The Detection Of Malicious Activities Within A Robotic Swarm

Ian Sargeant

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

2019

Declaration of Authorship

I, Ian Sargeant, hereby declare that these doctoral studies were conducted under the supervision of Doctor Allan Tomlinson.

The work presented in this thesis is the result of original research carried out by myself, whilst enrolled in the Department of Information Security as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Ian Sargeant
January 2018

Abstract

The philosophy behind a robotic swarm is that a large number of relatively basic robots will work in conjunction with each other, in order to complete a predetermined goal. Robotic swarm implementations have unique characteristics, taking significant inspiration from nature, and are being considered for multiple roles and tasks. However, the majority of the research assumes a benign operating environment for the robotic swarms.

This research initially considers how robotic swarms could be subject to various types of attack, that utilise the unique characteristics of the robotic swarms to an attacker's advantage. The research established that a robotic swarm could be maliciously manipulated and then considered how the robotic swarm could detect and then react to an attack.

This research considered two use-cases that demonstrated how different types of swarm could be attacked and how techniques, that utilised the unique characteristics of the robotic swarms, could successfully detect and contend with the attacks.

In order to conduct this research, taxonomies and generic models were developed. Simulations then utilised the models, in order to simulate attacks against the two use-cases. Intrusion Detection Systems (IDS) were then developed and modelled that utilised the unique characteristics of a robotic swarm.

This research determined that signature based IDS could be successfully utilised within robotic swarm and that anomaly based IDS were not suitable.

Simulations successfully demonstrated that signature based IDS, that utilised the unique characteristics of a swarm, could be undertaken in order to detect and then mitigate attacks against a robotic swarm.

Acknowledgments

I would like to take this opportunity to thank my supervisor, Dr. Allan Tomlinson, who has not only provided me with his advice, guidance, knowledge and expertise throughout this research but has also managed to keep my focus on the subject to hand.

I would also like to take this opportunity to thank my family. Especially my wife, Jen, and my children, Jack and Katie, whose support and encouragement has been considerable to me throughout this undertaking and should not be underestimated.

I would like to thank my work colleagues, who have also provide me with much needed support and encouragement at times. Thank you Darron, Steve, Richard, Colin and the teams.

Finally, I would like to acknowledge and thank my fellow coaches at Bredon Star RFU, who have always helped me out when I have not been able to attend because of this work. Drive on Star!

Publications

Sargeant I., Tomlinson A. Modelling Malicious Entities In A Robotic Swarm. *In IEEE/AIAA 32nd Digital Avionics Conference (DASC) 6th-10th October 2013. Pages 7B1-1 to 7B1-12.*

Sargeant I., Tomlinson A. Maliciously Manipulating a Robotic Swarm. *In Proceedings of the International Conference Embedded Systems, Cyber-physical Systems, & Applications (ESCS'16). Pages 122-128.*

Sargeant I., Tomlinson A. Review of Potential Attacks on Robotic Swarms. *In: Bi Y., Kapoor S., Bhatia R. (eds) Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016. IntelliSys 2016. Lecture Notes in Networks and Systems, vol 16. Springer, Cham.*

The following publication is to be presented at Intelligent Systems Conference (IntelliSys), September 2019.
Sargeant I., Tomlinson A. Undertaking Intrusion Detection Within Robotic Swarms. *In Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2019. IntelliSys 2019. Advances in Intelligent Systems and Computing. Springer.*

Symbols Used within this Thesis

The follow lists the major symbols used within this thesis:

S^o	- Original Swarm
s_i^o	- A Swarm Entity from within the Original Swarm
SA^o	- Attributes of the Original Swarm
RA^o	- Attributes of an Original Swarm Entity
S^m	- Malicious Swarm
s_i^m	- A Swarm Entity from within the Malicious Swarm
SA^m	- Attributes of the Malicious Swarm
RA^m	- Attributes of an Malicious Swarm Entity
S^d	- IDS Swarm
s_i^d	- A Swarm Entity from within the IDS Swarm
SA^d	- Attributes of the IDS Swarm
RA^d	- Attributes of an IDS Swarm Entity
E	- The Operating Environment for the Swarms
e_i	- Individual Elements of the Operating Environment
EA	- Attributes of the Operating Environment
c_i	- Condition
f	- Function

Contents

I Background 1

1 Introduction 3
1.1 Motivation . . . 4
1.2 Overview of Robotic Swarms . . . 4
1.2.1 Examples for the Proposed Uses of Swarms . . . 5
1.3 Statement of the Research Problem . . . 5
1.4 Contributions . . . 7
1.5 Structure of the Thesis . . . 8

2 Background 11
2.1 Introduction . . . 11
2.2 Emergence of Swarm Robotics . . . 11
2.2.1 Pre-Swarm Robotics . . . 11
2.2.2 The Term Swarm . . . 12
2.2.3 Swarm Intelligence . . . 12
2.2.4 Swarm Optimisation . . . 15
2.2.5 Swarm Engineering . . . 16
2.2.6 Swarm robotics . . . 16
2.3 Capabilities and Limitations of Swarm Entities . . . 16
2.4 Swarm Characteristics . . . 18
2.4.1 Autonomy . . . 18
2.4.2 Decentralised Control . . . 18
2.4.3 Large Number of Members . . . 19
2.4.4 Collective Emergent Behaviour and Swarm Intelligence . . . 20
2.4.5 Local Sensing . . . 20
2.4.6 Local Communications . . . 21
2.4.7 Scalability . . . 22
2.5 Taxonomy of Swarms . . . 23
2.5.1 Method-Based Taxonomy . . . 23
2.5.2 Collective Behaviour Based Taxonomy . . . 24
2.5.3 Swarm Behaviour Taxonomy . . . 24
2.5.4 Swarm Specifications and Attributes Taxonomy . . . 25
2.5.5 Proposed Taxonomy . . . 26
2.6 Methods of Robotic Swarm Operations . . . 29
2.6.1 Navigation . . . 29

2.6.2 Communication Methods . . . 31
2.6.3 Thresholds and Decision Making . . . 33
2.6.4 Self-Healing and Self-Reproduction . . . 34
2.7 Proposed Applications for Swarm Robotics . . . 36
2.7.1 Maintenance Tasks . . . 37
2.7.2 Communications Providers . . . 38
2.7.3 Emergency Response . . . 38
2.7.4 Use of Swarms in Space . . . 40
2.7.5 Military Communications . . . 41
2.7.6 Underwater Mine Countermeasures . . . 42
2.7.7 Landmine Detection . . . 43
2.7.8 Situational Awareness . . . 46
2.8 Discussion . . . 46
2.8.1 Capabilities and Limitations of Swarm Entities . . . 47
2.8.2 The Term Swarm . . . 48

II Contributions 49

3 Generic Models for Swarms 51
3.1 Introduction . . . 51
3.2 Generic Models . . . 52
3.2.1 Initial Swarm Definition . . . 52
3.2.2 Modelling a Malicious Swarm . . . 53
3.2.3 Modelling Swarm Interaction . . . 55
3.2.4 Modelling the Operating Environment . . . 56
3.3 Modelling Swarm Behaviour . . . 58
3.3.1 States Within a Swarm . . . 58
3.4 Modelling the States within Swarms . . . 61
3.4.1 Modelling the States of Swarm Entities . . . 61
3.4.2 Modelling the State of a Swarm . . . 62
3.5 Typical Use Cases for Swarm Applications . . . 64
3.5.1 Case 1 - Hunting Example . . . 64
3.5.2 Case 2 - Landmine Example . . . 64
3.5.3 Review of Case Examples . . . 66
3.6 Summary . . . 66

4 Taxonomy of Threats 67
4.1 Introduction . . . 67
4.2 Threat Modelling . . . 67
4.2.1 Denial of Service . . . 68
4.2.2 Masquerade . . . 69
4.2.3 System Penetration . . . 69
4.2.4 Authorisation Violation . . . 69

4.2.5 Planting . . . 70
4.2.6 Communications Monitoring (Eavesdropping) . . . 70
4.2.7 Modification of Data in Transit . . . 71
4.2.8 Misappropriation . . . 73
4.3 The Adversary . . . 73
4.3.1 Manner of Attack . . . 75
4.3.2 Attacker's Knowledge of a Swarm . . . 75
4.4 Attack Modelling . . . 76
4.4.1 Attacking Swarm Entities . . . 77
4.4.2 Attacking the Swarm . . . 77
4.5 Attack Methods . . . 78
4.5.1 Physical Tampering . . . 79
4.5.2 Software Attacks . . . 82
4.5.3 Attacks on Communications . . . 83
4.5.4 Incorrect Information Display . . . 90
4.5.5 Reconnaissance . . . 91
4.6 Discussion . . . 92
4.6.1 Case 1 - Cooperative Navigation (Hunting) . . . 93
4.6.2 Case 2 - Foraging and Local Recruitment (Landmine Locating) . . . 93

5 Simulating Attacks on Swarms 95
5.1 Introduction . . . 95
5.2 Case 1 - Cooperative Navigation . . . 96
5.2.1 Taxonomies . . . 96
5.2.2 Overview of the Previous Research . . . 97
5.2.3 Simulations . . . 104
5.2.4 Conclusion of Case 1 . . . 129
5.3 Case 2 - Foraging Techniques and Local Recruitment Schemes . . . 131
5.3.1 Taxonomies . . . 132
5.3.2 Overview of the Previous Research . . . 132
5.3.3 Simulations . . . 137
5.3.4 Conclusion of Case 2 . . . 144
5.4 Discussion . . . 145

6 Modelling Intrusion Detection Systems in Swarms 147
6.1 Introduction . . . 147
6.1.1 Overview and Background of Intrusion Detection Systems . . . 147
6.1.2 Systems Similar to Swarms . . . 148
6.2 Considering IDS for Swarms . . . 150
6.2.1 Background Assumptions for a Swarm Entity . . . 151
6.2.2 Intrusion Detection Methodologies within a Swarm . . . 151
6.3 Modelling Swarm IDS Techniques . . . 155
6.3.1 Modelling a Separate IDS Swarm . . . 156

6.4 Summary . . . 169

7 Simulations of Robotic Swarm Intrusion Detection Systems 171
7.1 Introduction . . . 171
7.2 Independent Swarm IDS . . . 171
7.2.1 Implementing the Straight Line IDS . . . 172
7.2.2 Implementing Stationary Detection IDS . . . 180
7.2.3 Combining the IDS Methods . . . 183
7.2.4 Independent IDS Simulation Results . . . 185
7.2.5 Conclusion of Independent IDS Swarm . . . 186
7.3 Internal Swarm IDS . . . 186
7.3.1 False Landmine Detection . . . 187
7.3.2 False Scent Generation Detection . . . 191
7.3.3 Additional Measures to Improve Performance . . . 192
7.3.4 Conclusion of Internal Swarm IDS . . . 193
7.4 Discussion . . . 194

III Conclusions 195

8 Discussion 197
8.1 Introduction . . . 197
8.2 Manipulating a Swarm . . . 197
8.2.1 Case 1 - Attacking Cooperative Navigation . . . 198
8.2.2 Case 2 - Attacking Foraging and Local Recruitment . . . 199
8.2.3 Summary of Attacking a Swarm . . . 200
8.3 IDS within a Swarm . . . 200
8.3.1 IDS within Cooperative Navigation . . . 200
8.3.2 IDS within Foraging and Local Recruitment . . . 201
8.3.3 Summary of IDS within a Swarm . . . 202
8.4 Future Work . . . 202

Annexes 209

A Intrusion Detection Systems 209
A.1 Introduction . . . 209
A.1.1 Overview and Background of Intrusion Detection Systems . . . 209
A.1.2 Background and history of Intrusion Detection Systems . . . 210
A.1.3 History of Computer Intrusion Detection Systems . . . 210
A.1.4 Categorising Intrusion Detection Systems . . . 212
A.2 Intrusion Detection System Technologies . . . 214
A.2.1 Network Based IDS . . . 214
A.2.2 Host Based IDS . . . 215

A.2.3 Wireless IDS 216
A.2.4 Typical IDS Components 218
A.3 Summary 218

B Characteristics of Current Robotic Swarm Implementations **219**

Bibliography **223**

List of Figures

2.1 Reynold's Rules for Synchronised Movements 14
2.2 Constraining a Boid's Variables 14
2.3 Method Based Taxonomy [24] 23
2.4 Collective Behaviour Based Taxonomy [24] 24
2.5 Swarm Behaviour Taxonomy [31] 25
2.6 Swarm Specifications and Attributes Taxonomy [60] 26
2.7 Response-Threshold Model [187] 27
2.8 Lennard-Jones Potential Function 28
2.9 Two Different Length Paths 29
2.10 Forced Return Path 30
2.11 Swarm Entity Edge Following Fixed Entities [158] 31
2.12 Swarm Gradient Formation [158] 31
2.13 Using Colour to Communicate and Indicate Orientation to Other Robots. 32
2.14 Short Range Local and Long Range Communications 32
2.15 Tracking a Contaminant Cloud 39
2.16 Underwater Swarm Movement 43
2.17 Local Scent Maximum 45

3.1 Venn Diagram Showing Relationship between the Attributes of the Original Swarm Entities and the Attributes of the Malicious Swarm Entities . . . 55
3.2 Modelling the Operating Environment 57
3.3 Influence of Swarm Entities 63

4.1 Modification of Communications at a Choke Point 71
4.2 No Modification of Communications 72
4.3 Swarm Searching for a Target 72
4.4 Target Located - Swarm Indicating Direction 72
4.5 Malicious Entity Presents False Information to Original Swarm 73
4.6 Single Jammer Attack on Communications 88
4.7 Multiple Jammers Attacking Communications 88
4.8 TCP Three Way Handshake [72] 89
4.9 SYN Attack 90

5.1 Messaging Within a Swarm 99
5.2 Information Distribution Prior to Join Up 101
5.3 Information Distribution After Join Up 101

5.4 Longer Path, Time: t . . . 102
5.5 More Efficient Path, Time: $t + n$ . . . 102
5.6 Operating Environment Mazes . . . 103
5.7 Searcher "Reflecting" Off Walls . . . 103
5.8 Baseline Results of Random Placements of Targets and Searchers . . . 106
5.9 Averaged Results for Fixed Separations . . . 107
5.10 Expanded View - Comparing Performance of NwS and NwR . . . 108
5.11 Two Targets - No Obstacles . . . 108
5.12 Two Targets - Obstacle in Operating Environment . . . 109
5.13 The Effects of Jamming on the Average Time to Locate a Target . . . 110
5.14 The Effects of Jamming by Comparing the Number of Malicious Entities 111
5.15 Linear Trend Line Showing the Effects of the Malicious Entities . . . 111
5.16 Impact on Searcher from One Random Walking Malicious Entity . . . 114
5.17 Principle of the Hunter Jammer . . . 115
5.18 Hunters Isolating Targets . . . 116
5.19 Average Amount of Searchers that Achieved their Goal: No Maze . . . 118
5.20 Average Amount of Searchers that Achieved their Goal: Simple Maze . . 118
5.21 No. of Malicious Entities and their Effect on Searcher Efficiency: No Maze119
5.22 No. of Malicious Entities and their Effect on Searcher Efficiency: Simple Maze . . . 119
5.23 Malicious Entity Hunting for a Target . . . 120
5.24 Malicious Entity Locates a Target . . . 121
5.25 Malicious Entity Moves Away from Target . . . 121
5.26 Malicious Entity Provides False Information . . . 122
5.27 Average Time to Locate a Target: No Maze . . . 123
5.28 Average Time to Locate a Target: Simple Maze . . . 123
5.29 Number of Malicious Entities to Effect Efficiency of Searchers: No Maze 124
5.30 Number of Malicious Entities to Effect Efficiency of Searchers: Simple Maze . . . 124
5.31 Average Percentage of Searchers that Achieved their Goal: No Maze . . 127
5.32 Average Percentage of Searchers that Achieved their Goal: Simple Maze 127
5.33 Number of Malicious Entities Required to Effect Efficiency of Searchers: No Maze . . . 128
5.34 Number of Malicious Entities Required to Effect Efficiency of Searchers: Simple Maze . . . 128
5.35 Comparing Efficiency of Attack Methods - No Maze . . . 130
5.36 Comparing Efficiency of Attack Methods - Simple Maze . . . 130
5.37 Comparing Efficiency of Attack Methods - Complex Maze . . . 131
5.38 Swarm Entity Operating States [112] . . . 133
5.39 Deterministic Foraging Movement . . . 134
5.40 Random Foraging Movement . . . 134
5.41 Cross-Section Through Scent Gradient . . . 135

