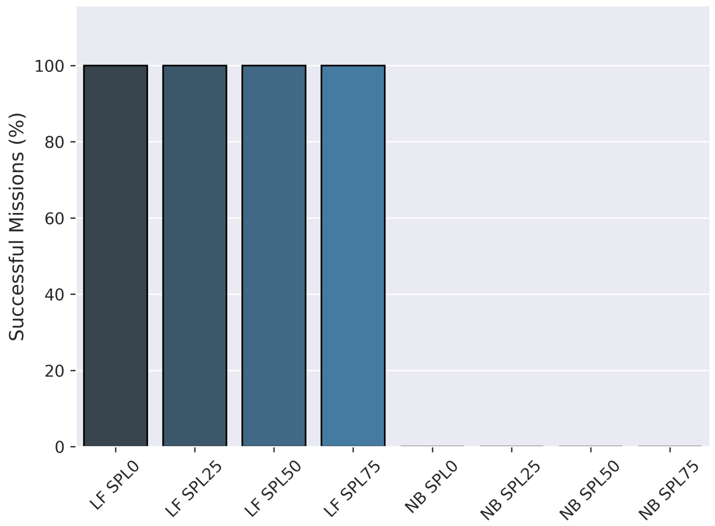


(d) Average Failed Tasks for *AH* with *ESD*

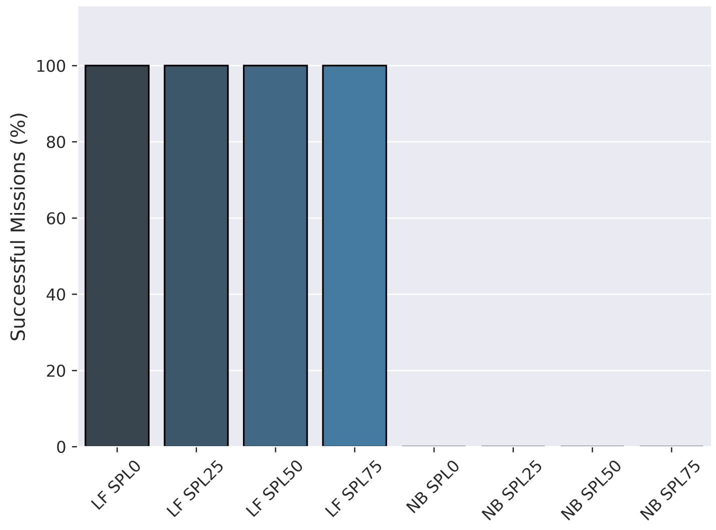
FIGURE 6.10: Average Failed Tasks per robot for all experiment configurations.



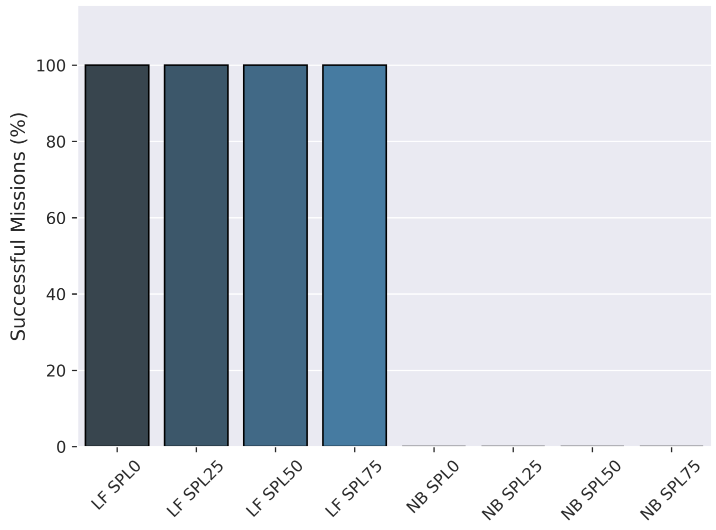
(a) Successful Communication in Missions for *WLAN* with *SLT*



(b) Successful Communication in Missions for *AH* with *SLT*

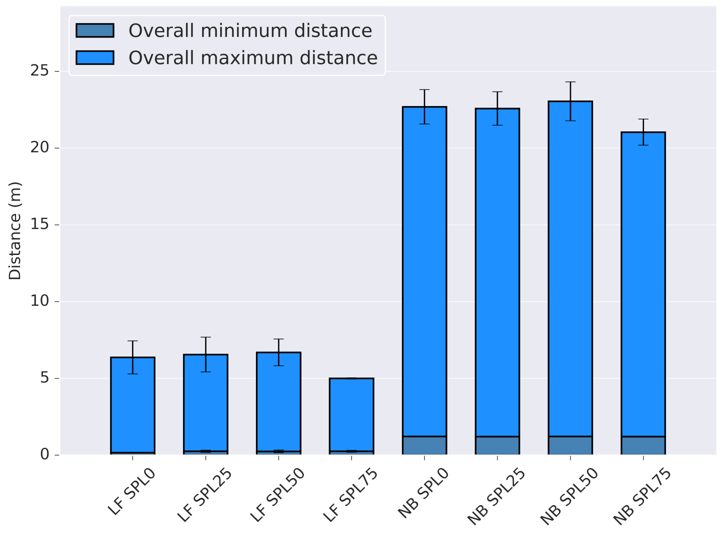


(c) Successful Communication in Missions for *AH* with *SSD*

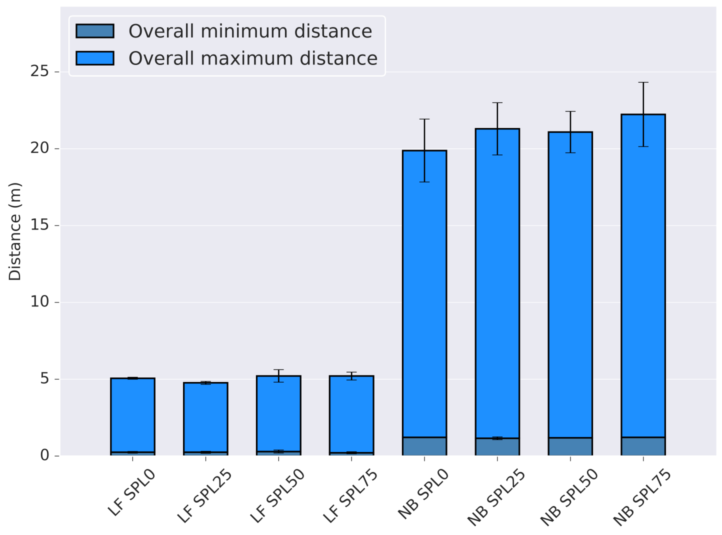


(d) Successful Communication in Missions for *AH* with *ESD*

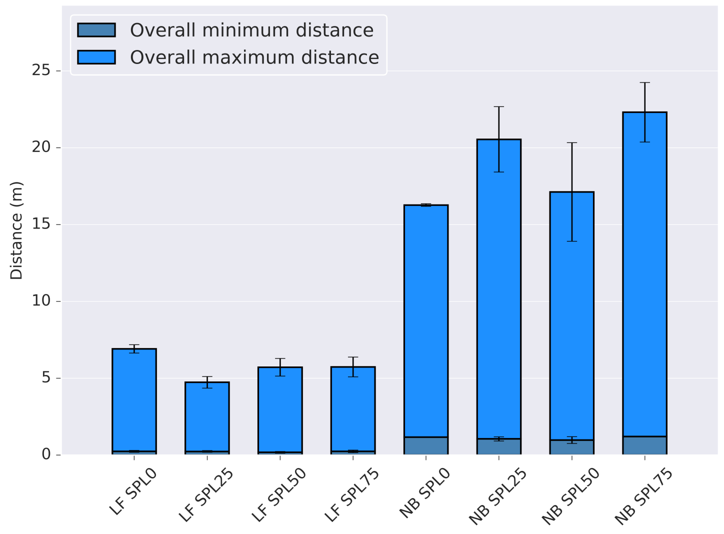
FIGURE 6.11: Percentage of Successful (Communication) Missions for all experiment configurations.



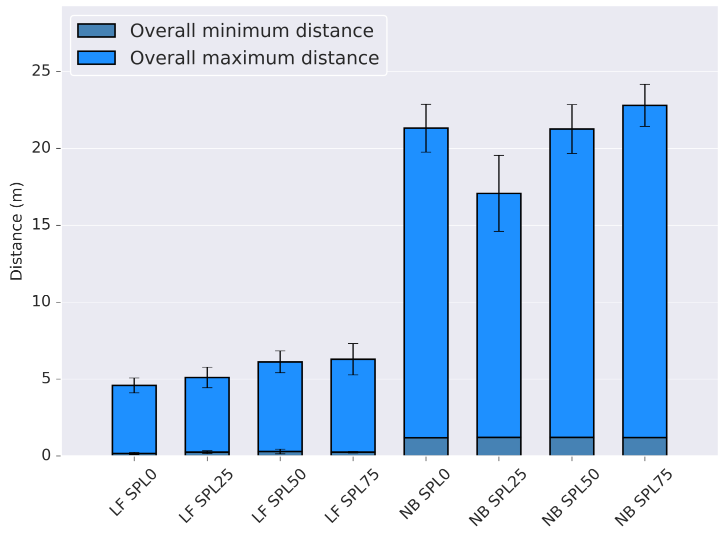
(a) Minimum and Maximum separation for *WLAN* with *SLT*



(b) Minimum and Maximum separation for *AH* with *SLT*



(c) Minimum and Maximum separation for *AH* with *SSD*



(d) Minimum and Maximum separation for *AH* with *ESD*

FIGURE 6.12: Minimum and Maximum separation for all experiment configurations.

6.6 Discussion

The *Total Movement Time* metric for *LF* is made up of the three sub-parts as described in Section 3.9, which is also demonstrated by the results in Figure 6.3. It is noticeable that for simulated experiments using *NB*, *Total Movement Time* is lower than for the physical experiments and vice versa for the *LF* behaviour. However, broadly examining the trend of results for *LF*, the physical experiments show that generally *Total Movement Time* decreases as *SPL* increases, which is the exact opposite for the simulated experiments. The results are not significant enough to conclude with certainty that the observed decreasing trend is valid or what it entails. The *NB* remain largely unchanged throughout both simulated and physical experiments.

In general, simulated and physical experiment results for the *Leader Disconnect Time* have similar trends and patterns. The important result to consider is that *Leader Disconnect Time* is approximately two times lower for experiments using the *SSD* configuration in both simulated and physical experiments, furthermore the same is true for experiment configurations using *ESD* for the physical experiments. This indicates that the more realistic threshold perturbations, *SSD* and *ESD*, are better suited for experiments examining communication quality than *SLT*.

The *Follower Waiting Time* results depicted, reveal possible advantages of having a smaller and more dynamic *warning* threshold applied on a network perturbation compared to that of a fixed and larger threshold, i.e. *SLT* approx. 4.0 m. The *Follower Waiting Time* results using *SSD*, as displayed in Figure 6.5c, are approximately twofold that of the *Follower Waiting Time* results using *ESD*, displayed in Figure 6.5d. However, the *Follower Waiting Time* results for experiments conducted with *SLT* using either network type (Figures 6.5a and 6.5b), approach values three times higher than those of the *Follower Waiting Time* with *ESD*. This is another indicator that the *SLT* threshold is not as accurate in representing signal strength degradation as

SSD and *ESD*.