5.42 3D Overview of Scent . . . 135
5.43 Cross-Section Through Local Scent Maximum . . . 136
5.44 3D Overview of Local Scent Maximum . . . 137
5.45 Baseline Results of Searching for Landmines: Deterministic Walk . . . 138
5.46 Baseline Results of Searching for Landmines: Random Walk . . . 139
5.47 Completion of Task to Locate Landmines - Deterministic Walk . . . 139
5.48 Completion of Task to Locate Landmines - Random Walk . . . 140
5.49 False Landmine Placements . . . 141
5.50 False Scent Locations . . . 142
5.51 Trapped Searcher . . . 143

6.1 IDS and Original Swarms . . . 156

7.1 IDS - Small Malicious Entity Configuration . . . 174
7.2 IDS - Medium Malicious Entity Configuration . . . 174
7.3 IDS - Large Malicious Entity Configuration . . . 175
7.4 Mobile IDS Implementation . . . 176
7.5 Kill All: Number of False-Positives for a Simple Maze with a Small Amount of Malicious Entities . . . 177
7.6 Kill All: Number of False-Positives for No Maze with a Large Amount of Malicious Entities . . . 177
7.7 Mobile IDS False Positive . . . 178
7.8 Kill Last: Number of False-Positives for a Simple Maze with a Small Amount of Malicious Entities . . . 179
7.9 Kill Last: Number of False-Positives for No Maze with a Large Amount of Malicious Entities . . . 179
7.10 Stationary IDS Implementation . . . 181
7.11 Average Times to Locate Malicious Entities . . . 182
7.12 False-Positives and Number of Independent Entities . . . 183
7.13 Searcher Entity and IDS Entity Navigate Towards Malicious Entity . . . 184
7.14 IDS Entities Located at the Target . . . 185
7.15 Extra IDS State . . . 187
7.16 Percentage Difference in Average Times for an IDS Enabled Swarm Compared to a Baseline Swarm . . . 189
7.17 Entities' States Within a Swarm Over Time . . . 190

8.1 Initial Swarm Conditions . . . 204
8.2 Confidences in Swarm Actions . . . 205
8.3 Entity with Weighted Confidence . . . 205

List of Tables

2.1 Comparison of Swarm Robotics and Other Systems [183] 46

3.1 Typical Attributes of a Swarm and a Malicious Intruder 66
3.2 Typical Attributes of the Swarm Entities 66

B.1 Kilobot Characteristics [99] 220
B.2 Colias Robot Characteristics [9, 191] 221
B.3 Mona Robot Characteristics [121] 221
B.4 Erratic Mobile Platform Characteristics [135, 147] 221
B.5 S-Bot Characteristics [181] 221
B.6 e-puck Characteristics [129] 222
B.7 e-puck2 Characteristics [71] 222

List of Algorithms

5.1 Communication-based navigation: the actions executed at each control step by each robot A 98

Part I

Background

1 Introduction

Much research has been undertaken in the field of swarm robotics and its potential applications. However, the majority of the previous research work assumes a benign operational environment for the robotic swarm. This work considers the effect of malicious attackers attempting to either prevent a robotic swarm from achieving its goal or influence the robotic swarm in such a way, so as to make it less efficient.

In order to assist in setting the context for this research, a brief overview of swarms within nature is provided, as are typical proposed uses of swarm technologies within robotics.

This work details the characteristics of robotic swarms and provides a taxonomy for robotic swarms, based on others previous work. This works demonstrates that, while the threat types for a robotic swarm are no different to the threats types for traditional information systems, there are differences with the attack types and attack methods. This is due to the unique characteristics of robotic swarms when compared to a traditional information system, or when compared to systems that are similar to robotic swarm systems, such as Wireless Sensor Networks and Ad-hoc networks.

In order to assist the reader, two use-case studies are presented through this thesis. These use-cases provide details of different swarm proposals, with different characteristics and end goals.

An abstract generic model is developed for robotic swarms, their entities and their operational environments. The generic model is used in conjunction with a taxonomy of threats, to enable this research to develop simulations for the robotic swarms, initially to investigate whether robotic swarms could be maliciously manipulated. The results of the simulations demonstrating that the robotic swarms could be successfully attacked and subsequently manipulated.

The work progressed to researching and implementing Intrusion Detection Systems (IDS), tailored for robotic swarms. This work was based on research around current IDS methodologies and it was determined that signature based IDS could be utilised but anomaly based IDS were not suitable for use in robotic swarms.

Simulations were undertaken that utilised signature based IDS functionality within scenarios that had previously been successfully attacked. The IDS functionality was shown to be successful in the majority of instances. That is, the robotic swarms were either more efficient at achieving their goal, when compared to being attacked without the IDS functionality, or the swarms were simply able to complete their tasks, when previously they had not.

This work concludes with a discussion of proposals for future research work within the field.

1.1 Motivation

Due to the increasing number of proposed uses for robotic swarms, it appeared that there had been very little thought or consideration given to the requirements for security within the proposed robotic swarm implementations. The majority of the literature concentrated on the potential uses and practical implementation issues of the robotic swarms. Therefore, the motivation behind this research was to gain an understanding of the current research within the area of robotic swarm research and focus on potential security issues within the robotic swarm implementations. It was realised that a significant number of security threats to a robotic swarm could be mitigated by existing techniques, such as cryptographic techniques that are proposed for wired, and wireless networks, fixed and mobile networks, computing and telephony, Wireless Sensor Networks and Ad-hoc Networks. However, it became apparent that robotic swarms had several unique features, when compared to the other systems, such as a lack of hierarchy and a decentralised control, these are detailed further in Section 2.4. Therefore, the focus of this work is to research the potential impact of security threats that are unique to the characteristics of a robotic swarm.

1.2 Overview of Robotic Swarms

Robotic swarm research is looking at multi-robot systems and is currently taking the majority of its inspiration from nature [188]. The philosophy behind a robotic swarm is that a large number of relatively basic robots will work in conjunction and cooperate with each other to complete a predetermined goal. It is generally agreed that a swarm has the following characteristics [49, 67, 75, 95, 142, 143, 160, 171, 183, 202, 203, 205]:

1. **Autonomy** Both the swarm and the individual entities within the swarm operate autonomously, without any external command and control.
2. **Decentralised control** The control of the robotic swarm is from the interactions of the robotic swarm entities with each other and the robotic swarm entities interactions with their operating environment. The robotic swarm is not controlled by an internal or external overarching control authority.
3. **Large number of members** To allow for redundancy and resilience within a swarm, large numbers of entities are used.
4. **Collective emergent behaviour** An individual entity would not be able to complete or undertake on its own. However, as a collective, the swarm entities interact with each other and the environment in order to reach the swarm's goal.

5. **Local sensing** The individual swarm entities have the ability to sense their local environment, in order to allow them to perform their required tasking in order for the swarm to realise its goal.
6. **Local communications** The swarm entities communicate within an area that is local to the entity. This allows for more efficient usage of the communications media, such as bandwidth constraints.
7. **Resilience to failures within the swarm** A swarm is designed such that it is resilient to failures, this is achieved by the large number of member entities within a swarm.
8. **Scalable in size** A swarm can alter its size in order to achieve its goal.

Indeed, Şahin [160] describes a swarm robotics as: "... the study of how large numbers of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interaction among agents and between agents and the environment." Where Şahin uses the term agent to refer to a robotic swarm entity.

1.2.1 Examples for the Proposed Uses of Swarms

Swarm robotics is generally within the research environment and actual applications for the use of swarm robotics are often theoretical. However, it is the proposed uses of swarms and the potential benefits that swarm robotics could deliver which is driving the majority of the research. There are proposed applications of swarm robotics in both civilian and military applications. Typical civilian applications include [45, 149, 180, 204] maintenance tasks, communications provision, use in space flights and space exploration, medical procedures and emergency response. The emergency response applications can include, but is not limited to, Ad-hoc communications, Search and Rescue tasking, contaminant tracking and provision of initial and continuous disaster scene information and situational awareness.

Typical military applications include [34, 112, 114, 184] communications provision, underwater mine clearance, landmine detection and assisting in the provision of situational awareness.

1.3 Statement of the Research Problem

Due to the many proposed uses for robotic swarm technologies and as a robotic swarm is an autonomous body operating, in many cases, beyond the control and visibility of the authority that released it, it is not inconceivable that swarms may operate in hostile environments where an attacker may detect the robotic swarm and attempt to manipulate its behaviour.

The research problem has been broken down to three separate questions that, based on the outcome of the previous question, follow on from each other.

Question 1: Can a malicious intruder affect the efficiency of a robotic swarm in achieving its goal by influencing the robotic swarm's emergent behaviour?

The initial research, based on robotic swarm taxonomies, was to gain an understanding of existing proposed implementations for robotic swarms. These implementations were then subjected to various malicious attacks, in order to ascertain if the malicious attacks would affect the efficiency of the robotic swarm.

The interest for this element of the research was whether a malicious intruder within the original robotic swarm could attack the original robotic swarm, so as to influence the original swarm's emergent behaviour and effect the overall efficiency of the original robotic swarm. The objective of the attacker could be to either prevent a robotic swarm from reaching its original goal, or reduce the efficiency of a robotic swarm in reaching its original goal.

For completeness, "conventional" attacks were also undertaken against the robotic swarm, such as jamming, in order to understand the effect of these attacks upon the robotic swarm but the focus of this part of the research was to understand if the unique characteristics of swarm could be utilised to an attacker's advantage.

Question 2: Can a robotic swarm detect that it is being manipulated?

Following on from the initial question, the research concentrated on the detection of a malicious attack upon a robotic swarm.

The research undertook a review of Intruder Detection Systems (IDS) technologies and techniques, considering their suitability within the context of robotic swarms. This included such principles as network IDS, with the research continuing to take into account the characteristics of a robotic swarm, as defined in Section 2.4, and an appreciation of design criteria that are unique to robotic swarm implementations.

Essentially the research question was to understand whether the principles of existing IDS could be applied to robotic swarms.

This question was then broken down into whether the IDS could be conducted by an independent IDS robotic swarm, which is separate to the original robotic swarm, or could the IDS function be undertaken by the original robotic swarm itself?

There is also the realisation that as the majority of robotic swarm technology is an emerging area of interest, the previous research is mainly undertaken within a benign research environment [24], such that robotic swarm IDS have not been considered. Therefore, this will be unique to both the concept of robotic swarms and the robotic swarm implementations.

Regardless of the IDS implementation, whether internal to, or external from, the original robotic swarm, the IDS must minimise any interference with, or influence on, the emergent behaviour of the original robotic swarm during its normal operation.

It is worth noting that it is possible for the characteristics of a robotic swarms implementation to actually hinder the detection of malicious activity, or lead to false-positives. For example, the emergent behaviour of a robotic swarm will be based on many factors, such as the environment in which a robotic swarm is operating within. A robotic swarm might react to a change in the operating environment, such that the robotic swarm's IDS believes that it is subject to an attack. To place this into context, imagine a robotic swarm entity is navigating by following a pheromone trail, in a similar manner to an ant foraging, and then the pheromone trail abruptly, and uncharacteristically, stops. The robotic swarm entity has reached the end of the pheromone trail but has not reached its goal. Is this change to the environment an attack on the robotic swarm that is attempting to disrupt or deny communications and preventing the swarm from reaching its goal, or has it simply rained and washed the pheromone trail away?

Similarly, as a robotic swarm generally has a large number of member robotic entities but the robotic entities only have a limited sensing range would, or could, a swarm be able to detect "extra" entities within the robotic swarm. This might be especially difficult, as certain robotic swarm implementations are designed so as to be scalable in size. These extra robotic swarm entities could be an attacker, attempting to infiltrate a malicious robotic swarm into the original robotic swarm, or it might just be new entities supporting the original robotic swarm deployment.

Therefore, an IDS that is proposed for a robotic swarm must take into account the characteristics of both the robotic swarm and its implementation.

Question 3: Can a robotic swarm mitigate the security risk if it detects that it is being interfered with?

The final question is essentially, if a malicious intruder can effect the emergent behaviour of a robotic swarm and the robotic swarm perceives that it is being subject to an attack, how should, or can, the robotic swarm react?

Initial research was undertaken as to the possible reactions to a perceived attack, with recommendations as to how this could be further explored within future work are detailed.

1.4 Contributions

The research presented in this thesis has made several contributions to knowledge, regarding potential attacks and subsequent detection of malicious activities within a robotic swarm.

In order to carry out this research, it was required to identify a taxonomy for robotic swarms. A taxonomy was developed which was based on previous swarm taxonomies and was tailored to this research, ensuring that different types of robotic swarm were considered.

The work I undertook in this research has led to the following contributions:

- Identified threat types for robotic swarms.
  This research undertook a detailed review of the types of threat against a swarm. The research proposes that there are no new threat types that are specific to a robotic swarm and details how the threats could be realised within a swarm.
- Identified attack types and defined attack methods that could be used against a robotic swarm.
  This research provides details of new and unique attacks that could be undertaken against a swarm.
- Produced a generic model for robotic swarms.
  A generic model was developed to allow the research to consider and simulate the various elements of a swarm, such as the swarm entities, failed entities, malicious entities and the swarm's operating environment.
- Demonstrated that the emergent behaviour of a robotic swarm could be altered.
  Simulations were undertaken which demonstrated that swarms could be maliciously manipulated.
- Concluded that Anomaly based Intrusion Detection Systems are not appropriate for robotic swarms.
  The research provides an argument as to why Signature based Intrusion Detection Systems are suitable for use within a swarm and Anomaly based Intrusion Detection Systems are not appropriate.
- Produced and successfully demonstrated robotic swarm based Intrusion Detection Systems.
  Based on earlier work within this research, which demonstrated that swarms could be maliciously manipulated, the same swarm implementations had Intrusion Detection System techniques applied to them. This research demonstrated that the Intrusion Detection Systems approaches improved the performance of the swarm's in achieving their goals.

1.5 Structure of the Thesis

This thesis is presented in three parts: Part I is the Background, Part II details the Contributions and Part III is the Conclusion of the research.

Part I: Background The Background part consists of this Introduction, Chapter 1, and the Background to the research Chapter 2.

Chapter 1 provides a brief overview of swarm robotics, the motivation behind the research, the research problems and the contributions made to knowledge.

Chapter 2 provides an overview of how robotic swarms evolved, describes the characteristics of robotic swarms, how they operate and provides a taxonomy for robotic swarms. It details proposals for robotic swarm implementations and provides examples of proposed applications for robotic swarms.