Network perturbations directly affect all shared messages, including pose (position) messages that are sent/received by robot team members. The results, for instance in Figures 6.10 and 6.11, show that *LF* significantly reduces the effects that perturbations have on shared status messages and overall communication. Although shared position messages are analysed by the *MRComm* framework, none of the implemented behaviours of the framework aid in reducing the impact caused by the perturbations.

The *Overall Near Collisions* results for both simulated and physical experiments are unexpected. It shows clearly that near collisions are decreasing as the number of messages dropped increases. If anything, the number of near collisions should be increasing, however there is an explanation for this result. In fact, there are two issues that are occurring when *Overall Near Collisions* are recorded.

1. The larger number of near collisions that occur for the physical experiments using *LF* are caused by a phenomenon denoted as "micro-collisions". Micro-collisions seem to affect physical more than simulated experiments, as it is noted that there is a twofold increase of *Overall Near Collisions* compared to results in Chapter 5.

2. The fact that collisions are overall decreasing as message loss increases is caused by a very simple problem, which is that team members' positional messages are being lost. A robot does not know if it is in collision or not, which makes it react slower. Reacting slower could cause some near collisions to either turn into real collisions, after which the robots go into recovery mode (i.e. dead reckoning, waiting, etc.), or near collisions do not occur and the robots manage to maneuver around each other. This issue affects both simulated and physical experiments, as the trends of both are very similar.

The first point is covered in-depth in Section 6.6.1 and the second is a by-product of communication quality degradation caused by the *SPL* network perturbation. There are a number of methods that could be used to improve this issue. For example, employing simple visual sensory input and estimating distance to near-by objects and alerting the robot, or complex visual sensory input that identifies near-by objects (i.e. robot, human or wall), if the robot is moving toward these objects and alerting the robot. Visually observing the MRT during some missions in both the simulated and physical environments proved that the robots occasionally collided and needed to use the implemented recovery behaviour to correct their position. The negative effects appear unexpected and hidden, making it hard to interpret how much these issues impact the different performance metrics.

The results of the *Overall Delay Time*, in Figure 6.7, are correlated to the *Overall Near Collisions* metric, as collisions directly increase the amount of delay time that a robot experiences. However, there is an inconsistency in the correlation between the simulated and physical experiment results. As observed, the *Overall Near Collisions* results in Chapter 5 are half that of the physical results for *Overall Near Collisions* obtained here. Moreover, the *Overall Delay Time* metric for the physical experiments is in fact lower in some cases compared to the simulated experiment results for the same metric. The lower *Overall Delay Time* and increase in *Overall Near Collisions* is explained by the micro-collisions effect, Section 6.6.1. However, because it cannot be statistically proven how many micro-collisions occur over actual near collisions, it is inappropriate to report these results as objective. Therefore, the results in Chapter 5 for *Overall Delay Time* and *Overall Near Collisions* more accurately represent the results for these metrics.

The *Overall Idle Time* is not necessarily a metric that proves how efficient a behaviour is, but it does show the amount of time that robot team members waste doing nothing. Figure 6.8 shows that the results across different experiment configurations experience the same trends. However, the main difference in *Overall Idle Time* is between the two behaviours. For example, experiment results using *LF* show that a small amount of *Overall Idle Time* is wasted and is not so disproportionately distributed per robot. Comparatively, experiments using *NB* demonstrate that a large amount of *Overall Idle Time* is wasted and is greatly disproportionately distributed per robot. Furthermore, comparing the *Overall Idle Time* results in Chapter 5 have very similar trends, albeit marginally lower mean.

The results in Figure 6.10 present the average number of tasks that failed to communicate the "SUCCESS" status message per robot. The *Average Failed Tasks* performance metric very clearly highlights the effects of the different behaviours, network types and network perturbations. For experiments with any configuration in combination with *LF*, all task status messages are communicated successfully. On the other hand, for the baseline *NB* behaviour there are distinctions in performance between the different combinations of network type and network perturbations. For example, in Figure 6.10a and configuration $\langle NB, WLAN, SPL0, SLT \rangle$ communication is perfect, which is the expected result. However, for the similar configuration $\langle NB, AH, SPL0, SLT \rangle$ in Figure 6.10b on average five out of the seven tasks are not communicated successfully, which is caused only by the change in network type. The overall results show that for any configuration using *NB* in combination with *WLAN*, the impact on communication performance is severe after $\geq SPL25$. Furthermore, the impact on communication performance is greatly increased when using *NB* and *AH*, as tasks are no longer received after $\geq SPL0$. The communication results for experiment configurations with *LF* are perfect and demonstrate the ability of the behaviour to adapt to dynamic network parameters, while still allowing each robot to operate individually and in the team.

Successful Missions results in Figure 6.11 complement the results in Figure 6.10, by more broadly showing how many missions completed without a single *task status message* being dropped (not received). The results observed in Chapter 5, showed that 20% of missions completed successfully for *NB* with $SPL25$, which was 0% here. This result is most likely due to the fact that physical experiments were not repeated as many times as simulated experiments in Chapter 5. However, it is a strong enough result to determine that even a small amount of dropped messages greatly impacts MRT communication performance. However, the *LF* behaviour demonstrates 100% communication capabilities in the MRT regardless of network type or perturbation used. The results from Figures 6.10 and 6.11 are promising for the novel functionality of the *LF* behaviour and the AON messaging function introduced to *MRComm*.

Indeed, there were some issues that were encountered in the physical experiments. These were not recorded in the performance metrics, but instead were visually observed and investigated during the design stage. The first issue was that during mission execution for $\geq SPL50$ in combination with any secondary network perturbation (i.e. $\{SLT, SSD, ESD\}$) the system would occasionally freeze. Investigating the issue led to the conclusion that the implementation of the AON message function in *MRComm* was not checking the order of received *task status messages*. Therefore, it was possible that a robot receives a message out of order forcing the *task execution* agent to switch from its current state to a previous state, which causes a deadlock for the affected robot. The whole experiment result would then discarded. Furthermore, it was noted that on occasion, because of naturally occurring faults in the system, an experiment with any configuration could fail either due to a mission timing out or a ROS system failure, such as the navigation system exhibiting local minimal error causing an infinite loop in the robot base control code, etc. ROS system errors generally occur if a system is poorly configured and parameters are not properly chosen. However, the rate of experiment failure was indeterminate and unreliable, leading to the assumption that the system errors are unpredictable and may be related to the work mentioned by Ore et al. [103].

6.6.1 Micro-Collisions

Micro-collisions are a phenomenon that affect physical robots more than simulated ones. However, it is difficult to estimate by how much more one is affected from the other. There are two reasons that could lead to micro-collisions occurring more often in physical experiments. Firstly, physical robots experience longer message delays and the possibility of messages arriving out of order is increased, due to a noisy physical communication network. This causes incorrect messages to arrive that may lead to false-positive flagging of a near collision, e.g. if the robot is no longer within collision range. The second and more complex reason is due to premature switching of states, which can be worsened if combined with the above reason (i.e. message delays and messages arriving out of order). An example of this is when two robots are moving head-on while a third is about to disconnect at a further distance. What occurs is the two robots about to collide pause to resolve the collision, and before it is resolved, a higher precedence rule warns both robots that another robot has disconnected, these robots are now forced to leave the collision state. The incorrect state switching will occur multiple times for a couple of milliseconds each time, causing multiple near collisions to be recorded in a short span of time as the robots finally get back to a stable state. Another example, which is receiving shared messages from robot team members that require a response.

The collision avoidance used by *MRComm* has two different zones, the "danger zone" and the "event horizon", Figure 6.13 re-plotted from Section 3.6 for convenience. The second zone, "event horizon", triggers an immediate near collision response, as it instantaneously overrides all other event precedence, causing premature switching of states. Since the "event horizon" area is so small it will cause very short (micro) collisions, until a robot is clear of that area, and switching of states happens almost instantly, hence this phenomenon is denoted micro-collisions.

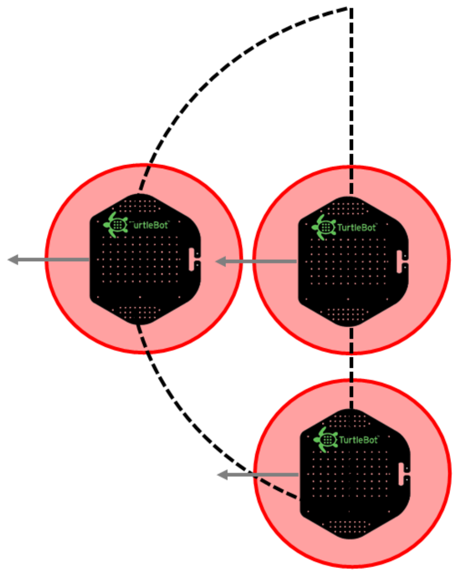


FIGURE 6.13: *MRComm* improved collision avoidance. The dotted semi-circle represents the "danger zone" of only a single robot and the transparent red circles represents the "event horizon" area for all the robots.

6.7 Summary

In this chapter I presented the *ESD* threshold network perturbation and the *ESD Pulse*. *ESD Pulse* has multiple utility functions that could be leveraged in search-and-rescue or faulty robot retrieval scenarios, apart from getting the actual signal strength from a robot team member. Moreover, I compared the results between the predictive *SSD* threshold and the true *ESD* threshold and found that both follow a similar trend, and that some performance metrics have similar results in Section 6.5. The advantages of being able to simulate signal strength degradation in a simulated environment and get considerably close result, can help identify early communication issues in an environment or possible outliers. However, to improve the *SSD* parameter, in future work it will be modelled on data gathered from multiple experiments with a physical multi-robot team first using *ESD*.

The experimental results here are a proof-of-concept of a truly distributed physical multi-robot system completing missions in an operational environment. Where each individual robot records its own data, navigates the environment and self-maintains its own communication (communication aware) to the team. Although the results presented in this chapter can not be considered statistically significant, as the sample size is small, they show that uninterrupted communication of shared messages is successfully carried out for experiments conducted on a physical MRT using the novel *LF* behaviour, identical to the statistically significant results observed in Chapter 5.

Chapter 7

Conclusions and Future Work

This thesis outlines the progress and contributions made toward the multi-robot communication domain. In Section 7.1, I summarise and highlight the main achievements. Section 7.2 briefly compares and showcases some of the important results from Chapters 4, 5 and 6. In Section 7.3, I look at possible improvements to the *MRComm* framework and outline future work. Finally, I conclude and summarise the thesis in Section 7.4.

7.1 Overview

Chapter 2 introduced the underlying principles of communication and multi-robot systems. Furthermore, it examined the components required to design and test a multi-robot communication framework and introduced telecommunication concepts that can be leveraged in MRS (ad-hoc networks, self-reestablishing communication, etc.). The chapter presented biologically [48], [52] and artificially [39], [54] inspired behaviour-based control, which motivated the creation of the communication-aware *LF* behaviour described in Chapter 3. Finally, the related work in Chapter 2 highlighted the significance that communication has on a MRT and demonstrated some works that employ telecommunication concepts in MRS. Here below are listed the contributions of this PhD:

1. The *MRComm* framework was developed (Chapter 3). It defined the network parameters required for analysing communication quality. It introduced two multi-robot coordination behaviours: the baseline *NB* behaviour, which was employed in [9] (and other related works) and which represents standard multi-robot communication in a surveillance or exploration task scenario; and the novel *LF* behaviour, which was developed in this research as a communication-aware behaviour with the objective to control the actions of a multi-robot team and enable continuous communication. A novel messaging protocol, denoted *AON*, was developed to work in conjunction with the *LF* behaviour to improve communication when different network perturbations are affecting multi-robot team experiments. The *MRComm* framework enabled the execution of MRT communication-based experiments in both simulated and physical environments, with minimum changes in functionality between the two. Moreover, for the purpose of executing physical robot experiments, an actual AH network was created for communication, which was made possible with the seamless integration of the ROS-based *FKIE* software package that additionally facilitates a truly distributed multi-robot system.