Part II: Contributions The contributions part details how the research was undertaken, the simulations that were performed, the results obtained and the analysis of the simulation results.

Chapter 3 provides a generic model for robotic swarms and their operating environments. Use cases are introduced for typical swarm uses, which are then followed through the remainder of the thesis.

Chapter 4 provides a taxonomy of the threats to which a robotic swarm could be subject. This includes details of the different types of threat and provides details of the adversary, including the manner and methods an adversary could use to attack the robotic swarm. Based on the robotic swarm taxonomy, the generic model that had been developed and the research relating to threats and attacks against robotic swarms, different robotic swarm types were selected to be subjected to attacks.

Chapter 5 provides details of how implementations for robotic swarms, that had been proposed by previous researchers, were reproduced. It then details how this research successfully attacked robotic swarms and how malicious activity altered the emergent behaviours of the robotic swarms.

Chapter 6 follows on from the successful attacks against the robotic swarms. It provides an overview of Intrusion Detection Systems and considers the implementations of Intrusion Detection Systems within robotic swarms and how this would be considered within the generic model for robotic swarms.

Chapter 7 concludes the contributions part of the thesis with simulations of Intrusion Detection Systems within robotic swarms, considering techniques that are both internal to the robotic swarm and by utilising a dedicated external robotic swarm.

Part III: Conclusions The thesis concludes with a Discussion in Chapter 8. The discussion provides a review of the research undertaken and further analyses the research results. There are also proposals provided for future work within the research area.

Annexes

Annex A provides additional supporting information regarding Intrusion Detection Systems. The annex provides an overview, background and history of Intrusion Detection Systems and provides details as to the various categories, technologies, implementations and components.

Annex B provides details of the characteristics of typically available robotic swarm robots.

2 Background

2.1 Introduction

This chapter provides background details on swarms, including swarms within nature, robotic swarms and the emergence of swarm robotics. Taxonomies of robotic swarms, the fundamental characteristics and attributes of robotic swarms and a comparison between robotic swarms and other similar technologies and systems are presented, as is a definition of swarm robotics.

2.2 Emergence of Swarm Robotics

Various people have contributed to the emergence and the understanding of swarms robotics, as they are generally recognised today.

2.2.1 Pre-Swarm Robotics

Prior to the term swarm robotics, and associated terminology and principles, there had been a significant amount of work undertaken in the field of robotics, biology, mathematics and physics. All of which have combined to provide various approaches, understandings and influences to the field of swarm robotics.

Robotics is an area of research and development which has undertaken significant change. Robots can be implemented either fully autonomously or semi-autonomously, and are generally utilised in either repetitive or dangerous situations. Also, they are sometimes used for learning and entertainment purposes.

The word "robot" was first introduced by the Czechoslovakian playwright, Karel Čapek, in a play entitled R.U.R. (Rossum's Universal Robots), in 1920. The word roboti was conceived by his brother, Josef Čapek, and is derived from the Czechoslovakian word "robota", meaning "serf labour". Robot is the English form of the word roboti.

Industrial robotics were initially conceived by George Devol and Joseph Engelberger in 1959 and the first programmable industrial robot was installed by General Motors at their Ternstedt plant, in 1961 [88]. The tasks of these industrial robots were to undertake repetitive tasks involved with the production of car parts for the automotive industry and working in die-cast production processes, with the associated high temperatures.

In 1969, General Motors installed the first production line of robots, in their Lordstown assembly plant, to undertake spot welding tasks. These spot welding robots allowed for more than 90% of the welding tasking to be undertaken by robots. Vision sensors were first incorporated within industrial robots in 1973, by Hitachi. The vision sensors allowed a robot to recognise bolts on a moving mould, which allowed the robot to interact with the moving bolts in a synchronised way. The first microcomputer controlled robot and the first microprocessor controlled robots arrived in 1974. In 2010, Fanuc developed a robot that was capable of sensing and then learning its own vibration characteristics. This allowed the robot to suppress vibrations in the robot's arm and therefore allowed for increased accelerations and speeds of the robot's arm.

Some of the first electronic autonomous robots were designed by William Grey Walter in 1948 and, due to their slow speed and shape, were often known as "tortoises".

2.2.2 The Term Swarm

The original term for a swarm, within the context of robotics, was cellular robots, as described by Beni [19]. However, around the same time, the term cellular robots was also being used by Fukuda in Japan, to indicate groups of robots that could work like the cells of an organism, in order to assemble more complex parts [69].

Therefore, Beni decided that new terminology was required, in order to describe this field of research, and the term he decided upon was the word *swarm*. The term swarm was chosen due to a casual conversation with a person called Alex Meystel, where Meystel suggested to Beni that he needed a "buzz word" to describe "that sort of 'swarm"'. Beni considered the use of the word swarm and how the groups of robots that was being researched were more than just a group of robots, due to the robots' various characteristics. Beni discussed his work with Guy Theraulaz, who had undertaken his own work on insects, and it became apparent to Beni that the term swarm was appropriate [189]. Beni suggested that it was clear that the concept of a swarm was quite appropriate for biologists, in trying to explain the behaviour of insect societies. It was therefore also appropriate for roboticists, researching collective task undertaking of robots, to also use the term swarm.

2.2.3 Swarm Intelligence

Beni also notes that the term "swarm intelligence" was also being used. Beni observed that the term swarm intelligence was a complicated term to justify, as the term "intelligence" is difficult to define [19].

Within nature, a typical example of swarm intelligence is the building of the complex structure of a termite mound [70]. Within swarm robotics, the aim is for systems that are capable of carrying out not just useful tasks, but also "intelligent" tasks, such as search and rescue operations.

Beni starts by proposing preliminary definitions for a swarm robot. Initially this was "A robot whose behaviour is neither random nor predictable" and this was enhanced to "A robot capable of forming material patterns unpredictably".

These then led to preliminary definitions for swarm intelligence. Initially, Beni defined an *Intelligent Swarm* as [19]: "a group of non-intelligent robots forming, as a group, an intelligent robot. In other words, a group of "machines" capable of forming "ordered" material patterns "unpredictably"." This was then modified to "a group of "machines" capable of "un-predictable" material computation". This was then further refined to "A group of non-intelligent robots ("machines") capable of universal material computation".

Corne, Reynolds and Bonabeau also note how natural organisms behave when they are in groups, such as swarms of ants, swarms of bees, colonies of bacteria, flocks of bees, or school of fish [42]. Essentially, the various groups exhibit behaviours that individuals do not, or cannot, undertake. They suggest that if the group is considered collectively, as a swarm, the swarm exhibits a behaviour that is more intelligent than the individuals within it. It is this observation that has led to the concepts and the algorithms which have become associated with swarm intelligence.

From this, they attempt to define swarm intelligence. They propose that this might be [42]: that useful behaviour emerges from the cooperative efforts of a group of individual agents, in which the individual agents are largely homogeneous, act asynchronously and in parallel. They also suggest that there is little, or no, centralised control and communications between agents is largely effected by some form of stigmergy. They do suggest that "useful behaviour" is considered to be relatively simple, their suggestion of typical examples being: finding food locations or nest building, and they do say that it is not writing a symphony, or surviving for many years in a dynamically changing environment.

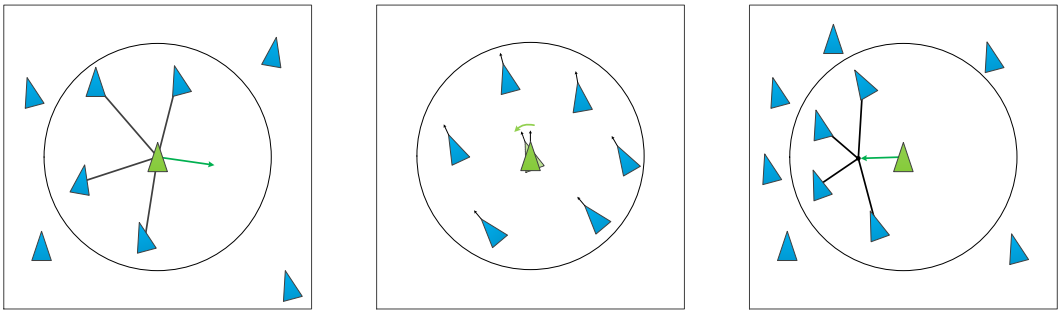
Synchronised movements of individuals within a swarm is also considered to be a field within swarm intelligence. Of particular note is the work undertaken by Craig Reynolds [152], on the problem of defining simple rules for individuals within a swarm to follow, that would lead to the natural and realistic behaviour of a simulated swarm.

Corne, Reynolds and Bonabeau [42] note that swarms in nature all tend to share common emergent behaviours, primarily:

1. The individuals members within the swarm stay close to each other, but will ensure that they are at a distance so as not to be too close, and there appear to be no collisions.
2. Swarms appear to change direction smoothly and collectively, as if the swarm was a single organism.
3. Unlike a single organism, and still smoothly and cleanly, swarms can sometimes pass directly through narrow obstacles, in the way that is similar to a stream of water passing around a vertical stick placed centrally in the stream's path.

The seminal work by Reynolds [153] provided the details and the demonstration of simple mechanisms that can explain the behaviours. The work is so widely recognised within the field of swarm intelligence, as well as the computer simulation community, that the mechanisms and rules first proposed by Reynolds are known as "Reynold's rules". The rules state that each individual in the swarm, often referred to as a "boid", adheres to the certain steering behaviours. A boid is required to maintain a minimum *separation* from other boids, a boid will steer itself so as to avoid coming too close to other nearby boids. Boids will attempt to steer towards the mean heading of the surrounding boids that are nearby, in order to achieve *alignment*. A boid will attempt to steer itself towards the mean location of other nearby boids, in order to achieve *cohesion*.

The three Reynolds rules of separation, alignment and cohesion can be seen in the Figure 2.1.



(a) Separation **(b)** Alignment **(c)** Cohesion

Figure 2.1: Reynold's Rules for Synchronised Movements

These rules can be refined to enhance simulations, such as by constraining the variables of an individual, as seen in Figure 2.2.

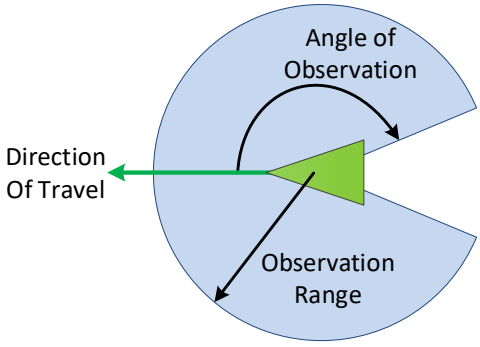


Figure 2.2: Constraining a Boid's Variables

2.2.4 Swarm Optimisation

Beni [19] and Corne, Reynolds and Bonabeau [42] all note that the term "swarm optimisation" was also being used in relation to swarm intelligence. The research around swarm optimisation is inspired by the natural behaviours of swarms in nature. By observing these swarms in nature, researchers are attempting to implement algorithms that either mimic, or are based on the overarching natural behaviour of a swarm, such as in the undertaking foraging and optimisation tasking and problems. The recognised seminal work on particle swarm algorithms being Kennedy and Eberhart [105] "Particle Swarm Optimization" (PSO) algorithm and the seminal works on ant colony optimisation being Dorigo's "Ant Colony Optimization" (ACO) algorithm [55] and Colorni et al "Distributed Optimization by Ant Colonies" [40]. Beni [19] suggests these algorithms are undertaken asynchronously and in a decentralised fashion, in a behaviour that is similar to swarms of insects.

The PSO research undertaken by Kennedy and Eberhart [105] cited Reynold's work [153] as contributing to its inspiration. In PSO, simple software agents, called particles, move in a search space of an optimisation problem. The position of a particle represents a candidate solution to the optimisation problem. Each particle searches for better positions in the search space by changing its velocity according to rules, as originally inspired by Reynold's behavioural models of bird flocking. The components that influence an updated position of a particle, within the search space, are the particle's *current velocity*, the particles *personal best* and the *global best*. A particles personal best is the "fittest" position the particle has yet encountered and a component of the particle's updated velocity is the direction from its current position, to its personal best. The global best is the best position that any particle within the swarm has yet discovered, essentially this is the best of the personal bests. The final component of the velocity update, which is shared by all particles, is a vector in the direction from the particle's current position to the globally best known position.

The positions and velocities of the particles are randomised and then move through the search space, updating their positions by adding vectors, based on a particle's current velocity and scaled vectors from the personal and global best components. As they move through the search space, the particles measures the "fitness" of their current position, updating personal and global bests accordingly.

The ACO algorithms are similar to how ants optimise problems in order to conserve energy. For example, in nature an ant will locate a food source by following pheromone trails, as laid down by other ants from within its colony. The colony finds and collects the food in what the colony perceives to be the most efficient manner, in order to conserve energy. Essentially, in an ACO algorithm, an artificial ant builds a solution to a problem, leaving a trail of "artificial pheromone" as it goes. Subsequent artificial ants build new solutions to the problem but are influenced by the pheromone trails that have been left by previous artificial ants, thus optimising the solution.

2.2.5 Swarm Engineering

The term "swarm engineering" was first introduced by Kazadi and suggests that swarm engineering is "the natural evolution of the use of swarm-based techniques in the accomplishment of high level tasks using a number of simple robots" [102]. Kazadi also suggests that swarms are difficult to engineer, due to the fact that groups of independently operating, but interacting, agents can reveal complex and unexpected behaviours, due to many reasons [103]. An example provided by Kazadi being that "small perturbations to the system, which cause rather small changes in the behaviours of individual agents, can cause very large changes in the overall behaviour of the system". Brambilla et al. suggest that swarm engineering is still "an emerging discipline that aims at defining systematic and well founded procedures for modelling, designing, realising, verifying, validating, operating and maintaining a swarm robotics system" [24].

Similarly, Winfield et al. suggest that swarm engineering is the bringing together of dependable systems engineering, such as such as safety critical systems engineering, and swarm intelligence [202].

2.2.6 Swarm robotics

It is noted that the field of "swarm robotics" concentrates on "what may be achievable by collections of small, simple robots furnished with relatively non-sophisticated ways to communicate" [42, 130, 159]. They highlight the swarm characteristics of robustness, flexibility and scalability, along with potential areas for the use of swarm robotics, suggesting uses such as agricultural, construction and exploration.

There have been many studies of and proposals for the use of swarm robotics and various research activities undertaken. Examples of these are detailed within Section 2.7, where various applications for swarm robotics have been proposed. There have been various research undertakings, which have concentrated on the various elements of swarm robotics. This has included: the physical design and implementation [14, 39, 66, 74, 124, 130, 131, 204], swarm behaviours [171, 175, 176, 177] and the objective of undertaking tasks with minimal energy use [154].

2.3 Capabilities and Limitations of Swarm Entities

The capabilities, such as the size, power and communications capabilities of a swarm entity will depend upon the implementation of the swarm. The literature discusses physical implementations in the ranges of 2mm x 2mm x 1mm for the I-SWARM implementation [204], 33mm diameter for the Kilobot [99] and the S-Bot, that was utilised in the Swarm-Bots research into self-assembly artefacts, measured 116mm in diameter and 100 high [181].

There is discussion within research to use nano-bot technologies in order to inject humans with the nano-bots to enable precise delivery of drugs, such as in cancer care. The US National Cancer Institute founded the Nanotechnology Characterization Laboratory (NCL) in collaboration with the US Food and Drug Administration (FDA) and National Institute for Science and Technology (NIST) with the objective of develop nanotech therapies and diagnosis [137].

Details and characteristics of typically available swarm robots are provided in Annex B .

As can be seen from Annex B, the capabilities of typically available swarm entity robots currently range from: the Kilobot [99, 157, 158], with a processing power of 8bits at 8MHz, with 32K of Flash, 1KB of EEPROM and 2KB of SRAM; through to the e-puck2 [71], with a processing power of 32-bit at 168 MHz, with 1024K of Flash and 192KB of RAM. Both of these devices were designed for a benign research and development environment.

With the potential processing power that is available, such as with the e-puck2, it could be possible to undertake security functions, such as utilising encryption techniques, on some of the devices. The e-puck2 has also been designed to allow additional "special to type" modules to be developed and connected to the device in order to undertake specific functions, a module could therefore be developed that is designed to undertake cryptography functions and remove this burden from the e-puck2's processor.

However, this research is interested in the cases where the robot entities have insufficient processing power to undertake security functions that require higher power and potentially resource intensive processes or operations, such as authentication mechanisms or encryption of the traffic capabilities.

There could also be further complications, such as key management across swarm entities that have been deployed at different times and locations. There is also the consideration that swarm entities will be lost and prone to capture, once released, which could lead to a key variable compromise and, if a compromise is actually discovered, the potential management overhead of then managing the key variables on the remaining swarm entities.

This research therefore assumes that the characteristics of the swarm entities capabilities are constrained to an extent that they are not capable of undertaking common security functions, such as authentication mechanisms or encryption techniques, and potential attacks against the limitations of the swarms and swarm entities need to be considered by other means.

2.4 Swarm Characteristics

As introduced in Section 1.2, it is generally agreed that that robotic swarms have various characteristics.

It is worth describing the robotic swarm characteristics in more detail, in order to allow the reader to gain a better understanding of the characteristics of a robotic swarm and how this can both benefit and constrain an individual robotic swarm entity, or indeed the robotic swarm itself. The explanations also provide an example of how the characteristics relate to nature. This is to assist in the reader's understanding of the characteristic and understand how, and why, other researchers have implemented the various characteristics.

2.4.1 Autonomy

When a robotic swarm is released, the robotic swarm operates autonomously on two levels, both from an individual robotic swarm entity perspective and from the perspective of the robotic swarm as a whole. For example, an individual robotic swarm entity will interact with other robotic swarm entities and its environment, the resulting actions of the individual robotic swarm entity will be determined by the individual robotic swarm entity, based on its own interactions. Secondly, due to a lack of centralised command and control, the robotic swarm, as a whole, will react in a way that is determined by the collective behaviour of individual robotic swarm entities or due to environmental events, that affect the majority of the robotic swarm entities, leading to a collective emergent behaviour. Examples in nature of the two levels of autonomy can be observed in some bees [22]. An individual bee will decide whether or not to follow another bee to a feeding place, based on its own knowledge of other feeding areas and the requirements of the hive. The bees, when thought of as a swarm, will also behave in ways such as acting together to maintain a constant temperature of the hive, by either adding or removing individual members, and the swarm moving location, due to either the size of the hive or a disturbance to the hive [22].

2.4.2 Decentralised Control

A robotic swarm differs from the usual implementation of multi-robot systems, in that in the case of multiple robots, the individual robots are usually controlled from a central location. In order to allow the reader to gain a better understanding of this decentralised control, examples of centralised control schemes are first provided, followed by the decentralised control scheme. An example of multiple robots being controlled from a central location, in a non-mobile scenario, would be a car production line. An example of a multiple centrally controlled mobile robots would be an automated warehouse, where robots interact with each other to store and collect items and then move the items over distances to pre-determined points.

In swarm robotics, the resultant behaviour of the robotic swarm is as a result of the interactions of the individual robotic swarm entities, interactions with the environment and the decision making capabilities of the individual robotic swarm entities [12, 22, 142, 204]. The robotic swarm is given an initial objective, but how the robotic swarm goes about completing the objective is not predetermined. For example, a robotic swarm may be required to travel a certain point. How the robotic swarm gets there is not controlled by the original requesting authority, but by the individual robotic swarm entities acting, and reacting, to various stimuli that guide the direction of travel of the overall robotic swarm. If the robotic swarm comes across an obstacle and there are two ways around it, a central controlling authority is not available to inform the robotic swarm entities which way to go, and the robotic swarm will essentially work it out. In nature, this can be seen when ants forage for food. Without any previous knowledge of where to forage, the ants will search for a food location and then produce pheromone trails for the other ants to follow. There is no central controller informing which ants where to go in the initial search for a food source, or, once multiple food sources have been found, which ants to go to which food source [22].

2.4.3 Large Number of Members

The majority of proposed robotic swarms suggest utilising large numbers of individual robotic swarm entities. This is due to several reasons and can be related to the required tasking of the robotic swarm. The large number of members allows for redundancy and resilience within the robotic swarm [142]. That is, if one robotic entity fails, then this is a small overall percentage of the robotic swarm and the robotic swarm should be able to complete its task. There are also the physical constraints of an individual robotic swarm entity, such as how far it can communicate or how much of the environment it can interact with and therefore be influenced by. There might therefore be physical constraints upon the robotic swarm that require large numbers of robotic swarm entities, in order to enable the robotic swarm to operate effectively and efficiently, in order to achieve the robotic swarm's goal. If a robotic swarm were required to move an object which was too big for an individual robotic swarm entity to move, multiple robotic swarm entities could push, and possibly pull, the object at the same time. It should be noted that in experiments to demonstrate co-operative working of multiple robotic swarm entities, there does become a point when there can be too many robotic swarm entities attempting to complete a task and the process becomes inefficient [142]. An example of large numbers of members in nature that perform a task as a swarm are termites building a termite mound. One termite initially decides where to start building a pillar and then other termites contribute to the build [22]. Also, if small numbers of termites are killed, the remaining quantity of termites available allows the construction to continue.

2.4.4 Collective Emergent Behaviour and Swarm Intelligence

Individually, a single robotic swarm entity would not be able to complete a task but working as a group, the robotic swarm has the ability to do so. The robotic swarm exhibits a collective emergent behaviour, in order to achieve the required goal. An example of this could be that a robotic swarm is required to navigate from its current location to another point. However, to get to the final point, the robotic swarm is required to negotiate adverse terrain, such as a debris field in a search and rescue tasking. The issue could be as simple as crossing a gap that is larger than an individual robotic swarm entity. To complete the task, the robotic swarm could form bridges out of the robotic swarm entities, in order to span the gap and allow the robotic swarm to proceed. The complex behaviour that can be observed from the swarm is often referred to as swarm intelligence [12, 22, 112, 114, 184, 204]. It is worth noting that "complex behaviour" within robotic swarms is still extremely basic when compared to the biological systems, which swarm robotics is often compared to. The I-SWARM project suggested that an important requirement is that the robotic swarm entities need a means to distinguish between their "kin", to be able to create effects [204]. They suggest that a possible solution could utilising communications between entities, by the use of either near-field communication methods or infra-red LEDs.

It is suggested that many of the collective activities of social insects are self-organised and that by using self-organising systems to solve problems leads to swarm intelligent systems [22]. The expression Swarm Intelligence was first used in the context of one or two dimensional cellular robotic systems used to generate patterns through nearest neighbour patterns [22]. The meaning has since been extended to included attempts to design algorithms, or problem systems and implementation, that take inspiration from the collective behaviour of social insects or animal colonies and societies.

In nature, weaver ant (*Oecophylla*) workers form chains of their own bodies, this allows them to cross wide gaps and pull stiff leaf edges together to form a nest [22]. Several chains can join together to form bigger ones, which workers can then travel over. Such chains can then create enough force to pull the leaf edges together.

It has been suggested that a variety of collective behaviours might be required, in order to solve a given problem, with the suggestion that insects have four not mutually exclusive functions in enable collective behaviours [70]. The four functions being: coordination, cooperation, deliberation and collaboration.

2.4.5 Local Sensing

The individual robotic swarm entities have local sensing abilities, in order to allow them to perform the required tasking of the robotic swarm [204]. Local sensing provides the robotic swarm entities with the ability to gain an awareness, and knowledge, of the existence of other individuals and the operating environment.

For example, a robotic swarm may be required to find the heat source of a human body, in a search and rescue scenario. An assumption could be that the human body is warmer than the surroundings and cooler than other heat sources, such as localised fires. The individual robotic swarm entities are capable of searching for the characteristics of the human body's heat profile and ignore any other heat sources that do not fit the profile. The size and interaction of the robotic swarm could make this an efficient method of locating the body; whilst the individual robot's local sensing actually performs the task. An example in nature of local sensing is where ants and bees go off in random directions to search for food [22]. When an individual locates a food source, the individual informs the rest of the swarm of the location of the food source, in order to allow the food to be collected.

2.4.6 Local Communications

The communications between the robotic swarm entities is generally accepted to be at a local level only. That is, the robotic swarm entities have a limited communications range between individual robotic swarm entity and this is mainly due to the communications techniques used. These are often near field Radio Frequency (RF) or via optical methods, such as visual or infra-red LEDs, for remote communications between the individual robotic swarm entities [204]. There are also communication techniques where the robotic swarm entities communicate between each other when they are physical connected to each other.

An advantage of local communications is that essentially the available bandwidth is increased [45]. That is, multiple robotic swarm entities can transmit on the same frequency without interfering with each other, as long as they are far enough apart. This frequency reuse also means that, although fewer frequencies are utilised, a frequency management scheme might be the trade off required. That is, a robotic swarm entity could monitor a particular frequency, to ensure it is not in use by another robotic swarm entity, prior to sending a transmission. An example of local communications in nature is the "waggle dance", that is performed by bees to inform other bees of where to go to find nectar [22]. The only bees to receive the message are the bees local to the bee performing the waggle dance; other bees will not receive the message.

Also, communications distances can vary, depending upon the circumstances. The bees that communicate using the waggle dance are the bees that are local to the bee undertaking the dance. However, if a bee hive is under threat, such as from an attack by a predator, then the bees can release a pheromone, in order to raise the alarm and recruit bees to defend the bee hive. The pheromone attracts bees from further away, in order to attempt to defend the hive. Similarly, distress or warning calls from animals need to be distinct from, and possibly travel further than, normal social communications between animals. It is worth noting that an alarm call might actually give the location away of the calling animal. This could result in the animal sacrificing itself, in order to save the others within the community.

A form of local communications that is utilised within nature and is proposed for use within swarm robotics is stigmergy [92, 179].

Stigmergy is basically a description that describes an indirect communication taking place between individuals. The term “stigmergy” was first proposed and introduced by French biologist Pierre-Paul Grassé, in 1959 [79]. This was in reference to termite behaviour when nest building and was defined as "Stimulation of workers by the performance they have achieved".

The principle of stigmergy is that traces are left and modifications are made, by individuals, to their physical environment, that results in a record being made in the physical environment [188]. This record can then be used as a feedback mechanism, in order to direct either the individual themselves, or other members of a group, in order to produce a collective behaviour and can be used to constrain individual behaviours. Various forms of record are used within stigmergy, such as pheromone scent gradients, where an individual might follow an increasing pheromone scent density, and material structures, where physical markers are made within the operating environment [145].

What is important to note with utilising stigmergy for communications, is that the individual entities do interact with each other, in order to achieve coordination amongst themselves, but they interact *indirectly*.

2.4.7 Scalability

Scalability is the ability to have extra members added to the robotic swarm, in order to assist with a tasking, or removing members from a robotic swarm, when a tasking has been completed or no longer needs a large amount of robotic swarm entities in order to achieve its goal. Scalability also allows for a robotic swarm to be supplemented by extra robotic swarm entities, in order to compensate for failed robotic swarm entities. It also allows a small number of robotic swarm entities to undertake an initial part of a task, such as searching for an object, that then needs to be increased in order to fulfil the entire tasking, such as recovering an object.

It is suggested that a robotic swarm should be able to operate under a wide range of group sizes and that its operation should be relatively undisturbed by changes in the group sizes [159].

2.5 Taxonomy of Swarms

There are several proposals for defining a robotic swarm taxonomy in the literature. These taxonomies are based on several factors, such as physical design considerations, collective behaviours and design methodologies [24]. Typical examples being the swarms size, communications mediums, coordination, control, design approach, processing abilities and collective behaviours, such as foraging and construction [60, 61, 62].

Other proposals suggest that the attributes for consideration should be the highly dependent features of group architecture, how members deal with resource conflicts, how a collective learns and adapts to a task, how cooperation is motivated and how planning is addressed [8, 31]. Indeed, certain taxonomies also consider the environment in which the entities are operating [8].

The following discussion provides details of various swarm taxonomies, as detailed within research literature.

2.5.1 Method-Based Taxonomy

It has been proposed by Brambilla et al. [24] that swarms robotics methods can be grouped according to design methods and analysis methods, as shown in Figure 2.3.

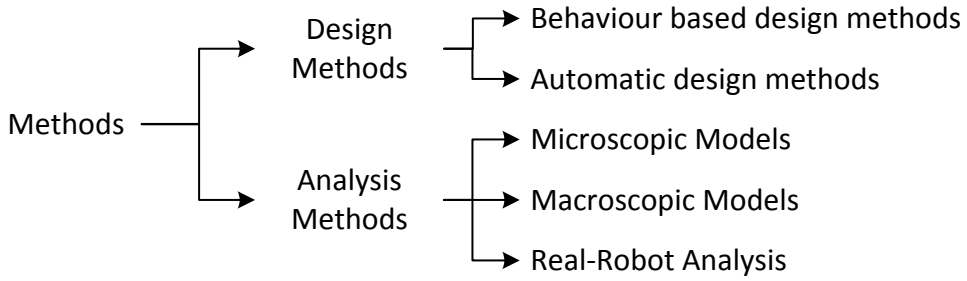


Figure 2.3: Method Based Taxonomy [24]

They suggest that the most common design method is behaviour based, generally a bottom-up process, but can be a top-down process, and is often based on the observations of animals in nature. They categorise behaviour based designs into the three main categories: probabilistic finite state machine designs; virtual physics designs; and other design methods. They also suggest that automatic design methods are classified as either evolutionary robotics or multi-robot reinforcement. These design methods are employed with the aim of reducing the effort of developers, as they do not require the intervention of the robotic swarm developer.

They suggest that the analysis methods for swarm robotics can be modelled at two different levels: the individual level, or *microscopic* level, modelling the characteristics of of single individual entities and interactions between them; the collective level, or *macroscopic* level, which models the characteristics of the entire swarm. They also suggest the use of real robots, as opposed to simulations, to confirm robotic swarm behaviours.

2.5.2 Collective Behaviour Based Taxonomy

Brambilla et al. [24] also suggest that swarm robotics can be categorised by their collective behaviours, as shown in Figure 2.4.