2. Pre-*MRComm* framework results were gathered as a baseline (Chapter 4), which analysed communication performance in a multi-robot team, using the standard multi-robot coordination behaviour (denoted *NB* in Chapter 3) of the *MRTeAm* framework in [9]. The *SPL* network perturbation was designed to disrupt the communication between robots during the execution of an experiment. It does this by dropping a certain percentage of shared messages.

   Furthermore, the experiments based on the *MRTeAm* framework used the *ROS Master Bridge* for inter-robot communication, which is different in its design compared to *MRComm*. The experiments were conducted on both simulated and physical robots and the results aided in the development of the *MRComm* framework, particularly in the development of the network parameters, performance metrics and in the re-design of the communication. The communication was re-designed as the boundaries explored in this thesis required a more distributed MRT approach (i.e. *AH* communication). As mentioned previously, *MRComm*'s inter-robot communication uses the *FKIE* software package, which allows for a more robust and truly distributed MRS and communication. A new set of experiments were conducted using *MRComm* and its distributed communication method, which were configured similarly to the baseline preliminary experiments conducted with *MRTeAm*. The results from the two frameworks were briefly compared, and showed an improvement and consistency of some performance metrics for *MRComm* over its predecessor, *MRTeAm*.

3. Simulated robot experiments were carried out using *MRComm* and the special *LF* behaviour and *AON* messaging protocol (Chapter 5). Furthermore, the *SLT* and *SSD* network perturbations were designed and tested with these experiments. The results showed promising performance when using the *LF* behaviour compared to the *NB* behaviour. When using the *LF* behaviour, the robot team was able to respond and recover from the perturbations. Therefore, enabling continuous communication throughout the duration of a mission for the simulated robots. However, it was noted that not all performance metrics are positively impacted. This is anticipated and is caused by the design of the *LF* behaviour, which increases the overall mission time and distance travelled by the robots.

4. Low-cost physical robots were used to conduct experiments using an

actual self-maintained *AH* network and are "truly distributed" in their communication and execution of the mission. The truly distributed moniker, described in Chapter 2, requires that each of the main components (control, communication and data recording) of a MRS are organised using a distributed approach, which is achieved and demonstrated by the proof-of-concept experiments conducted in Chapter 6. The physical robot experiments showed similar overall results to those of the simulated robots, when compared with the same experiment configuration. The *ESD* network perturbation (i.e. actual signal strength) was designed and tested with these experiments. It was used to validate how well the simulated version, *SSD*, performed when used with physical robots and additionally with simulated robots. Furthermore, I showed that *SSD* follows a similar trend and that it could hypothetically reach the same output levels as *ESD*. This is very useful when wanting to emulate accurate signal strength degradation performance in a simulated environment. As mentioned previously, the results were similar to those of the simulated robots. Therefore, the results showed completely improved communication performance when using the *LF* behaviour with *AON* as apposed to the standard *NB* behaviour and no special messaging. Similarly to Chapter 5, when robots used the *LF* behaviour, the overall mission time and distance travelled increased, but communication was improved dramatically and so was task completion status.

7.2 Discussion

In Chapter 4, the preliminary experiment results obtained using *MRTeAm* cannot be directly compared to the experiment results obtained using *MRComm*. This is because different scenarios were used for the *MRTeAm* results compared to the *MRComm* results. However, examining the statistical significance when using increased *SPL* perturbation, there are noticeable improvements in how shared messages are communicated. Moreover, the *MRComm* results use the extended performance metrics, which more accurately depict communication performance within multi-robot coordination.

Chapter 5 and 6 present simulated and physical robot experiment results, respectively, and they are comparable as a result of using the same scenario, having the same limitations placed on the network and performing the same experiment configurations. It should be noted that there are negligible differences in some of the performance metrics when comparing the same experiments between Chapters 5 and 6. Furthermore, it can be argued that these are caused by the unpredictable nature of the noise experienced in the physical environment and network. Another possible cause is because in Chapter 5 more repeat experiments were conducted than in Chapter 6. However, it is common to run more experiment iterations in a simulated environment than with physical robots, typically due to comparative amount of time taken develop a software framework that works interchangeably, to setup and run a physical system and finally the wear-and-tear on a physical platform. The imperative results in both the simulated and physical experiments are those that represent communication-centric performance metrics, i.e *Average Failed tasks* and *Successful Missions*. These crucial results are similar, proving that continuous communication is possible and desirable in both simulated and physical experiments when using the *MRComm* framework and the communication-aware *LF* behaviour.

7.3 Future Work and Improvements

MRComm Improvements

As noted in Chapter 6, occasionally a multi-robot coordination mission failed and the cause was identified to be the *AON* messaging function. A requirement of *AON*, which needs to be satisfied, can cause an infinite loop (deadlock) in the system. The requirement is that robot(s) stop whatever action they are performing and begin communicating (re-sending) the previously received message until the entire team has agreed that they have received it. The particular cause is that the final robot(s) to confirm that the rest of the team has received the message may experience a long delay, as a result of increased network perturbations. Therefore, it can be deduced that the last robot was still sending out the previous message while another robot may have already started sending a new message and awaiting confirmation on that. This causes two different messages to enter the *AON* messaging pipeline with no way for the team to determine which is the correct message they need to agree on, causing the deadlock. A way to mitigate this issue has already been determined. The root cause of this issue is that message order is not kept. In future improvements, I propose to use a message time-stamp or to add a simple internal message counter that stamps each message before it is sent. In this way, a robot that receives a message can determine if it is in fact a new message or a previously received message and it can then choose the correct message and disregard the incorrect one.

As mentioned in Chapter 6, the physical robots suffer from micro-collisions. To remedy this, I could employ a method that records actual collisions and false-positive collisions. To record actual collisions, each time robots are within a certain proximity of each other, which qualifies as a near collision, this event will be time-stamped, and if the same robots have multiple time-stamps all occurring within the same (or marginally different) time frame they will be flagged as false-positive near collisions.

As noted in the results of Chapters 5 and 6, experiments using *LF* do not favour that well in all performance metrics, especially in *execution phase time* (duration of mission) and *distance travelled*. It is not possible to completely mitigate the performance disparity between the *NB* and *LF* behaviours, as their purpose is contradictory and *NB* always has the robots taking the quickest route to tasks while *LF* focuses on communication and tries to group robots together to maintain it. A simple solution to improve performance for those particular metrics would be to adjust the current utility function, which is used to decide which robot is leader next time a disconnect occurs. Currently the utility function used tries to choose the robot with the closest task, but also one which was not picked previously as a leader. This was to try to make robots less biased and spread their task completion evenly throughout the map. However, as noted in Chapters 5 and 6 this is not always the case and some tasks always get completed first. Therefore, the utility function could be adjusted to use a greedy algorithm, such as Dijkstra's algorithm or Huffman encoding etc., or some other algorithm, and analysed to see which of these improves performance most. Moreover, further improvements could be made to decrease mission completion time by using a new utility function and message. Chapter 6 demonstrates that robots remember the "last known" team member position, even when team members are no longer within communication distance, and they would try to return to that "last known" team position, which will get them back within communication distance. Therefore, the new utility function could be used once a leader has been chosen. It could then calculate, for each robot, the closest task and if a task is within a certain "limited" distance it will send a message to the rest of the robots (including the leader) to wait in their positions. Even if the robot(s) leave the threshold of communication they will promptly return to the "last known" team position and send all new messages to team members and then continue their mission. The new message would be used to initialise this new behaviour and utility function.

Future Work

As mentioned in the above section, a new utility function could be used to pause movement and allow for robots to complete additional task. This is very similar to the *RPS* strategy presented in the work in [54]. However, robots would continuously move toward their assigned tasks while periodically employing the strategy to regroup, share information and continue with the mission. Such a hybrid communication-aware behaviour can be greatly optimised for both communication performance and mission time, therefore should outperform a multi-robot strategy purely based on *RPS*. It would be beneficial to implement and evaluate the performance of some of the different strategies and algorithms proposed in [39] [54] and [40]. Even if the strategies do not improve performance, they may reveal some other aspect of communication or performance, which could supplement the existing algorithms and functionality of *LF*.

In future work, it would be beneficial to increase the number of behaviours that are communication-aware and with this improve the functionality of the network, such as introducing multi-hop routing of shared messages. Such functionality could allow robots to form a long chain (formation) to continue communication while performing distant tasks (e.g. search-and-rescue). Furthermore, it could allow central robots to act as bridges for communication between other distant robots. It would be valuable to analyse how such functionality could benefit different task compositions, for example from independent (*IT*), single robot (*SR*) tasks to multi-robot (*MR*) and constrained tasks (*CT*) etc.

Finally, it would be valuable to introduce a human-in-the-loop to the system. The human operative could be given multiple mission assignments that would benefit the performance and knowledge of themselves and the multi-robot team. A human operator could monitor the tasks received by the robot team and confirm which have been completed (i.e. the human would know what the robots know). Therefore, in the case that a robot is out of communication range and a task was not accounted for at the end of a mission, a human operator could force the team to continue the mission and go back, find the task and complete it again. Moreover, the operator could act as the new *"task assigner"* agent and assign which task is received by which robot, etc. This opens up a plethora of future possibilities in the broader research field of human-robot team coordination/interaction.

7.4 Summary

The work presented in this thesis examines the effect that network perturbations and network type have on multi-robot team communication, with the goal of developing a communication-aware behaviour that tries to guarantee continuous communication. The preliminary part of this work was to design an experiment and evaluate the empirical performance of simulated and physical robots when the most common type of network perturbation was employed. The outcome of these results was used in the design of more advanced and complex network parameters and the extension of the performance metrics, leading to the investigation of the variation of simulated and actual network signal quality. Moreover, this led to the development of a novel behaviour and a high-level messaging function that is used to facilitate continuous communication in a multi-robot team. The exhaustive results from both simulated and physical robot experiments show that, in most cases, continuous communication can be achieved for a multi-robot team, at the cost of some other decreased performance (e.g. increased mission execution time), however there is a significant communication-specific performance uplift.

Technology will always keep improving, multi-robot and human-robot collaboration will continue expanding, and if there is one *message* to summarise the outcome of this PhD, it is to *stay connected*!

Appendix A

Additional System Architecture

A.1 Original Finite State Machines

This section contains the original FSM that were generated by the ROS based SMACH software package. They contain all the states that the robots/agents could enter, including the transition actions. The first FSM is of the *task assigner* agent (Figure A.1), followed by the *NB task executioner* FSM (Figure A.2) and finally the *LF task executioner* FSM (Figure A.3). There are separate Leader (Figure A.4) and Follower FSMs (Figure A.5). For instance, green indicates an active state that the robot was in, during the image capture of the FSM, red indicates all the common termination states of the FSM and grey indicates dormant sub-FSM. However, the colours displayed in the Figures are unimportant here.

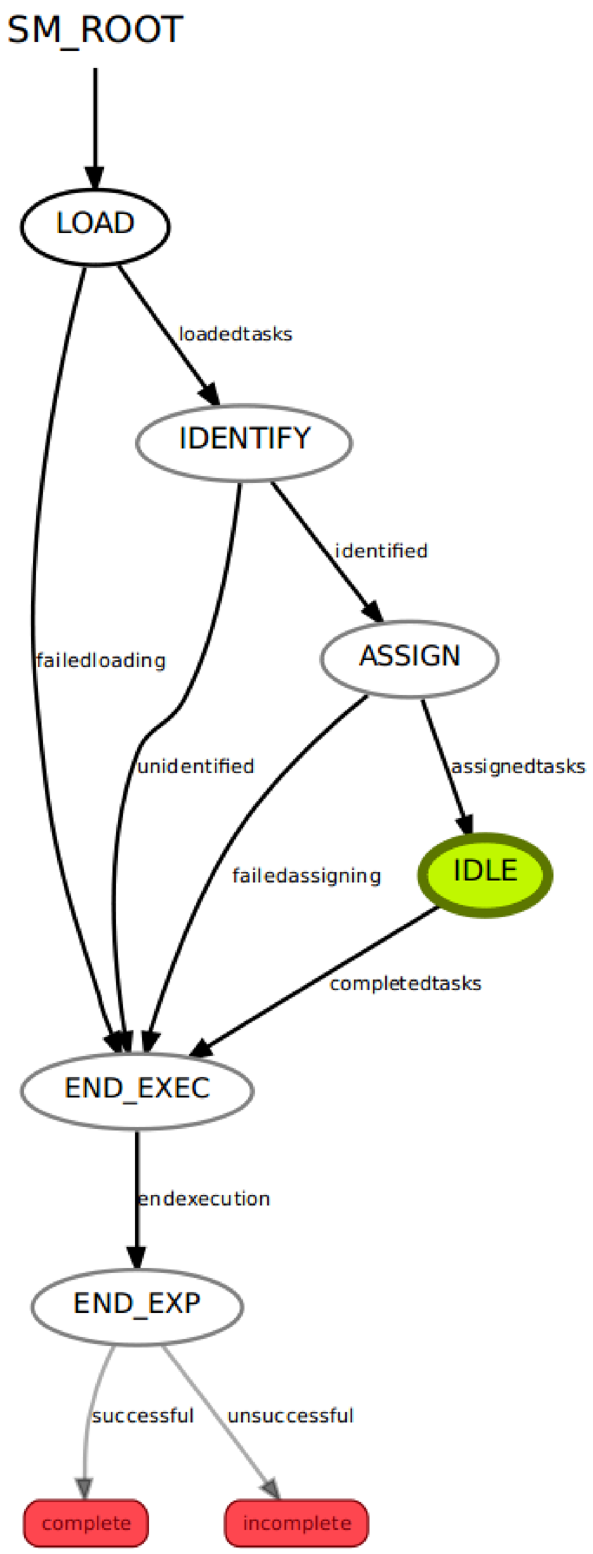


FIGURE A.1: Finite State Machine of the *task assigner* agent.

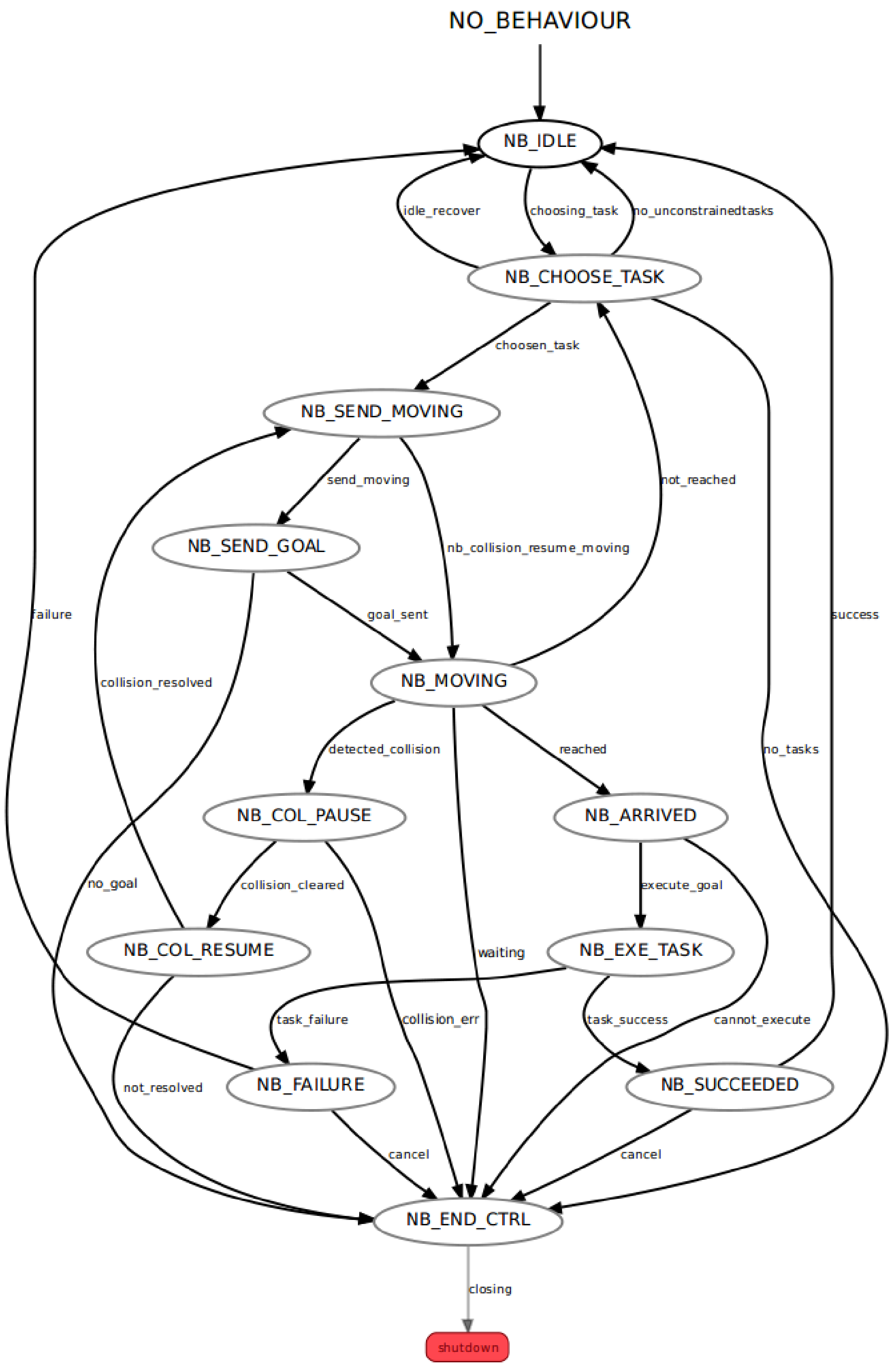


FIGURE A.2: Finite State Machine of a *task executioner* agent for *NB*.

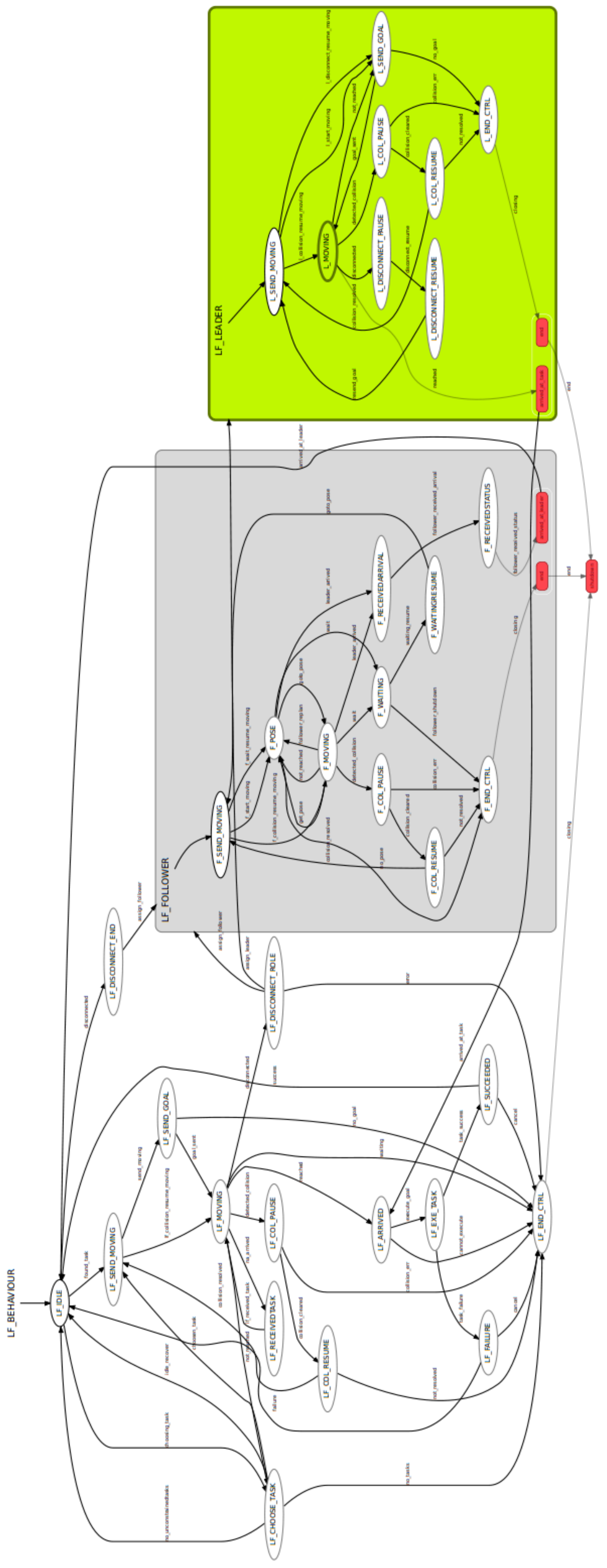


FIGURE A.3: Finite State Machine of a *task executioner* agent for *LF*.