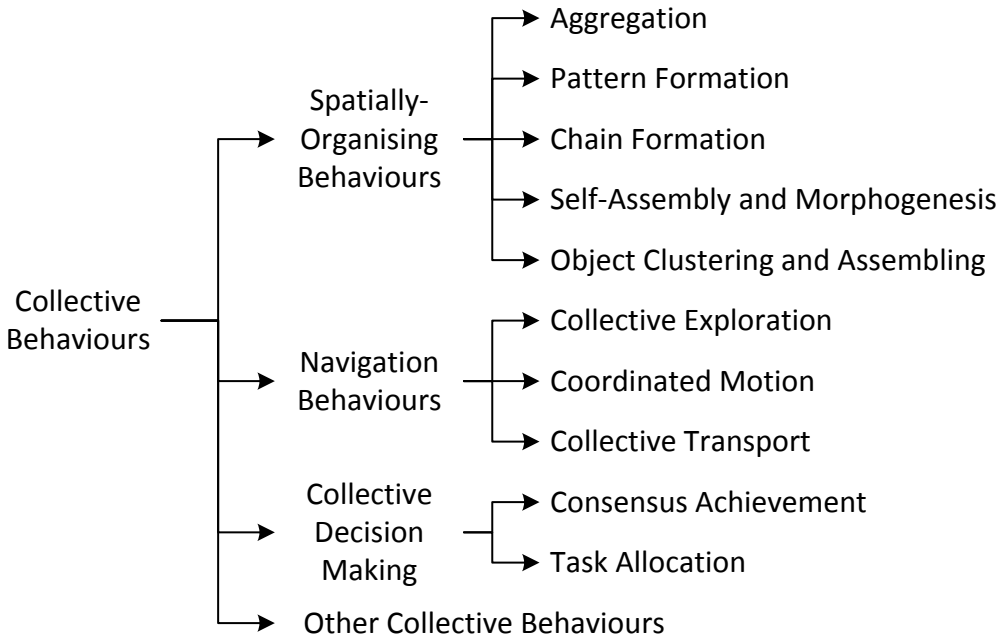


Figure 2.4: Collective Behaviour Based Taxonomy [24]

They propose four categories for the collective behaviours. These categories detail how robotic swarms: organise and distribute themselves; navigate and coordinate their movements; undertake decision making; and any other collective behaviours that do not correspond to the previous other categories.

2.5.3 Swarm Behaviour Taxonomy

Cao et al. [31] defined the taxonomy from a collective behaviour perspective, as shown in Figure 2.5. They propose separating the behaviours in the five main categories of: group architectures; resource conflicts and how conflicts are resolved; the origins of cooperation and how cooperation between entities is achieved; how the collective entities learn how to solve problems; and how the collective entities conduct path planning from a geometric problems perspective.

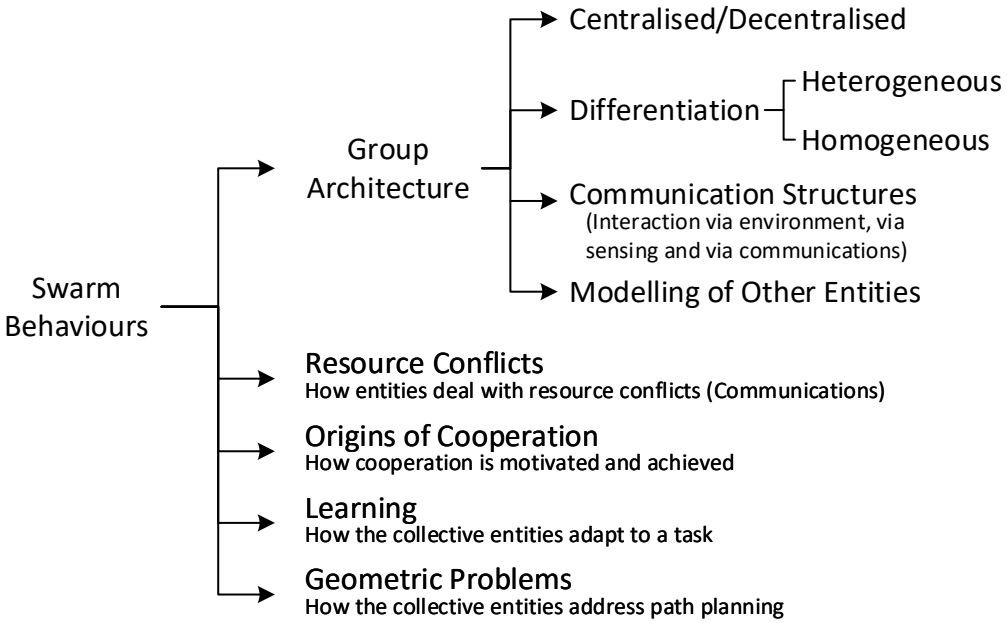


Figure 2.5: Swarm Behaviour Taxonomy [31]

2.5.4 Swarm Specifications and Attributes Taxonomy

Dudek et al. [60] suggests a taxonomy that is defined on a swarm's characteristics and its attributes, as shown in Figure 2.6. They suggest that a robotic swarm taxonomy could consider the collective swarm as a whole, as opposed to individual entities. They considered a robotic swarm's specifications and its attributes, suggesting that this could be separated into seven categories. These are: the collective size of the robotic swarm; the communications range between entities within a robotic swarm; the communication topology used; the quantity of information that can be transferred; how the robotic swarm entities can re-configure their organisation; how much processing capability a robotic swarm entity possesses; and whether the robotic swarm's composition is either homogeneous or heterogeneous. They provide an example of of task that is to search for a lost child or robot (known as the "find Robbie" task). They suggest that if the collective size of the robotic swarm approached infinity, then the flood filling of the operating environment would eventually lead to the task being completed, as all the space is filled by searching robots. In this instance, they also comment that that the robotic swarm entities would not need to communicate with each other. They also suggest that although there might be instances where robotic swarm entities do not communicate with each other, that they would still need to observe each others actions, in order to understand behaviours. They also discuss communication to either nearby robotic swarm entities or communications to the entirety of the swarm, based on the communications range of the robotic swarm entities.

They suggest that the communications topology could vary between a broadcast methodology through to individual addresses, where communications could take place through either a tree structure or via a "graph" structure, to improve redundancy within communications links. They do comment that individual addressing can add complications, such as a reduced interchangeability or distinctive roles for individuals, unless a dynamically based actions and roles are considered. When considering bandwidth, they considered this from a cost perspective. That is, the communications bandwidth is either free, the communications overhead can be ignored, through to very high cost, where the cost of communications cost more than a robotic swarm entity moving from one location to another. They suggest that this implies very independent robotic swarm entities.

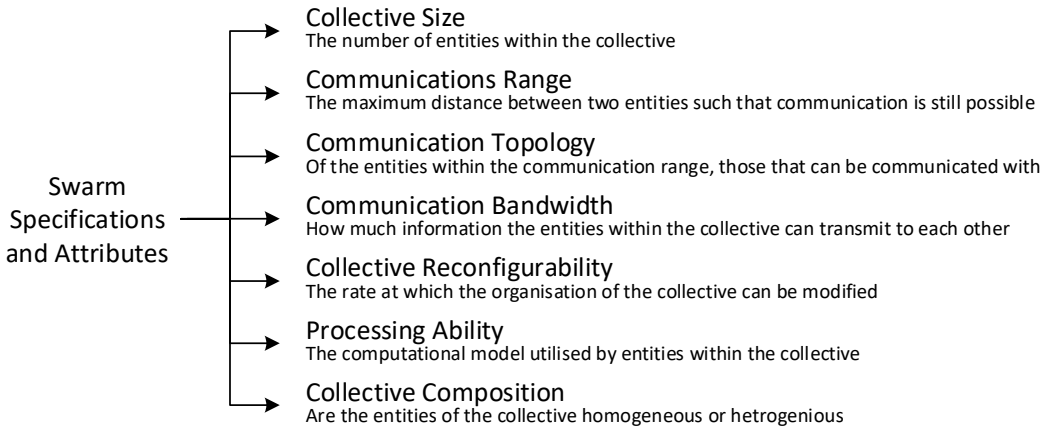


Figure 2.6: Swarm Specifications and Attributes Taxonomy [60]

It could be argued that a robotic swarm is homogeneous in nature, as, in its purest sense, all the robotic swarm entities ought to be the same. However, when considering practical implementations of robotic swarms, they might be considered heterogeneous, as robotic swarm entities might perform different functions, such as marking locations as opposed to locating objects. Conversely, this could be considered as multiple homogeneous robotic swarms working together, in order to achieve a single goal.

2.5.5 Proposed Taxonomy

Based on the taxonomies detailed from the literature review, the proposed applications for swarms, see Section 2.7, and the objectives of this research undertaking, see Section 1.3, then a simple taxonomy is proposed. The proposed taxonomy is based on how robotic swarm entities interact between themselves and proposes the interactions to be either Finite State Machines Implementations [1, 143], Physics Based Implementations [178] and Particle Based Implementations [5].

It is believed that all three proposals can be simulated but might not be suitable for "real world" implementation. That is, there are already robots entities for swarms that are based on finite state machines [143]. However, particle based implementations are currently used in models and simulations only [5].

Finite State Machines

Finite State Machines and Probabilistic Finite State Machines are a common approach to the control of robotic swarm entities. When utilising Finite State Machines, a robotic swarm entity does not plan its future actions but basis its next action on its current state and the current state of its inputs, such as its sensors or received communications [8, 143, 144, 172]. The probability of transition between states can be fixed [175, 177], where a defined probability of change is used throughout the deployment and operation of a swarm, or it can be variable, using a mathematical function to define the probability of a change of state [52, 109, 203]. A typical response threshold function, that has been used to study collective behaviour [187] can be seen in Figure 2.7. The transition probability, p, depends on a stimulus, s, which represents the transition urgency, as governed by a threshold on the stimulus, θ, and a sensitivity parameter, β. In the example provided in Figure 2.7, $\theta = 500$ and $\beta = 8$.

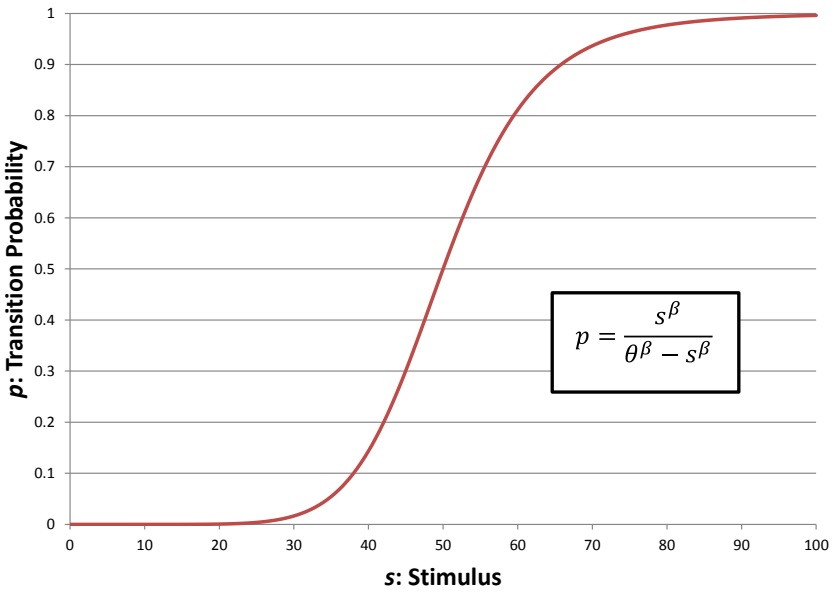


Figure 2.7: Response-Threshold Model [187]

Physics Based

In Physics Based models, each entity is regarded as a particle and the entities exert kinetic forces on other entities, in a Newtonian physics based approach [24]. These virtual physics based designs can use the concepts of artificial potential field, as described in an early work on the subject that was undertaken by Khatib in 1985 [108]. Khatib suggests the use of potential energy in the Lagrangian, where in classical mechanics, the natural form of a Lagrangian is defined as the Kinetic Energy, T, of a system, minus the system's Potential Energy, V [132]. A Lagrangian can therefore be define as: $L = T - V$.

The most commonly used artificial potential is the Lennard-Jones potential function [24]. This is shown in Figure 2.8, where the potential, v, is dependent upon the separation distance, d, between two entities. The depth of the potential function corresponds to ε and σ is the desired separation distance between two entities. In Figure 2.8, $v = 2.5$ and $\sigma = 0.3$. The Lennard-Jones potential function is a model that approximates the interactions between a pair of neutral atoms or molecules and was first described by Lennard-Jones in 1924 [97].

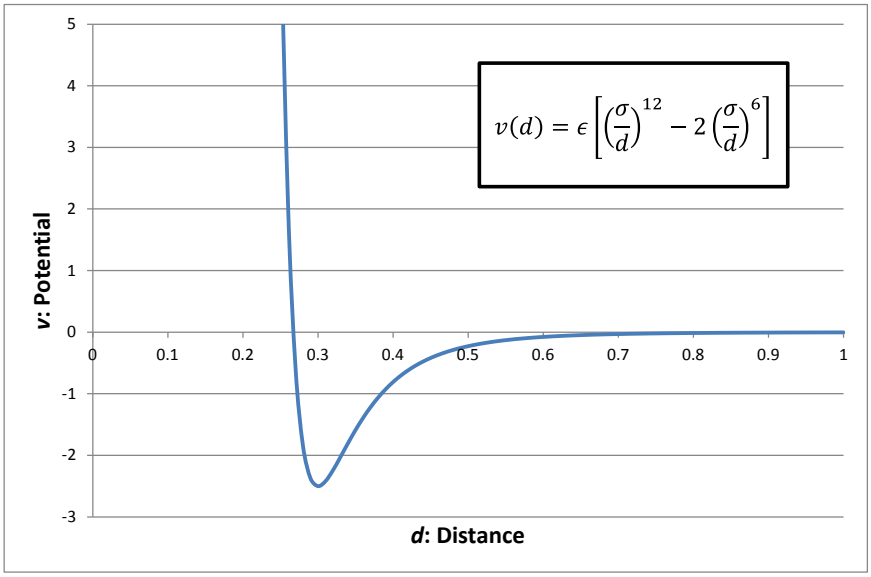


Figure 2.8: Lennard-Jones Potential Function

Bayindir discusses the concept of *artificial physics*, using virtual forces between swarm entities to determine the movement of the entities. This approach considers the interaction of swarm entities and their operating environment and can be used to model how many animal formations are observed in nature. Essentially, attractive forces are used to bring entities together and repulsive forces are used to prevent collisions, if they become to close [16].

Particle Based

Particle based modelling is similar to the kinetic physics based approach, in that the entities are treated as gas particles, and uses kinetic theory [15, 106]. However, the entities are assumed to have no mass and the entities are modelled as an ideal gas and have no potential energy, therefore a system consists entirely of kinetic energy. Collisions, such as with other entities or the operating environment are modelled as purely elastic collisions that maintain conservation of momentum.

2.6 Methods of Robotic Swarm Operations

2.6.1 Navigation

Many techniques have evolved in nature, in order to allow swarms to navigate within an environment. Research has endeavoured to better understand the mechanisms at work, especially as many of the techniques appear to utilise mechanisms that are very efficient and tailored to particular environments. There are several techniques that are used by nature that could be utilised by a robot swarm. A simple example is that of how ants decide upon which path to take in order to get to a food source [22, 23]. The ants are given a choice of two different paths between their nest and the food source, one of which is twice as long as the other, as shown in Figure 2.9.

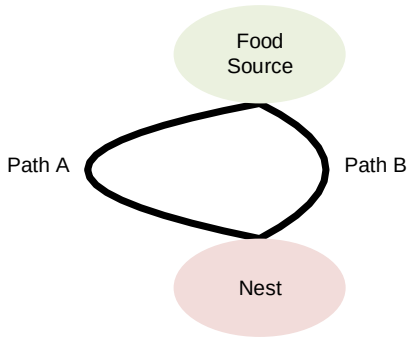


Figure 2.9: Two Different Length Paths

Initially, the ants travel down both paths, laying a pheromone trail as they bring food back to the nest. However, over a period of time the shorter path becomes the preferred path, as approximately twice the amount of pheromone is laid down by the ants. That is, in the time it takes an ant to travel the longer path, Path A, another ant can travel twice along the shorter path, Path B. Therefore, twice as much pheromone is laid on Path B when compared with Path A. The increase of the pheromone density on the shorter trail then encourages more ants to choose the shorter path, which in turn increases the amount of pheromone present on the trail. Experiments have also shown that the pheromone trail does have a limited life, due to environmental effects, such as rain, and the fact that the pheromone naturally evaporates away over time.

Certain ants also have the ability to detect when they have turned through more than 90°, relative to their starting orientation [22]. This is shown in the change to Path B in Figure 2.10. When this is observed during path finding, the ants can use this as a trigger mechanism to retreat along the initial route to the starting point and then seek another route.

Robotic swarms could use a similar mechanism when attempting to locate potential paths to follow. However, this could be detrimental to the efficiency of the robotic swarm as a shorter path might be discounted due to its orientation.

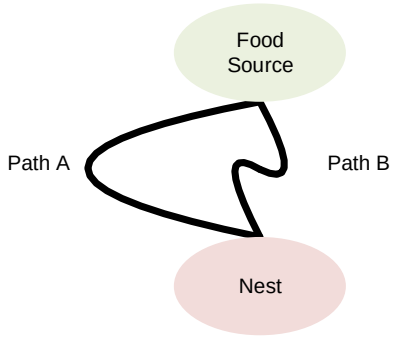


Figure 2.10: Forced Return Path

For a robotic swarm to carry out tasks to achieve a goal, the robotic swarm has to be able to navigate around their environment. Robotic swarm navigation can take place with either fixed references within the environment, or robotic swarm entities can navigate in respect to each other. Nouyan et al. described these approaches as employing distributed control mechanisms that mostly rely on local information [142].

They suggest that the approaches can be roughly viewed as two categories of distributed multi-agent path planning, where the path is formed by a network of immobile devices or where the robots serve as landmarks or beacons themselves. When the path is formed by a network of immobile devices, the devices are placed either a priori, at fixed positions, or by the robots themselves.

They investigated several methods, including when the robots serve as a landmark or a beacon themselves. These methods include *chain forming*, which can either be where every robot in the chain emits a signal, which indicates their position in the chain and every robot has to discriminate between as many signals as there are robots, or the robots rely on regular physical contact to adjacent robots, in order to maintain the chain.

Another method they considered was that of *gradual expansion*. These are several approaches to gradual expansion, one option is where a group for robots are gathered in one place and they gradually expand their locations outwards, whilst maintaining contact with their nearest neighbour. In order to maintain local communications, one robot broadcasts a signal locally and periodically, whilst the other robots move at random. When a robot does not perceive a signal any more, it retreats to re-establish the contact. It then becomes a static beacon itself, thereby expanding the network. Another form of gradual expansion is the gas expansion model that leads to a uniform distribution. This is where a group of robots spreads in the environment using simple expansion/repulsion mechanisms.

The final method they considered was that of *guided growth*. In this method, one robot is selected as the leader and the other robots follow the leader. In this way the robot structure stretches to form a line.

Rubenstein et al. considered the use of *edge following* and *gradients* during their research of robot swarm self assembly [158]. They developed a swarm entity that they entitled "Kilobot", which measures 33mm in diameter and can communicate up to around 70mm, utilising reflected infrared light off the operating ground surface [157]. The principles of edge following can be seen in Figure 2.11. The moving entity, shown in red, maintains a fixed distance 'd' to the centre of the closest stationary entity, shown in green.

Their proposed principle for gradients, shown in Figure 2.12, is that an entity sets its value to one greater than the minimum value of its neighbours that are within a distance of 'g', with the source entity always maintaining a value of 0.

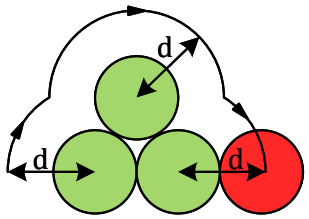


Figure 2.11: Swarm Entity Edge Following Fixed Entities [158]

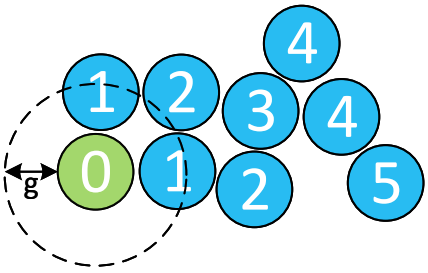


Figure 2.12: Swarm Gradient Formation [158]

2.6.2 Communication Methods

The chosen method of communications to be used between robots varies with the required goal of the swarm. This can be related to the physical size, constraints and the actual design and implementations of the robotic swarm entities.

The current forms of proposed communications between robots concentrate on optical, in the form of visible LEDs and infra-red LED communications, Radio Frequency (RF) communications, both near field and far field RF communications, physical connections between robots and the use of stigmergy. There are advantages and disadvantages to the various forms of communications and it might be advantageous, in certain circumstances, to utilise multiple techniques.

As an example, the I-SWARM project proposes near field RF communications and LEDs to communicate between robots [204]. Others utilised LEDs within the visible frequency range, utilising different colours to represent different exchanges of information, gain an understanding of the orientation of other robots, as shown in Figure 2.13, and the interactions between robots [142]. As detailed previously, the swarm will tend to communicate within an area local to the transmitting robot. This, again, has advantages and disadvantages. The main advantage is that it allows the overall bandwidth for the swarm to effectively be increased, as multiple robots can be transmitting on the same frequency but not interfere with each other. However, this frequency re-use might require a management overhead. Similarly, the use of LEDs can be utilised in such a way that the receiving robot will only react to local robots and are not capable, or choose to ignore, distant robots [142].

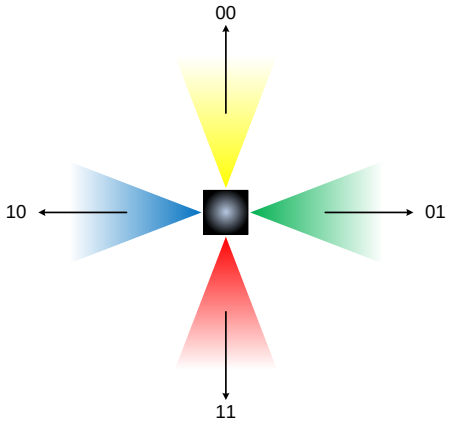


Figure 2.13: Using Colour to Communicate and Indicate Orientation to Other Robots.

Due to the limited range from an individual robot, messages that are required to be "broadcast" within the swarm often have to be forwarded by individual robots [114].

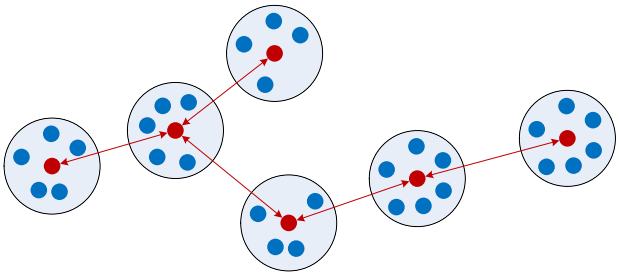


Figure 2.14: Short Range Local and Long Range Communications

A concept of forwarding was proposed within the MinuteMan project at UCLA. Within Minuteman, each group with the swarm communicates with a backbone network using the assistance of a dynamically elected leader, which they call the "Landmark". The election algorithm is constantly running in the background and if a landmark fails, it is immediately replaced. This is shown in Figure 2.14.

2.6.3 Thresholds and Decision Making

Research suggests that in nature, swarms utilise thresholds to assist the swarms in decision making [144]. A simple example of this is the *Pheidole genus* ant [201]. The colony is made up of minor and major ants. The minor ants effectively "do the house work and feed the larva", whilst the major ants hunt for food and defend the nest. The split is approximately 70% minor to 30% major. However, if enough minor ants are removed from the nest, a threshold is reached where major ants begin performing minor tasks, in order to maintain the mechanisms of the nest. There are suggestions that all the individual ants are based around the same underlying process and that they have been physically adapted to assist in their principle role. Within nature, the swarm can alter thresholds by responding to stimuli, a simple example to illustrate this is that of the honey bee, where, when required, a honey bee will perform the task of an "undertaker", when stimulated by dead nest mates [22]. The dead nest mates provide a stimulus to the undertaker honey bees to remove the corpses to a distance away from the bee hive. The more dead bees then the greater the stimulus and, similarly, the removal of the corpses reduces, or removes, the stimulus.

Within robotic swarms, similar proposals have been considered. The primary role of a robot swarm entity might be to locate and move an object back to a certain location. However, once a robotic entity has located the required object, it will attempt to recruit other robotic swarm entities, in order to move the object. Once enough robotic swarm entities have been recruited to be able move the located object, the combined robotic swarm entities will then undertake the task of moving the object to the required location.

There are also suggestions that robotic swarm entities can become specialised, with particular tasks becoming more emergent, or that task allocation can become adaptive [56]. All robotic swarm entities start the same level of response to an external stimulus and the more often the task is performed by a robotic swarm entity, the lower the level of stimulus that is required to trigger that robotic swarm entity to undertake that task, essentially the threshold for the particular task is lowered. Similarly, a robotic swarm entity that is not undertaking a task will increase its threshold to that task. Therefore, the stimulus required to trigger the robotic swarm entity to undertake that particular task increases. However, in order to ensure that a robotic swarm will still function, the higher thresholds will still trigger the required response if they are met, such as a reduction in robotic swarm entities with a lower threshold. This could be caused, for example, by removing them from the robotic swarm. That is, if the lower-threshold robots undertake the required tasking, the high-threshold robots will never be triggered into undertaking the task. If, for whatever reason, the threshold of the higher-threshold robots is met, then the high-threshold robots will react accordingly, including lowering their threshold for the task. This can also assist with the division of labour within a swarm, ensuring that not all the swarm members attempt to undertake the same task and therefore other required tasks will still be undertaken.

2.6.4 Self-Healing and Self-Reproduction

There have been schemes proposed where the robots are effectively capable of self-healing and self-reproduction, such as the use of autonomic computing and by the cannibalism of dead or damaged robotic swarm entities [45]. The proposal involves the robotic swarm entities performing continuous background "health checks" upon themselves as individuals. If a robotic swarm entity suspects that it is developing a fault, it will either attempt to solve the issue by itself, or it will call on other members of the swarm for assistance. The previous research suggests a variety of mechanisms and are best described by an example.

It is suggested that the individual robotic swarm entities within the swarm will perform system health checks as they go. If an issue is suspected, the robot swarm entity will attempt to self-diagnose and if it cannot determine an appropriate course of action, or the course of action requires intervention from another robotic swarm entity, it will request assistance from another robotic swarm entity. The initial robotic swarm entity might also do this if the suspected course of action could render the initial robotic swarm entity inoperative. An example of this could be the initial robotic swarm entity performing a reboot, that it might never recover from, or it might require assistance to prevent an even greater issue from occurring, such as the second robotic swarm entity guiding the robot during a reboot sequence to, prevent it from colliding with other robotic swarm entities. This could be a genuine problem if the robotic swarm is operating in the environment of space.

Courses of action range from allowing the problem robotic swarm entity to self-diagnose and attempt to correct the issue, such as by stopping and restarting software routines, through to allowing another external robot to essentially shut down and rebuild the problem robot, such as by reinstalling the problem robotic swarm entity's software and firmware builds via the external robotic swarm entity.

The effect of performing a software rebuild upon a robotic swarm entity within a swarm also raises other interesting problems. Consideration would need to be given to the task if it was suspected that the problem robot has information that is important to the completion of the swarm's tasking and goal. Also if the failing robot is currently performing a task, which other robots will take over this task? There might also need to be consideration of what actions to undertake if this is a common-mode failure, which has not occurred in the other robots yet.

A suggestion put forward by Dai et al., as to the process to follow if this happens is as follows [45]. Steps of curing:

1. After diagnosis, a "prescription" is determined.
2. Healing begins on the patient-robot, according to the steps/conditions of the prescription.
3. If the patient-robot is almost down with poor performance, a neighbouring robot is selected to remotely cure the patient, based on the prescription.

4. In the worst case scenario, the patient-robot is totally down or its communications fail to work, a neighbouring robot will be assigned to physically connect to the patient robot, in order not to abandon it. There is a socket designed for physical connection. After connection, the good robot can copy everything out of the patient-robot and then try to cure it via the prescription.
5. If the problem is identified and solved, it is reported and then the two robots are separated. Otherwise, if the patient-robot cannot be cured and dies, the good robot that is connected to the "corpse" checks the "organs". The healthy robot discards the unwanted or failed organs, to decrease weight, whilst retaining useful organs to enhance computing/storage ability or power for the good robot. The scavenged organs can also become a source for self-reproduction.
6. A failure in one robot can possibly occur in another robot, due to common mode failure. Therefore, after a successful cure for a robot, the corresponding failure and cure procedure should be broadcast within the swarm, to inform other robots of the required process for checking/debugging the related faults.

This raises other interesting problems, such as what to do with the reprogrammed robotic swarm entity, such as tasking? Dai et al. suggest that once the corresponding software from the "parents" has been copied to the "offspring", the parents then have to "educate" the offspring, by inputting the parents' experience, and indoctrinate the offspring into conducting the swarm's mission [45].

The proposals also provide an interesting discussion on how to cope with hardware issues and discusses the topic of self-reproduction. Their paper suggests that there are major challenges with self-reproduction. These challenges include how to ensure parents can generate the component parts, that are required to produce the next generation, when is the time to start the self-reproduction processes and how do parents educate the offspring to work for the mission?

They propose that the requirement for self-reproduction is determined by whether the system needs to add new members, such as if there were not sufficient robotic swarm entities in order to finish a mission, and whether there are sufficient resources to produce a new member robotic swarm entity.

The biggest issue facing the robotic swarm is probably how to source hardware components, to allow self-reproduction. Their paper suggests that the sources of components could be from either dead robots, form backup parts that are located on the robots or from dividable body parts, which are component parts that can be divided and then formed into the original part, such as stretchable antennas.

The process of self-reproduction is then essentially to assemble a framework for the appropriate parts, around which the parts themselves are assembled. The corresponding software is then copied from the "parent" to the new "offspring", and the parent then "educates" the offspring, by inputting the parents' experiences.

Hardware and software failures are therefore interlinked. If a robotic swarm entity suffers a catastrophic software failure, that cannot be repaired or successfully rebuilt, or the robotic swarm entity suffers a catastrophic hardware failure, such as the power source failing, then the robotic swarm entity is essentially dead. The proposal is therefore, in principle, similar to insects. A "healthy" robotic swarm entity attains a threshold that indicates there is a dead, or dying, robotic swarm entity. The healthy robotic swarm entity performs an "undertaker" role and moves the dead robotic swarm entity to a "cemetery". In the insect world, an insect will notice a dead insect and remove it from the immediate area. Ants initially move the corpse to a random location and then further dead ants are placed on the pile [22]. Bees appear to pick up the dead bee and remove it from the nest and drop it some distance away [22].

The reason for wanting to relocate dead robotic swarm entities from within, or removing from, the robotic swarm is proposed for several reasons. If the dead robotic swarm entity is left within the robotic swarm it might adversely affect other members, as other members might attempt to interact with it, or the other swarm entities might not be able to recognise or perceive the presence of the dead robotic swarm entity and therefore collide with it. The removal of dead robots is therefore seen as a way of helping to preserve the robotic swarm. The rationale behind relocating the dead robotic swarm entities within the robotic swarm is that it might be seen as more beneficial to keep the dead robots local to the swarm, in order to allow other robotic swarm entities to cannibalise the dead robotic swarm entities at a later date. This is best described by an example. If a robot swarm is performing a deep space exploration mission [76, 125] and several years into the mission a robotic swarm entity develops a fault, component parts of that robotic swarm entity might be extremely useful to the rest of the robotic swarm, such as power cells, fuel propellant, processors, solar panels, etc. It might, therefore, be beneficial to consume an amount of fuel propellant by a healthy robotic swarm entity, in order to move the dead robotic swarm entity out of the robotic swarm but allow it to follow at a safe distance. This is effectively forming a "scrap yard" of spare parts and a cemetery of dead parts.