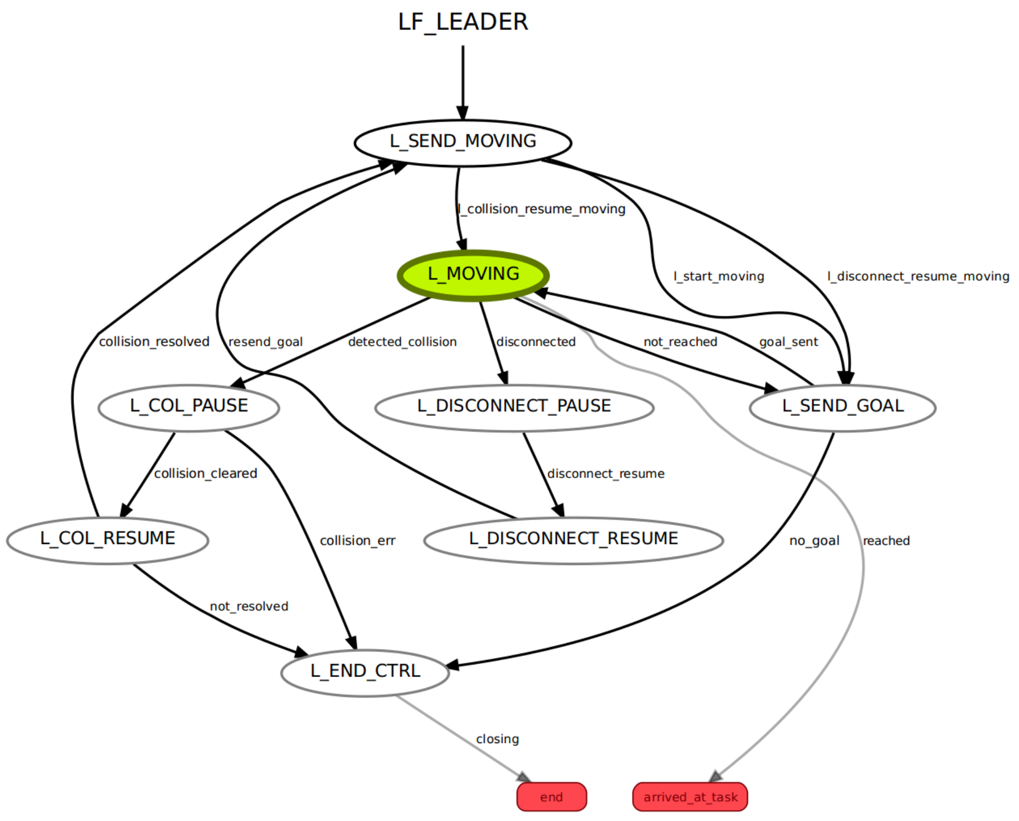


FIGURE A.4: Finite State Machine of the *task assigner* agent assigned the Leader role (*LF*).

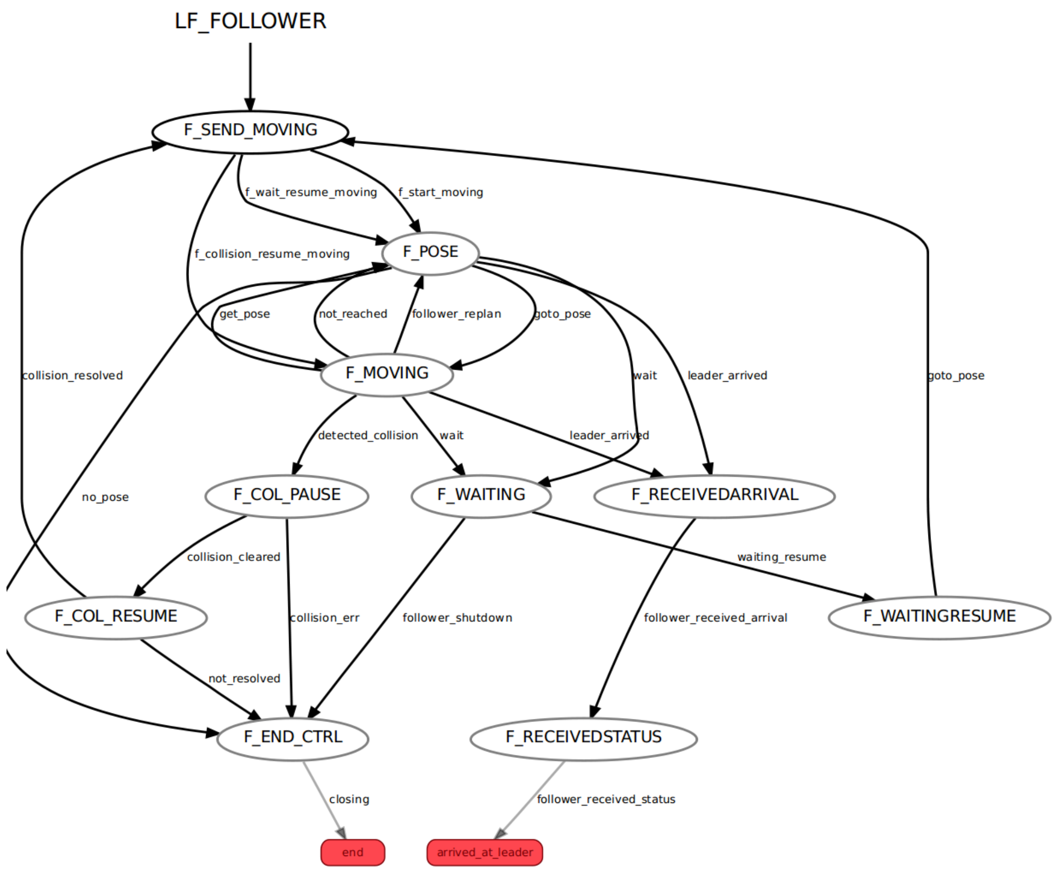
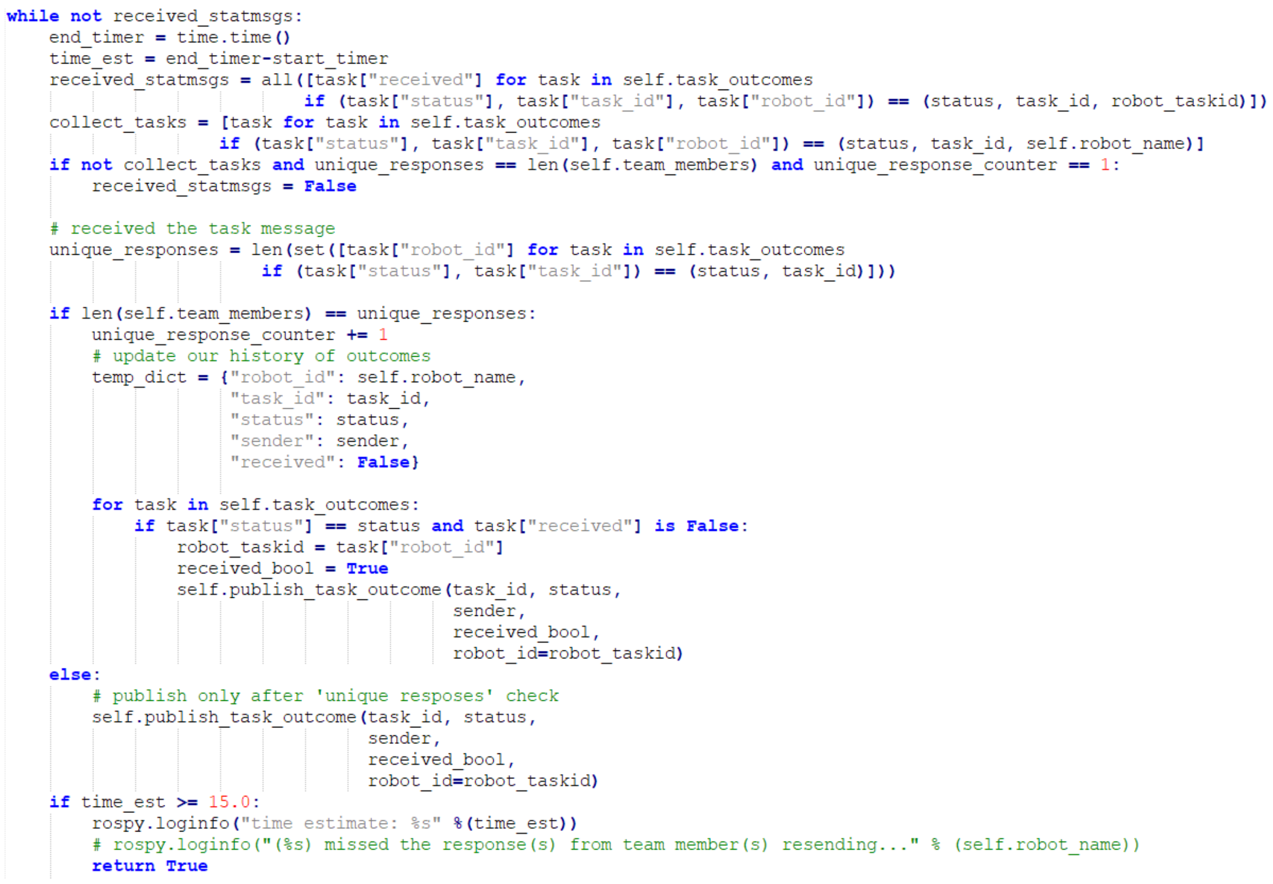


FIGURE A.5: Finite State Machine of the *task assigner* agent assigned the Follower role (*LF*).

A.2 Important Code Samples of Algorithms

This section contains partial code samples for some of the algorithms that are examined in the thesis. The first code snippet, Figure A.6 is on the AON messaging function. The second code snippet, Figure A.7, shows a portion of the functionality of *LF*, specifically the role assignment procedure.



```
while not received_statmsgs:
    end_timer = time.time()
    time_est = end_timer-start_timer
    received_statmsgs = all([task["received"] for task in self.task_outcomes
                            if (task["status"], task["task_id"], task["robot_id"]) == (status, task_id, robot_taskid)])
    collect_tasks = [task for task in self.task_outcomes
                    if (task["status"], task["task_id"], task["robot_id"]) == (status, task_id, self.robot_name)]
    if not collect_tasks and unique_responses == len(self.team_members) and unique_response_counter == 1:
        received_statmsgs = False

    # received the task message
    unique_responses = len(set([task["robot_id"] for task in self.task_outcomes
                        if (task["status"], task["task_id"]) == (status, task_id)]))

    if len(self.team_members) == unique_responses:
        unique_response_counter += 1
        # update our history of outcomes
        temp_dict = {"robot_id": self.robot_name,
                     "task_id": task_id,
                     "status": status,
                     "sender": sender,
                     "received": False}

        for task in self.task_outcomes:
            if task["status"] == status and task["received"] is False:
                robot_taskid = task["robot_id"]
                received_bool = True
                self.publish_task_outcome(task_id, status,
                                          sender,
                                          received_bool,
                                          robot_id=robot_taskid)
    else:
        # publish only after 'unique resposes' check
        self.publish_task_outcome(task_id, status,
                                  sender,
                                  received_bool,
                                  robot_id=robot_taskid)
    if time_est >= 15.0:
        rospy.loginfo("time estimate: %s" %(time_est))
        # rospy.loginfo("(%s) missed the response(s) from team member(s) resending..." % (self.robot_name))
        return True
```

FIGURE A.6: Code snippet showing partially the functionality of AON.

```
score_ready = role_ready = False
util_collected_scores = False
roles_collected = False
rospy.loginfo("({0}): Beginning to assign roles...".format(self.get_rname()))
# Force the robot to stop moving...
if self.robot_ctrl.aclient.get_state() == ACTIVE:
    self.robot_ctrl.aclient.cancel_goal()
    # Clear costmaps just in case
    self.robot_ctrl.clear_map_layers()
# Mark that team is about to disconnect to assign roles (end NA_MOVING)
self.robot_ctrl.publish_task_status(self.robot_ctrl.current_task.task_id,
                                    mrta.msg.TaskStatus.DISCONNECT_ROLE)
time.sleep(0.50)  # Sleep longer (allow time before re-publishing)
# If utility_cost returns None then exit disconnect_role with error...
if not self.utility_cost():
    rospy.loginfo("Couldn't calculate UTILITY COST...")
    return False
# While a robot isn't ready continue waiting
while not util_collected_scores:
    util_collected_scores = all([ready["ready"] for ready in self.robot_ctrl.util_scores])
    util_tasks = [ready for ready in self.robot_ctrl.util_scores]
    if not util_tasks:
        util_collected_scores = False
    # Publish (same score) until all member scores collected
    rospy.Rate(4).sleep()
    if len(self.robot_ctrl.team_members) == len(self.robot_ctrl.util_scores):
        score_ready = True
    # prevent deadlock (i.e. if task outcome msgs arrive)
    if self.robot_ctrl.received_msg:
        rospy.loginfo("RECEIVED ANOTHER MESSAGE IN DISC_ROLE...")
        self.set_cstate("lf_disc_role")
        return True
    self.robot_ctrl.publish_utility_cost(self.util_score, score_ready)
# If assign_role returns None then exit disconnect_role with error...
if not self.assign_role():
    rospy.loginfo("Couldn't ASSIGN ROLE...")
    return False
# Wait for the whole team to publish their roles
while not roles_collected:
    roles_collected = all([ready["ready"] for ready in self.robot_ctrl.roles])
    role_tasks = [ready for ready in self.robot_ctrl.roles]
    if not role_tasks:
        roles_collected = False
    # Publish (same role) until all member roles are collected
    rospy.Rate(4).sleep()
    if len(self.robot_ctrl.team_members) == len(self.robot_ctrl.roles):
        role_ready = True
    # prevent deadlock (i.e. if task outcome msgs arrive)
    if self.robot_ctrl.received_msg:
        rospy.loginfo("RECEIVED ANOTHER MESSAGE IN DISC_ROLE...")
        self.set_cstate("lf_disc_role")
        return True
    self.robot_ctrl.publish_role(self.role, role_ready)
```

FIGURE A.7: Code snippet showing partially the functionality of the role assignment procedure for *LF*.

Appendix B

Additional Results

B.1 Trajectory Drawings

Here random experiment recordings were chosen, one using *NB* (Figure B.1) and the other *LF* (Figure B.2), and their trajectories were translated and mapped onto a floor plan of the operational environment. Unfortunately, for *LF* in Figure B.2 the robots repeated many paths while following the leader, hence the map is busier.

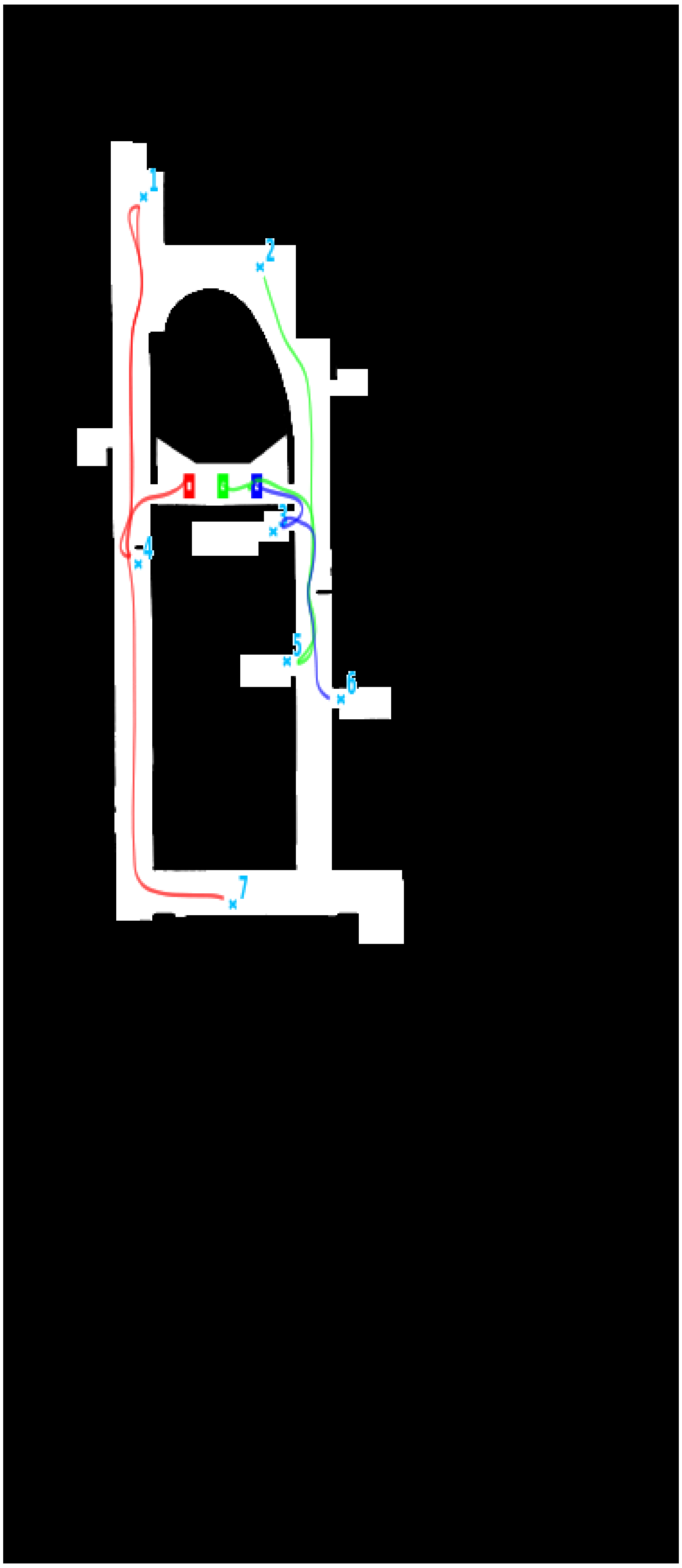


FIGURE B.1: Trajectory map of the *NB* behaviour in the operational environment.

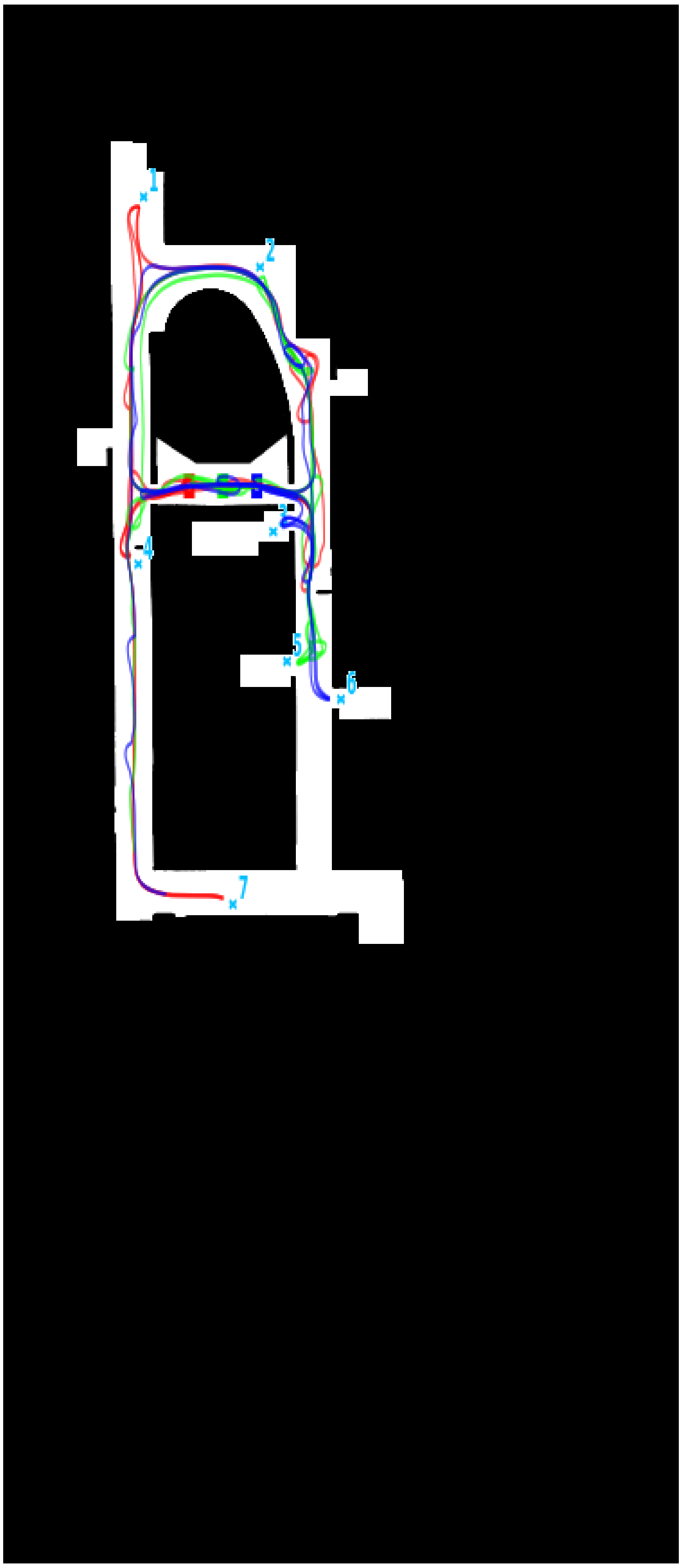


FIGURE B.2: Trajectory map of the *LF* behaviour in the operational environment.

B.2 Pseudo-random Generator

Figure B.3 is a simple plot showing the pseudo-randomness of a general purpose computer. A random value is selected 10,000 times from a range of integers 1-10 and the results are plotted in Figure B.3.

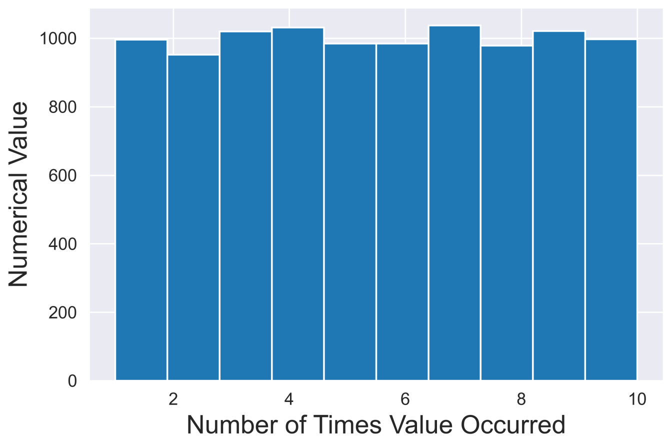


FIGURE B.3: Pseudo-randomness achieved by a computer.

B.3 Task Messages

Task messages are laid out and represented in a certain format in ROS, read from YAML files. The basic composition of messages sent to robots is described below.

```
# task_id:          A unique identifier.
# type:             Hypothetical 'SENSOR_SWEEP' for now,
# arrival_time:   Time (seconds) after start of an experiment
                              that the task "appears"
# location:         x,y coordinates of the task's location
# num_robots: Number of robots required to complete the task
# duration:       Time (seconds) required to complete the task
# depends:        Array of tasks (task_ids)
                              that this task depends on
```

```
# example

task_id: '1'
  type: SENSOR_SWEEP
  location:
    x: 5.2
    y: 50.6
  arrival_time: 0
  num_robots: 1
  duration: 0
  depends: []
```

Bibliography

[1] Robert Bogue. "Growth in e-commerce boosts innovation in the warehouse robot market". In: *Industrial Robot* 43.6 (2016), pp. 583–587. DOI: 10.1108/IR-07-2016-0194.

[2] Yugang Liu and Goldie Nejat. "Robotic Urban Search and Rescue: A Survey from the Control Perspective". In: *Journal of Intelligent & Robotic Systems* 72.2 (2013), pp. 147–165. ISSN: 1573-0409. DOI: 10.1007/s10846-013-9822-x.