2.7 Proposed Applications for Swarm Robotics

As swarm robotics is still very much within the research environment, actual applications for the use of swarm robotics are still very theoretical. However, it is the proposed uses of robotic swarms, and the potential benefits of swarm robotics could deliver, that is driving the majority of the research.

This section will give an overview of proposed applications for robotic swarms, including both civilian and military applications.

2.7.1 Maintenance Tasks

The I-SWARM project have proposed the use of swarm robots to be used in closed environments for maintenance tasks, such as monitoring component wear within engines and mechanical plant, where down time of the equipment for routine maintenance can be both time consuming and costly [204]. The robotic swarms are used to monitor components and report if they suspect an inconsistency. As an example, if a robotic swarm entity suspected a bearing running hotter than it should do, other robotic swarm entities would be recruited, in order to also investigate and confirm the potential issue. If an issue was suspected, the robotic swarm would ensure that a report is communicated, via a communications point, to the "Outside World", in order to pass the information on. The advantage of using swarm robots is that the robotic swarm entities would be small enough, so as not to interfere with the system under observation, and there are no single points of failure within the robotic swarm. That is, if one robotic swarm entity senses a problem, others will investigate to confirm that it is a component, or the system, at fault and not the individual robotic swarm entity. Where as, if an individual sensor were to be used, there could easily be a single point of failure. The robotic swarm entities can also investigate areas within the system that would otherwise not be able to be monitored, due to reasons such as the location of component parts within a system, where a system designer does not want to run a cable to a location, just to monitor an item, or in an otherwise purely mechanical system with no electrical or electronic component parts.

Swarm robotics could also be used on legacy installations, where the operator would like to monitor the system but does not wish to retro-fit the system with the necessary sensors.

This concept could easily be extended to other system maintenance tasks, where the system under observation is either difficult, time consuming or dangerous to reach. Typical examples of this are within pipelines, nuclear power plants and chemical tanks. The robotic swarm entities could also be made small enough and from materials that would not cause damage to the system under observation and could be used as a commodity item. That is, the robotic swarm could be used to inspect aircraft jet engines [43]. Once the inspection had been conducted, any robotic swarm entities that were not recovered would be constructed such that they would not cause any subsequent damage to the jet engine.

2.7.2 Communications Providers

The University of Essex has conducted research on the potential use of robotic swarms for Mobile Cluster Computing (MCC). They began work on the Flying Gridswarms project [150] and then followed this with the UltraSwarm project [86].

The original concept behind Gridswarm was the use of Unmanned Aerial Vehicles (UAVs) that are linked by short-range high-bandwidth wireless networks and configure themselves as a distributed parallel computer. The computational resources would work together to process sensor information, gathered by the robotic swarm, and direct the robotic swarm's collective behaviour. As an aside, they also provide an interesting fact that a typical flock of starlings (about 2,000 birds) contains as much brain tissue as a single human.

As well as looking at airborne assets, they were also interested in the interaction with UAVs and ground based assets, using the UAVs to direct the ground assets and process any sensor information from the ground assets. If required, the UAVs would also pass this information back to a central repository.

The Gridswarm project then moved to the UltraSwarm project, where small micro and mini rotary wing UAVs are being proposed to work as a robotic swarm, for the purpose of problem solving in order to fulfil the required tasking of the robotic swarm. Similar to the Gridswarm project, the UltraSwarm project's vision is to enable a group of UAVs to control their own motion and wirelessly network the UAVs together, in order to form a single, and powerful, computing resource.

They suggest that typical advantages of this proposed application are: multiple simultaneous viewpoints of targets, quick surveying of large areas, robustness through potential redundancy of individuals and the small size of the micro and mini rotary wing UAVs.

2.7.3 Emergency Response

Many of the proposals for swarm robotics are in the field of search and rescue and disaster recovery. These have been broken down to aid in understanding.

2.7.3.1 Ad-hoc Communications

An example of this that proposes a communications application is the SMAVNET II project, by Laboratory of Intelligent Systems at Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland [110]. Their aim is design a swarm of Micro Air Vehicles (MAVs), that are capable of autonomously establishing emergency wireless networks (SMAVNETs) between multiple ground-users in a disaster area. Their requirement is to be able to deploy the robotic swarm in any environment to support the disaster recovery.

The MAVs are designed to be minimal and should be low-cost, low-weight and have simple electronics. It is worth noting that, in the case of SMAVNET, they do not propose the use of any positioning sensors, such as cameras or GPS, due to size weight and power constraints, as well as the cost and the usability within certain environments. They therefore propose that the robotic swarm entities rely on local communication with their immediate neighbours and proprioceptive sensors, such as compasses and gyroscopes, which will provide heading, speed, altitude and angular velocities.

2.7.3.2 Contaminant Tracking

There are proposals to use robotic swarms to track chemical contaminants, such as a toxic release. These can range from observing vegetation, such as plant life along a river, through to flying through a suspected cloud of contaminant [180]. In the example of searching for or tracking a chemical cloud of contaminants, as shown in Figure 2.15, the aim of the robotic swarm is to provide situational awareness of an airborne contaminant release. In the proposed scenario, multiple UAVs fly to the suspected point of the contaminant and, if they find it, will act as a robotic swarm to track and map the contaminant cloud.

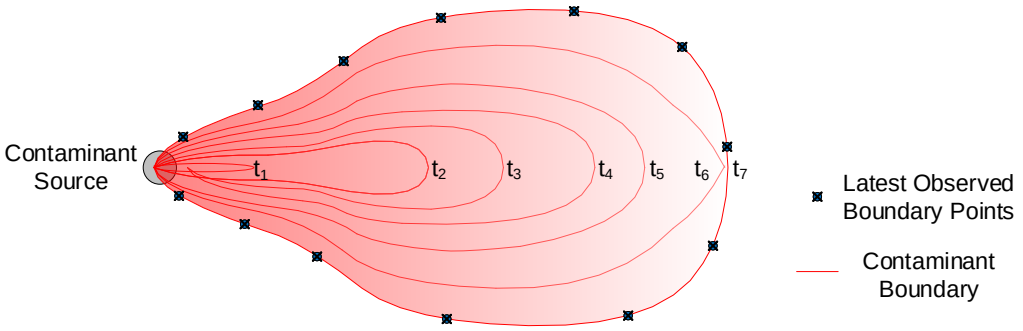


Figure 2.15: Tracking a Contaminant Cloud

There have also been proposals for locating, tracking and subsequent clean up operations for oil spills at sea [101], including a nanomaterial robot by MIT that can absorb oil up to twenty times its own weight [126].

2.7.3.3 Search and Rescue

The GUARDIANS (Group of Unmanned Assistant Robots Deployed In Aggressive Navigation by Scent) is a European project developing and applying the concept of autonomous robots in urban search and rescue operations. The project examined possible forms of interaction between fire-fighters and a robotic swarm, with the aim being to assist fire-fighters by working cooperatively with them [135].

Their proposal was to employ two types of swarm:

1. First employ a robot swarm to gain essential information about the incident, prior to engaging with it.
2. Secondly, employ a robot swarm as an aid to fire-fighters, once they engage with the incident.

Their suggested ways in which a robot swarm may assist fire-fighters are as follows:

1. Notify the fire-fighters of possible hazards such as obstacles, high temperatures and chemicals;
2. Indicating, unambiguously, the direction to the scene of the incident or the direction to the exit point;
3. Grouping – it is important for fire-fighters that the swarm stays within a relatively close distance to the fire-fighters to allow them freedom of action.

Their work focused on the specific search and rescue activity of fire-fighting in a large warehouse and specific advice was provided by the South Yorkshire Fire and Rescue Service.

Their work finished at the completion of the funding, although subsequent articles were published regarding their work [147].

In 2013, a multinational group began work on the SHERPA project [123]. The group included universities, industrial partners and an end-user of the research. The aim of this project was to utilise multiple ground and airborne robots that would collaborate with humans in order to improve search and rescue activities in Alpine environments. They envisage swarms of SHERPA teams collaborating towards a common task, such as search and rescue or patrolling a dangerous area [123]. The concept of SHERPA is to deploy a heterogeneous team of robotic platforms to interact with and compliment the human rescue activities, both within both the summer and winter conditions of the Alpine environment [29, 30, 122].

2.7.4 Use of Swarms in Space

The NASA Goddard Space Flight Centre has been actively researching the use of robotic swarms in space, in a programme called Autonomous Nano-Technology Swarm (ANTS) [76, 125]. The ANTS programme proposes the use of robotic swarms in space exploration and has identified their use in several scenarios. The scenarios include lunar missions, including their use in exploration, communications and instrumentation, a mission to survey Saturn and a mission to prospect asteroids. The proposal for the prospecting mission of asteroids is comprised of a swarm of 1000 robotic swarm entities, with robotic sub-swarms of 10 different robotic swarm entity types with 100 robotic swarm entities in each, allowing the robotic sub-swarms to specialise in the required tasking.

The robotic swarm entities within the ANTS programme, called TeTWalkers, are comprised of 12 tetrahedrons, providing them with multiple degrees of freedom and having the ability to roll, climb and flatten. Their proposal is for multiple tetrahedral to be combined, in order to form continuous and, if required, complex structures. An example they provide of this could be a parabolic dish antenna, which is formed from several different robotic swarm entities. This could be an example of where various robotic sub-swarms would work together to reach an end goal. One robotic sub-swarm would form the structure of the antenna, whilst a members of another robotic sub-swarm would form the RF modules and another robotic sub-swarm would provide the power source for the transmissions. As the robotic swarm entity is designed to be self-healing, the robotic swarm entity will discard and replace broken parts of its structure.

The command and control of the ANTS robotic swarms is to send it a command, such as move across the lunar surface to a given point, using radio frequency (RF) communications. However, how the TeTWalker robotic swarm entities reaches the required goal will not be specified to the robotic swarm. The robotic swarm will be fully autonomous and will have decision making capabilities, in order to complete the task and to overcome any problems or issues that are encountered on the Moon's surface.

Separately, there has been a proposal to use a robotic swarm that have mirrors to alter the course of the Near Earth Asteroid (NEA) *Apophis*. A study in 2007 by Professor Massimiliano Vasile at the University of Glasgow, found that a collection of mirrors in space, located on and controlled by a robotic swarm, could focus the Sun's energy onto the surface of the asteroid in order to vaporise part of its surface. The flow of gas produced would propel the asteroid onto a new orbit [120]. For the context of the reader, Apophis is a 210-330 meter asteroid, that on the 13th April 2029 will come relatively close to Earth, it will miss Earth by about 22,000 miles. It will come past the Earth again, in 2036, and it was initially calculated that there is a small possibility, about 1 in 45,000, that it could be on a collision course, due to further observations and improved computational techniques this was recalculated to about 1 in 250,000 [26, 46, 134]. Upon its discovery in 2004, Apophis was briefly estimated to have a 2.7% chance of impacting the Earth in 2029, the impact has since been ruled out due to additional observations [26].

2.7.5 Military Communications

The Centre for Autonomous Intelligent Networks and Systems (CAINS), University of California, Los Angeles (UCLA), has been conducting research with the US Office of Naval Research (ONR). Their research, entitled the MinuteMan Project, and is essentially the concept of the "internet in the sky" [75]. They are researching the concept of Multimedia Intelligent Network of Unattended Mobile Agents, with the aim of providing efficient, reliable, low latency communications between members within a team, between separate teams and between command posts.

Their proposal is for a mobile architecture to connect air, ground and maritime assets, with each of the environments providing different challenges for the robotic swarms to operate within. Because of this, they propose different robotic swarms for each of the different environments, with communications between them.

They propose that the robotic swarms have the requirements for individual robots to be able to communicate within swarms, for swarms to be able to communicate between each other, for robotic swarms to be able to communicate to external command posts and for the robotic swarms to be able to communicate with sensors that are deployed separately, but within, the swarms' operating environments, which could be both ground or sea environments.

Their proposal is for a high bandwidth backbone network, consisting of several UAVs. This backbone network is dynamically and autonomously reconfigured to maintain robotic swarm contact and connects to various airborne and ground based assets, in order to provide the required connectivity. However, if part of the airborne backbone is disabled, their aim is for the network to re-route communications via ground assets, although this will be at a reduced performance, communications will still be maintained.

2.7.6 Underwater Mine Countermeasures

In the recent conflicts, robotic vehicles have been used to undertake tasks, such as surveillance, reconnaissance and mine searching. These are tasks that could be repetitive and are dangerous in their nature. It is suggested that utilising robot swarms in a hostile environment may well be the new concept for war, with robotic swarms operating alongside the war fighter in air, land and naval operations [75, 184].

Research has been undertaken into the possibility of performing underwater mine countermeasures, utilising swarm robotics [184]. In this proposal, the control of the swarm uses a hybrid of combining behaviour-based and systems-theoretic approaches, in order to provide more functional control of the robotic swarm. The objective of the hybrid system is to manoeuvre and direct a robotic swarm effectively to the location of an underwater mine.

Their research used Artificial Potential Fields (APF) as a means to generate robot behaviours [24]. Where the APF methods attempt to use simple laws of nature, attractive and repulsive potentials, to draw or repel an object, a robotic swarm entity, from one location to another. The attractive potential results in a rapid motion towards the separation destination if the distance is large but decreases gradually as the robot approaches the final position, finally stopping at the target. The repulsive potential performs the opposite. That is, as the robotic entity moves towards an obstacle, the magnitude of the distance between the two points becomes smaller, approaching zero. As a result, the repulsive potential increases as the distance to the obstacle closes, effectively reaching infinity when the robot is close to the object.

This infinite repulsive vector guarantees that there can be no collision, unless some other vector also increases unbounded. Because of this, it is common in motor schema systems to generate potentials that are all upper-bounded with the sole exception of the obstacle avoidance routine. The principle can been see in Figure 2.16.

Their control architecture utilises a subsumption based approach, where the priority of tasking is defined. In this instance the priority is:

1. Avoid mines.
2. Avoid obstacles.
3. Aggregation and separation.
4. Random search pattern.

The environmental information triggers the appropriate response of the robotic swarm entity, in deciding on whether to suppress the lower order behaviours. An example of this would be if the swarm had determined that it needed to avoid a mine, it would also avoid obstacles. However, it might not perform a random search pattern, as the robotic swarm entity had effectively found a mine, by attempting to avoid it, and therefore would undertake the task appropriate to locating a mine, such as inform the robotic swarm operator of the find.