[3] Boonsri Dickinson. *Robots to the rescue: searching for survivors, checking on structural damage in Japan*. 2011. URL: https://www.zdnet.com/article/robots-to-the-rescue-searching-for-survivors-checking-on-structural-damage-in-japan (visited on 03/14/2011).

[4] Ken Goldberg and Ben Kehoe. *Cloud Robotics and Automation: A Survey of Related Work*. Tech. rep. UCB/EECS-2013-5. California, United States: Electrical Engineering and Computer Sciences University of California at Berkeley, 2013. URL: https://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2013-5.pdf.

[5] Eugene Kim. *Amazon is now using a whole lot more of the robots from the company it bought for $775 million*. 2015. URL: https://www.businessinsider.com/amazon-doubled-the-number-of-kiva-robots-2015-10 (visited on 10/22/2015).

[6] Jon Fingas. *Starship's delivery robots now serve Purdue University*. 2019. URL: https://www.engadget.com/2019/09/09/starship-delivery-robots-at-purdue-university (visited on 09/09/2019).

[7] The Explorer. *Autonomous robots herald the future of farming*. 2019. URL: https://www.theexplorer.no/solutions/autonomous-robots-herald-the-future-of-farming.

[8] L. Grimstad and P. J. From. "Software components of the Thorvald II modular robot". In: vol. 39. 2018, pp. 157–165. DOI: 10.4173/mic.2018.3.2.

[9] Eric Schneider. "Mechanism Selection for Multi-Robot Task Allocation". PhD thesis. University of Liverpool, 2018.

[10] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[11] Sergi Hernandez Juan and Fernando Herrero Cotarelo. *Technical Report: Multi-master ROS systems*. Tech. rep. IRI-TR-15-1. Barcelona, Spain: Institut de Robòtica i Informàtica Industrial (IRI), 2015, p. 16. URL: http://www.iri.upc.edu/files/scidoc/1607-Multi-master-ROS-systems.pdf.

[12] Ifiok Otung. *Communication Engineering Principles*. New York, United States: PALGRAVE, 2001. ISBN: 0-333-77522-8.

[13] Harald T Friis. "A note on a simple transmission formula". In: *Proceedings of the IRE* 34.5 (1946), pp. 254–256.

[14] S. Sendra et al. "IEEE 802.11a/b/g/n Indoor Coverage and Performance Comparison". In: *6th International Conference on Wireless and Mobile Communications*. 2010, pp. 185–190. DOI: 10.1109/ICWMC.2010.46.

[15] E. Lau and W. Chung. “Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments”. In: *International Conference on Convergence Information Technology (ICCIT)*. 2007, pp. 1213–1218. DOI: 10.1109/ICCIT.2007.253.

[16] Hristo D. Hristov. *Fresnal Zones in Wireless Links, Zone Plate Lenses and Antennas*. 1st. Norwood, MA, USA: Artech House, Inc., 2000. ISBN: 0890068496.

[17] Chwan-Hwa (John) Wu and J. David Irwin. *Introduction to Computer Networks and Cybersecurity*. Florida, United States: CRC, Taylor and Francis Group, 2013. ISBN: 978-1-4665-7214-0.

[18] Jay Prakash, Deepak Kumar Gupta, and Rakesh Kumar. “Soft Computing Based Cluster-Head Selection in Mobile Ad-Hoc Network”. In: *Journal of Artificial Intelligence*. 2017, pp. 98–111. DOI: 10.3923/jai.2017.98.111.

[19] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd, Global Edition. Essex, England: Pearson Education Limited, 2016. ISBN: 9780136042594.

[20] Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Massachusetts, USA: MIT Press, 1999. ISBN: 0262232030.

[21] L. Vig and J. A. Adams. “Multi-robot coalition formation”. In: *IEEE Transactions on Robotics* 22.4 (2006), pp. 637–649. DOI: 10.1109/TRO.2006.878948.

[22] Bo Chen, Harry H. Cheng, and Joe Palen. “Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems”. In: *Transportation Research Part C: Emerging Technologies* 17.1 (2009), pp. 1 –10. ISSN: 0968-090X. DOI: 10.1016/j.trc.2008.04.003.

[23] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. “A Survey and Analysis of Multi-Robot Coordination”. In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399. DOI: 10.5772/57313.

[24] B. Gerkey, R. T. Vaughan, and A. Howard. “The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems”. In: *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*. 2003, pp. 317–323.

[25] E.I. Sklar et al. “A Framework in which Robots and Humans Help Each Other”. In: *Help Me Help You: Bridging the Gaps in Human-Agent Collaboration, Papers from the 2011 AAAI Spring Symposium*. 2011.

[26] Vladimir Zadorozhny and Michael Lewis. “Information fusion for USAR operations based on crowdsourcing”. In: *16th International conference on information Fusion (FUSION)*. 2013, pp. 1450–1457.

[27] Marin Lujak et al. “Towards Robots-Assisted Ambient Intelligence”. In: *5th International Conference on Multi-Agent Systems and Agreement Technologies*. Ed. by Francesco Belardinelli and Estefanía Argente. Springer International Publishing, 2018, pp. 490–497. ISBN: 978-3-030-01713-2. DOI: 10.1007/978-3-030-01713-2_34.

[28] Zieliński C. et al. “Rubik’s cube as a benchmark validating MRROC++ as an implementation tool for service robot control systems”. In: *Industrial Robot: An International Journal,Emerald Group Publishing* 34.5 (2007), pp. 368 –375. ISSN: 0143-991X. DOI: 10.1108/01439910710774377.

[29] Cynthia Breazeal. “Emotion and sociable humanoid robots”. In: *International Journal of Human-Computer Studies* 59.1 (2003), pp. 119 – 155. ISSN: 1071-5819. DOI: doi.org/10.1016/S1071-5819(03)00018-1.

[30] Elizabeth Gibney. “Google AI algorithm masters ancient game of Go”. In: *Nature* 529 (2016), pp. 1476–4687. DOI: 10.1038/529445a.

[31] Sklar E., Eguchi A., and Johnson J. "RoboCupJunior: Learning with Educational Robotics." In: *In: RoboCup 2002: Robot Soccer World Cup VI. Springer, Berlin, Heidelberg* 2752 (2002). Applications of Affective Computing in Human-Computer Interaction.

[32] Torsten Andre, Daniel Neuhold, and Christian Bettstetter. "Coordinated multi-robot exploration: Out of the box packages for ROS". In: *IEEE Globecom Workshops (GC Workshops)*. 2014, pp. 1457–1462. DOI: 10.1109/GLOCOMW.2014.7063639.

[33] Dieter Fox et al. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*. 1999.

[34] Sebastian Thrun et al. "Robust Monte Carlo localization for mobile robots". In: *Artificial Intelligence* 128.1 (2001), pp. 99 –141. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(01)00069-8.

[35] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics Automation Magazine* 4.1 (1997), pp. 23–33. ISSN: 1558-223X. DOI: 10.1109/100.580977.

[36] Mary Koes, Illah Nourbakhsh, and Katia Sycara. "Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints". In: *Proceedings of the 20th National Conference on Artificial Intelligence*. AAAI Press, 2005, pp. 1292–1297. ISBN: 1-57735-236-x.

[37] E. G. Jones et al. "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2006, pp. 570–575. DOI: 10.1109/ROBOT.2006.1641771.

[38] Tucker Balch. “The impact of diversity on performance in multi-robot foraging”. In: *Proceedings of the 3rd International Conference on Autonomous Agents (AGENTS)*. ACM Press, 1999. DOI: 10.1145/301136.301170.

[39] Tucker Balch and Ronald C. Arkin. “Behaviour-Based Formation Control for Multirobot Teams”. In: 14.6 (1998), pp. 926–939. ISSN: 2374-958X. DOI: 10.1109/70.736776.

[40] Yan Meng, Jeffrey V. Nickerson, and Jing Gan. “Multi-robot Aggregation Strategies with Limited Communication”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 2691–2696. DOI: 10.1109/IROS.2006.281991.

[41] Elizabeth A. Jensen, Ernesto Nunes, and Maria Gini. “Communication-Restricted Exploration for Robot Teams”. In: *Workshops at the 28th AAAI Conference on Artificial Intelligence*. AAAI Publications, 2014.

[42] Tyler Gunn and John Anderson. “Dynamic Heterogeneous Team Formation for Robotic Urban Search and Rescue”. In: *Journal of Computer and System Sciences*. Vol. 81. 3. 2015, pp. 553 –567. DOI: 10.1016/j.jcss.2014.11.009.

[43] B. P. Gerkey and M. J Mataríc. “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems”. In: *The International Journal of Robotics Research* 23.9 (2004), pp. 939–954. DOI: 10.1177/0278364904045564.

[44] David Landén, Fredrik Heintz, and Patrick Doherty. “Complex Task Allocation in Mixed-Initiative Delegation: A UAV Case Study”. In: *Principles and Practice of Multi-Agent Systems*. Ed. by Nirmit Desai, Alan Liu, and Michael Winikoff. Springer Berlin Heidelberg, 2012, pp. 288–303. ISBN: 978-3-642-25920-3. DOI: 10.1007/978-3-642-25920-3_20.

[45] Eric Schneider et al. “Auction-Based Task Allocation for Multi-robot Teams in Dynamic Environments”. In: *Towards Autonomous Robotic Systems*. Ed. by Clare Dixon and Karl Tuyls. Springer International Publishing, 2015, pp. 246–257. ISBN: 978-3-319-22416-9. DOI: 10.1007/978-3-319-22416-9_29.

[46] Eric Schneider, Elizabeth I. Sklar, and Simon Parsons. “Evaluating Multi-Robot Teamwork in Parameterised Environments”. In: *Towards Autonomous Robotic Systems*. Ed. by Lyuba Alboul, Dana Damian, and Jonathan M. Aitken. Springer International Publishing, 2016, pp. 301–313. ISBN: 978-3-319-40379-3. DOI: 10.1007/978-3-319-40379-3_32.

[47] Eric Schneider, Elizabeth I. Sklar, and Simon Parsons. “Mechanism Selection for Multi-Robot Task Allocation”. In: *Towards Autonomous Robotic Systems*. Ed. by Yang Gao et al. Springer International Publishing, 2017, pp. 421–435. ISBN: 978-3-319-64107-2. DOI: 10.1007/978-3-319-64107-2_33.

[48] Craig W. Reynolds. “Flocks, Herds and Schools: A Distributed Behavioral Model”. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH. ACM, 1987, pp. 25–34. ISBN: 0-89791-227-6. DOI: 10.1145/37401.37406.

[49] L. E. Parker. “ALLIANCE: An Architecture for Fault Tolerant Multi-robot Cooperation”. In: *IEEE Transactions on Robotics and Automation* 14.2 (1998), pp. 220–240. ISSN: 2374-958X. DOI: 10.1109/70.681242.

[50] C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos. “Safe decentralized and reconfigurable multi-agent control with guaranteed convergence”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 267–272. DOI: 10.1109/ICRA.2017.7989034.

[51] Douglas Vail and Manuela Veloso. “Multi-Robot Dynamic Role Assignment and Coordination Through Shared Potential Fields”. In:

Multi-Robot Systems. Ed. by A. Schultz, L. Parker, and F. Schneider. Kluwer, 2003.

[52] Jonathan Ian Timmis et al. "An Immune-Inspired Swarm Aggregation Algorithm for Self-Healing Swarm Robotic System". In: *Biosystems* 146 (2016), pp. 60–76. ISSN: 0303-2647. DOI: 10.1016/j.biosystems.2016.04.001.

[53] Bastian Broecker et al. "Hybrid Insect-Inspired Multi-Robot Coverage in Complex Environments". In: *Towards Autonomous Robotic Systems*. Ed. by Clare Dixon and Karl Tuyls. Springer International Publishing, 2015, pp. 56–68. ISBN: 978-3-319-22416-9. DOI: 10.1007/978-3-319-22416-9_8.

[54] Toru Takahashi, Yasuhiko Kitamura, and Hiroyoshi Miwa. "Organizing Rescue Agents Using Ad-hoc Networks". In: *Highlights on Practical Applications of Agents and Multi-Agent Systems. Advances in Intelligent and Soft Computing (AINSC)*. Vol. 156. Springer, Berlin, Heidelberg, 2012. DOI: 10.1007/978-3-642-28762-6_17.

[55] Sadat Chowdhury and Elizabeth I. Sklar. "Investigating the Impact of Communication Quality on Evolving Populations of Artificial Life Agents". In: *Artificial Life Conference Proceedings* 27 (2015), pp. 546–553. DOI: 10.1162/978-0-262-33027-5-ch096.

[56] M. Veloso and P. Stone. "Individual and collaborative behaviors in a team of homogeneous robotic soccer agents". In: *Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160)*. 1998, pp. 309–316. DOI: 10.1109/ICMAS.1998.699074.

[57] Wei Ren and Nathan Sorensen. "Distributed coordination architecture for multi-robot formation control". In: *Robotics and Autonomous Systems* 56.4 (2008), pp. 324 –333. ISSN: 0921-8890. DOI: 10.1016/j.robot.2007.08.005.

[58] Gautham Das. “Task Allocation Strategies for Multi-Robot Coordination”. PhD thesis. Ulster University, 2015.

[59] N. Ando et al. “RT-middleware: distributed component middleware for RT (robot technology)”. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 3933–3938.

[60] Y. Cao et al. “An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination”. In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 427–438.

[61] L. Sabattini et al. “Distributed Control of Multirobot Systems With Global Connectivity Maintenance”. In: *IEEE Transactions on Robotics* 29.5 (2013), pp. 1326–1332. ISSN: 1941-0468. DOI: 10.1109/TRO.2013.2267971.

[62] A. Mannucci, L. Pallottino, and F. Pecora. “Provably Safe Multi-Robot Coordination With Unreliable Communication”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3232–3239. ISSN: 2377-3774. DOI: 10.1109/LRA.2019.2924849.

[63] A. Ghaffarkhah and Y. Mostofi. “Communication-Aware Motion Planning in Mobile Networks”. In: *IEEE Transactions on Automatic Control* 56.10 (2011), pp. 2478–2485. ISSN: 2334-3303. DOI: 10.1109/TAC.2011.2164033.

[64] Y. Kantaros and M. M. Zavlanos. “Global Planning for Multi-Robot Communication Networks in Complex Environments”. In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1045–1061. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2593045.

[65] Sergio Caccamo et al. “RCAMP: A resilient communication-aware motion planner for mobile robots with autonomous repair of wireless connectivity”. In: *IEEE/RSJ International Conference on Intelligent*

Robots and Systems (IROS). 2017, pp. 2010–2017. DOI: 10.1109/IROS.2017.8206020.

[66] Gianluca Antonelli et al. "A Self-Configuring MANET for Coverage Area Adaptation through Kinematic Control of a Platoon of Mobile Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 1332–1337. DOI: 10.1109/IROS.2005.1545160.

[67] U. Witkowski et al. "Ad-hoc Network Communication Infrastructure for Multi-robot Systems in Disaster Scenarios". In: *IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance*. 2008.

[68] Mahbubur Rahman et al. "Relay Vehicle Formations for Optimizing Communication Quality in Robot Networks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6633–6639. DOI: 10.1109/IROS.2017.8206577.

[69] Zendai Kashino, Goldie Nejat, and Beno Benhabib. "A Multi-Robot Sensor-Delivery Planning Strategy for Static-Sensor Networks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6640–6647. DOI: 10.1109/IROS.2017.8206578.

[70] Joshua Reich and Elizabeth Sklar. "Robot-Sensor Networks for Search and Rescue". In: *In Proceedings IEEE International Workshop on Safety, Security and Rescue Robotics*. 2006.

[71] I.Rekleitis et al. "Limited communication, multi-robot team based coverage". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 2004, pp. 3462–3468. DOI: 10.1109/ROBOT.2004.1308789.

[72] Mikko Lauri, Eero Heinänen, and Simone Frintrop. "Multi-Robot Active Information Gathering with Periodic Communication". In:

IEEE International Conference on Robotics and Automation (ICRA). 2017, pp. 851–856. DOI: 10.1109/ICRA.2017.7989104.

[73] Iain Davidson and William Noble. "Tools and language in human evolution". In: *Tools, Language and Cognition in Human Evolution*. Ed. by K. R. Gibson and T. Ingold. Cambridge: Cambridge University Press, 1993, pp. 363–388.

[74] Doreen Kimura. *Neuromotor Mechanisms in Human Communication*. Oxford, United Kingdom: Oxford University Press, 1993.

[75] W. Tecumseh Fitch. *The Evolution of Language*. Cambridge, United Kingdom: Cambridge University Press, 2010.

[76] Sara Kiesler and Lee Sproull. "Group decision making and communication technology". In: *Organizational Behavior and Human Decision Processes* 52.1 (1992), pp. 96–123. ISSN: 0749-5978. DOI: 10.1016/0749-5978(92)90047-B.

[77] Malcolm R. Parks and Kory Floyd. "Making Friends in Cyberspace". In: *Journal of Computer-Mediated Communication* 1.4 (Mar. 1996). ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.1996.tb00176.x.

[78] Laura Garton, Caroline Haythornthwaite, and Barry Wellman. "Studying Online Social Networks". In: *Journal of Computer-Mediated Communication* 3.1 (June 1997). ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.1997.tb00062.x.

[79] Elizabeth Keating and Gene Mirus. "American Sign Language in virtual space: Interactions between deaf users of computer-mediated video communication and the impact of technology on language practices". In: *Language in Society* 32.5 (2003), 693–714. DOI: 10.1017/S0047404503325047.

[80] Cemil Oz and Ming C. Leu. “American Sign Language word recognition with a sensory glove using artificial neural networks”. In: *Engineering Applications of Artificial Intelligence* 24.7 (2011), pp. 1204–1213. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2011.06.015.

[81] Jeannette Sutton et al. “What it Takes to Get Passed On: Message Content, Style, and Structure as Predictors of Retransmission in the Boston Marathon Bombing Response”. In: 2015. DOI: 10.1371/journal.pone.0134452.

[82] Andrea H. Tapia, Nicolas LaLone, and Hyun-Woo Kim. “Run Amok: Group Crowd Participation in Identifying the Bomb and Bomber from the Boston Marathon Bombing”. In: *11th International ISCRAM conference proceedings*. Theme 06 - Ethical, Legal and Social Issues of IT Supported Emergency Response. 2014.

[83] Angel Ruiz-Zafra et al. “SUCRE: Supporting Users, Controllers and Responders in Emergencies”. In: *11th International ISCRAM conference proceedings*. Theme 05 - Decision Support Systems. 2014.

[84] Amro Al-Akkad et al. “Help Beacons: Design and Evaluation of an Ad-Hoc Lightweight S.O.S. System for Smartphones”. In: 2014, pp. 1485–1494. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557002.

[85] Amro Al-Akkad et al. “Tweeting When Online is Off? Opportunistically Creating Mobile Ad-hoc Networks in Response to Disrupted Infrastructure”. In: *11th International ISCRAM conference proceedings*. Theme 13 - Social Media in Crisis Response and Management. 2014.

[86] Robin R Murphy et al. “Interacting with Trapped Victims Using Robots”. In: *IEEE International Conference on Technologies for Homeland Security (HST)*. 2013, pp. 32–37. DOI: 10.1109/THS.2013.6698972.

[87] Tsvetan Zhivkov, Eric Schneider, and Elizabeth I. Sklar. "Measuring the Effects of Communication Quality on Multi-robot Team Performance". In: *Towards Autonomous Robotic Systems (TAROS) 18th Annual Conference*. Ed. by Yang Gao et al. Springer International Publishing, 2017, pp. 408–420. ISBN: 978-3-319-64107-2. DOI: 10.1007/978-3-319-64107-2_32.

[88] Tsvetan Zhivkov, Eric Schneider, and Elizabeth Sklar. "Establishing Continuous Communication through Dynamic Team Behaviour Switching". In: *Embedded Intelligence: Enabling & Supporting RAS Technologies (UK-RAS)*. 2019, pp. 83–86.

[89] Tsvetan Zhivkov, Eric Schneider, and Elizabeth Sklar. "MRComm: Multi-Robot Communication Testbed". In: *Towards Autonomous Robotic Systems (TAROS) 20th Annual Conference*. Ed. by Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang. Springer International Publishing, 2019, pp. 346–357. ISBN: 978-3-030-25332-5. DOI: 10.1007/978-3-030-25332-5_30.

[90] R. Braden. *Requirements for Internet Hosts – Communication Layers*. Standard RFC1122. Internet Engineering Task Force (IETF), 1989. URL: https://tools.ietf.org/html/rfc1122#page-20.

[91] Intel. *Intel Dual Band Wireless-AC 7265*. 2015. URL: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/dual-band-wireless-ac-7265-brief.pdf.

[92] David Rowan. "Kinect for Xbox 360: The inside story of Microsoft's secret 'Project Natal'". In: *Wired* (2010).

[93] Oxford University Press (OUP). *Switch*. Lexico.com, 2019.

[94] Sadat Chowdhury. "A Study of the Impact of Interaction Mechanisms and Population Diversity in Evolutionary Multiagent Systems". PhD thesis. The City University of New York, 2016.

[95] Stephen Hemminger. “Network Emulation with NetEm”. In: *Open Source Development Lab*. Open Source Development Lab, 2005.

[96] Makoto Matsumoto and Takuji Nishimura. “Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator”. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30. ISSN: 1049-3301. DOI: 10.1145/272991.272995.

[97] S. S. Shapiro and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* 52 (1965), pp. 591–611. ISSN: 00063444. DOI: 10.2307/2333709.

[98] H. B. Mann and D. R. Whitney. “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other”. In: vol. 18. 1. 1947, pp. 50–60. DOI: 10.1214/aoms/1177730491.

[99] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. New York: Academic Press, Inc., 1988.

[100] *Python library: Scipy v1.4.1*. https://docs.scipy.org/doc/scipy/reference/stats.html. 2019.

[101] S. Y. Seidel and T. S. Rappaport. “914 MHz path loss prediction models for indoor wireless communications in multifloored buildings”. In: *IEEE Transactions on Antennas and Propagation* 40.2 (1992), pp. 207–217. ISSN: 1558-2221. DOI: 10.1109/8.127405.

[102] Qian Dong and W. Dargie. “Evaluation of the reliability of RSSI for indoor localization”. In: *International Conference on Wireless Communications in Underground and Confined Areas*. 2012, pp. 1–6. DOI: 10.1109/ICWCUCA.2012.6402492.

[103] J. Ore, S. Elbaum, and C. Detweiler. “Dimensional inconsistencies in code and ROS messages: A study of 5.9M lines of code”. In: *IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS). 2017, pp. 712–718. DOI: 10.1109/IROS.2017.8202229.