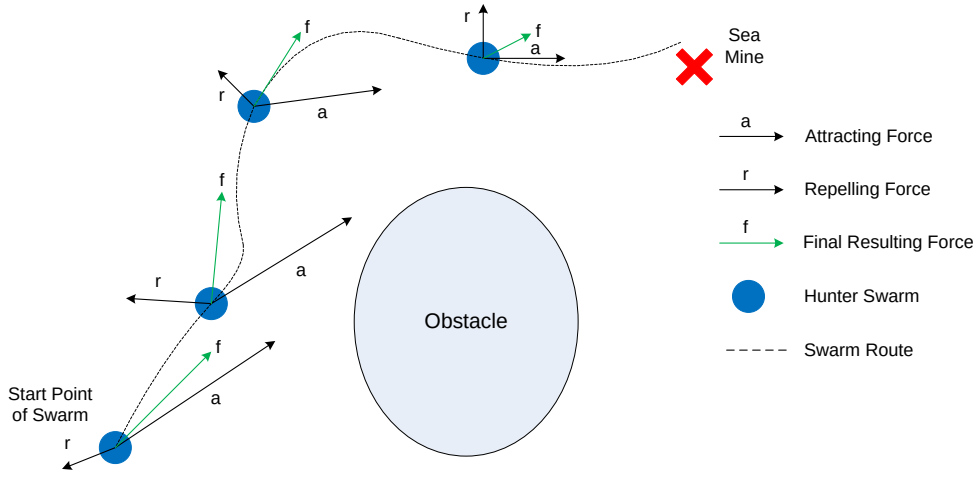


Figure 2.16: Underwater Swarm Movement

2.7.7 Landmine Detection

There is research investigating the possibility of using robotic swarms to locate landmines [34, 112, 113, 114]. The amount of time available to detect and make safe a region that contains landmines will depend upon circumstances at the time. If it is a war fighting situation, where troops are actively involved in a battle, it might be appropriate to either just mark the locations of the landmines or mark a safe passage through the landmines, as opposed to making the landmines safe.

This will allow the troops pass through the mined area, as this might be a lower risk to life and a quicker process than making the landmines safe, which would be pertinent if the troops are under enemy fire. However, if the activity was to clear the landmines, such as after a conflict for humanitarian reasons or troops transiting the area after a battle, then the objective might be to not only locate the landmines but also to make them safe, by either defusing or destroying the landmines.

The suggestion in the research is for a two-fold approach to the landmine detection problem [34]:

1. Undertake an effective foraging strategy, such that the robotic swarm entities can detect the landmines within an operational area.
2. When a landmine is located, utilise a mechanism to recruit other robotic swarm entities that are currently foraging, in order to make the landmine safe.

Their proposal for the foraging task was based on visual clues and memorisation, in an attempt to not go over previously inspected locations within the operating environment and so as to ensure all ground is covered [22, 114]. Their research was for a robotic swarm entities to initially detect and locate a landmine, followed by the recruitment of other robotic swarm entities to assist in the defusing of a landmine, to make the landmine safe. This process would then repeat, in order to locate and make safe other landmines within the operating environment.

The researchers made the assumption that a particular number of robotic swarm entities were required at the location of the landmine, in order to defuse the landmine. No details were provided, or assumptions made, as to how the landmine would be made safe. It should also be noted that their research did not consider any reduction in robotic swarm entities. That is, it is assumed that no robotic swarm entities will be destroyed by the landmines within the operating environment and once a landmine has been made safe, those robotic swarm entities will then attempt to search and locate other landmines.

When a robotic swarm entity located a landmine, the entity would change from a foraging mode to a waiting mode, as it waited for other robotic swarm entities to join it at its location. When enough robotic swarm entities were present, they would then defuse the landmine. During the waiting process, the robotic swarm entity would produce a pheromone to attract other robotic swarm entities to the defusing task. A robotic swarm entity that was in a foraging mode that detected the pheromone would enter a scent following mode, until it reached the landmine. Where it would either enter a waiting state, or enough robotic swarm entities would be present in order to commence rendering the landmine safe.

This can lead to several interesting issues, such as if two landmines are located in close proximity to each other. As robotic swarm entities around each landmine produce pheromone to recruit other robots to assist in defusing the landmines, the pheromone concentration between the two landmines could be of a level that forces robotic swarm entities to believe that they are within a landmine area.

This will constrain the robotic entity to both stop following a pheromone scent and prevent it from performing its own normal foraging. However, this potential denial of service to the robotic swarm entity should be relatively short, because as the actual landmines are defused, the pheromone concentration will diminish and the robotic entity will begin to forge again. This can be seen in Figure 2.17 and was reported within the original research.

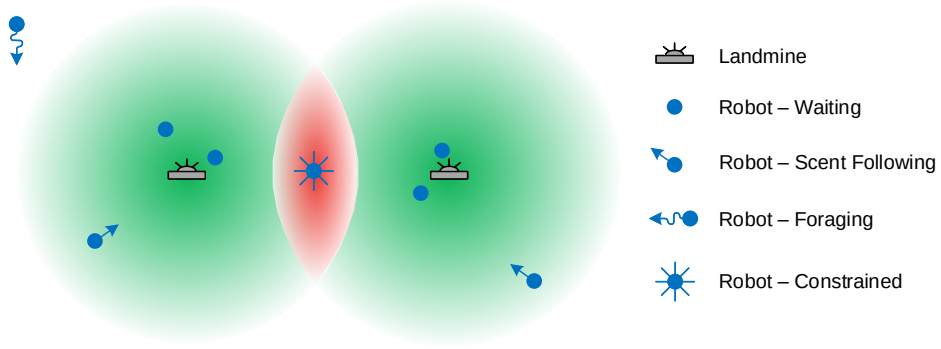


Figure 2.17: Local Scent Maximum

Their research also reported that the number of landmines to number of robotic swarm entities is important, as when the robotic swarm entities wait at the landmines for the prerequisite number of robotic swarm entities to arrive to defuse the landmine, it is possible for all the other robotic swarm entities to also be waiting at other landmines [34]. This caused all the robotic swarm entities to be waiting at landmines and the system basically entered a state of "dead-lock", where the robotic swarm entities would wait for an infinite time for others to arrive. They called this a *frozen state* and a suggested solution by the researchers to this problem was to suggest that a robotic swarm entity always moves away from a pheromone scent boundary, therefore only robots within the actual area of a scent release will become scent following robots. A foraging robotic swarm entity that is not within the area of a scent release would turn away from any scent boundary that it tried to travel through and would continue to forage for undiscovered landmines. This also makes the assumption that the scent can move faster than a robotic swarm entity, which would be possible if the implementation utilised a radio frequency beacon, as opposed to an actual physical scent.

Another suggested solution to this problem was that a robotic swarm entity would randomly "time-out" from a waiting state and return to a foraging state. The important factor here would be that the robotic swarm entity would have to wait until the pheromone level had dropped to an appropriate threshold level before it is reset back to a foraging state, which would then allow it to enter a scent following state. In their experiments, they essentially made the robot move away fifteen feet, which just over twice the robot's scent detection range, which was seven feet. This was in order to prevent the robotic swarm entity from immediately reacquiring the initial pheromone scent.

Eventually, one of the landmines would be defused and those robotic swarm entities would then be released back to a foraging state, which would then assist other robotic swarm entities that were in the frozen state, waiting for other robotic swarm entities.

2.7.8 Situational Awareness

The US Army Research Laboratory established a Micro Autonomous Systems and Technology (MAST) Alliance [151] , in order to perform research to enhance the war fighter's tactical situational awareness, in both urban and complex terrains, such as caves and mountains. This was initially a five year program that was then extended to ten years and ran from 2008 until 2018. Upon the completion of MAST, the US Army began a new collective technology alliance, know as Distributed Collaborative Intelligent Systems and Technology (DCIST) [48].

The objective of these research alliances is to provide situational awareness capabilities to the war fighter, which are currently either impossible or dangerous to obtain. Their requirements are for solutions that can work with humans but operate with little or no direct human supervision in a variety of environments. The research is being conducted on the development of both small air and ground robotic swarm systems for the acquisition, dissemination and delivery of improved situational awareness [18].

2.8 Discussion

As stated in Section 1.2, the characteristics of swarms are generally agreed to be that the swarm has a large number of members, that operate autonomously with decentralised control, leading to an emergent collective behaviour and swarm intelligence. The individual members have a local sensing capability and can only communicate locally, to either other swarm members or the immediate environment.

Tan et al. provided a table that demonstrates different characteristics between robotic swarm characteristics and the characteristics of similar systems [183]. This can be seen in Table 2.1.

	Swarm Robotics	Multi-robot System	Sensor Network	Multi-agent System
Population Size	Variation in great range	Small	Fixed	In a small range
Control	Decentralised and autonomous	Centralised or remote	Centralised or remote	Centralised or hierarchical or network
Homogeneity	Homogeneous	Usually heterogeneous	Homogeneous	Homogeneous or heterogeneous
Flexibility	High	Low	Low	Medium
Scalability	High	Low	Medium	Medium
Environment	Unknown	Known or unknown	Known	Known
Motion	Yes	Yes	No	Rare
Typical Applications	Post-disaster relief	Transportation	Surveillance	Net resources management
	Military applications	Sensing	Medical care	Distributed control
	Dangerous applications	Robot football	Environmental protection	

Table 2.1: Comparison of Swarm Robotics and Other Systems [183]

It is recognised that within actual implementations of robotic swarms, there might be a requirement to communicate to the robotic swarm as a whole. That is, a controlling release authority of a robotic swarm might wish to stop a robotic swarm from operating or attempt to influence the robotic swarm's behaviour.

Examples of this could be for safety reasons. If a robotic swarm were undertaking a search and rescue task and the swarm began to potentially endanger possible victims, such as by disturbing a debris field within a disaster area. The controlling authority might then wish to either stop the robotic swarm from operating or command the robotic swarm to move to a particular location, such as the original release site. This could be undertaken by the robotic swarm's controlling authority, which is external to the released robotic swarm, undertaking a broadcast to command the entire robotic swarm.

Similarly, if the operators believed the task to be complete, they might command the robotic swarm to leave an environment. An example of this could be in a fire-fighting task, if the role of the robotic swarm were to assist fire-fighters in moving around a smoke filled building to recover people and all the people had been recovered from a building, then there is no need to continue the search and therefore the robotic swarm can be removed. The reason for removing the robotic swarm might not only be due to the goal of the robotic swarm being completed but also to remove the robotic swarm from a potentially hostile environment and therefore preserve the robotic swarm entities for future use.

A robotic swarm entity might also wish to communicate beyond its local environment, such as when the overall robotic swarm's goal has been achieved. An example of this might be if a robotic swarm locates a missing causality within a search a rescue undertaking. This could be for several reasons, to inform the controlling authority of the success and possibly to inform the other entities within the swarm to change their goal from searching and to move to a known location for recovery, such as the original release point.

However, within this research the individual swarm entities will be constrained to the capabilities and limitations as previously described in Section 1.2.

2.8.1 Capabilities and Limitations of Swarm Entities

As discussed in Section 2.3, the capabilities and limitations of a swarm entity depended upon its design and subsequent uses. That is, certain devices, such as the e-puck2, are probably capable of implementing security functionality, either onboard the host device or with the use of an additional module. Other devices, such as the Kilobot, would not be capable of implementing security functionality.

The research into the use of nano-bot technology currently does not consider the utilisation of security functionality within the robot designs. The designers are concentrating on the miniaturisation and potential medical benefits and possible issues, such as ensuring that the body does not reject the devices and preventing the body from attacking the devices as foreign objects.

The characteristics of typically available swarm robots are provided in Annex B.

This research is interested in the deployment of swarm's where their associated swarm entities have insufficient processing power to undertake security functions, where the swarms are deployed within hostile environments, such that the swarm entities are subject to attack, and where the swarms follow the characteristics, as described in Section 1.2. Therefore, this research assumes that other means of protecting the swarm and its swarm entities from potential attack are required.

2.8.2 The Term Swarm

It is also important to emphasise that the term swarm is often used with different means or contexts, when compared to how swarms are considered in this research. The term swarm is often used in the context of an Unmanned Airborne Systems (UAS), specifically the Unmanned Airborne Vehicles (UAVs) of the UAS. In order to ensure safety of flight of a UAV, it is often in the control of an operator, which does not meet swarm characteristic 1, the requirement for autonomy for a swarm, as detailed in Section 1.2. A UAV can be programmed to take-off, undertake predetermined activities and then land at a known location, which does not meet swarm characteristic 4, the requirement for the swarm to exhibit a collect emergent behaviour. An example of this is was research being in a project entitled MedizDroids, where various types of UAVs and UASs where being considered in the use of mosquito control [4]. The UAVs would automatically undertake operations in order to spray indoor and outdoor surfaces with a larvacide. Although this could utilise swarm principle in the future, the term swarm in this instance was essentially a large amount of UAVs undertaking the task.

Similarly, the term swarm is often used within the military context. In this instance the term can relate to a swarm in the context of this research, such as *Perdix* micro-drones research being conducted by the US Department of Defense, which are released in swarms and work collectively and autonomously to achieve a common goal [192, 193], or again it can relate to a large amount of individual items. Such as a swarm of UAVs or a swarm of aircraft, where the UAVs or aircraft are being controlled by either another system or a pilot.

However, this research will focus on the original characteristics of the robotic swarm and assume that once a robotic swarm has been released then there will be no external controlling authority that can communicate with the released robotic swarm.

Part II

Contributions

3 Generic Models for Swarms

3.1 Introduction

This chapter provides details of a generic model that enables the representation of swarms, the swarms' operating environments and malicious intruders within the swarms. The models are based on the swarm characteristics, and their operating environments, that were presented in Chapter 2.

There has been a significant amount of research undertaken in to the proposed uses [169, 180, 204] and physical implementations of swarms [14, 39, 66, 74, 124], along with the associated modelling [21, 25, 52, 53, 68, 81, 128, 195, 196] and simulation techniques [20, 47, 119, 199]. However, to date, none of the models or simulations appear to have considered security implications for a swarm, which is possibly due to the relative infancy regarding the subject of swarms and their proposed applications. The current research, including models and simulations, assume trust of the interacting swarm entities as a pre-requisite condition. The current physical implementations have also been undertaken in relatively benign conditions. However, this research is concerned with swarms which are designed to operate in hostile environments, when considering security.

In this chapter, I propose a generic model for a swarm and introduce the concepts of modelling swarm entities and malicious intruders within the swarm [164]. This research only considers "healthy" swarm entities, failed swarm entities had been considered but it was felt that this would detract from the focus of this research and has therefore not been detailed. There had already been research undertaken within this area [38, 203]. Generic models are also provided for the environment, in which the swarms and their entities operate within.

This chapter proposes how an external hostile swarm, specifically with a malicious intent, could affect an original swarm, how this could be modelled and provides examples of how malicious swarms could effect a victim swarm.

This chapter also introduces two swarm use-cases that will be used throughout the remainder of the thesis, these are detailed in Section 3.5.

3.2 Generic Models

The following sections provide generic models for an overall swarm, a malicious swarm and the swarms' operating environment, including their various attributes. The generic models also take account of the swarm entities, which operate within the various swarms, and the entities individual attributes.

3.2.1 Initial Swarm Definition

The initial swarm is defined as a large number of entities that have attributes and will exhibit actions based on interactions with other swarm entities and the environment in which the swarm operates [160].

The *original swarm* is defined as a set of entities S^o, such that:

$$S^o = \{s_0^o, s_1^o, \ldots, s_n^o\} \tag{3.1}$$
$$= \{s_i^o\} \tag{3.2}$$

Where: $i = 0, \ldots, n$

Where n is the number of entities within the swarm and each swarm entity, s_i^o, has the same specific attributes.

The attributes will be specific to the design of the swarm and will contain information that enables the swarm to operate. Typical examples of attributes could be the swarms location, velocity and time variables.

The *attributes* for the *original swarm* S^o are defined by:

$$SA^o = \{sa_0^o, sa_1^o, \ldots sa_z^o\} \tag{3.3}$$
$$= \{sa_i^o\} \tag{3.4}$$

Where: $i = 0, \ldots, z$

Where sa_i^o is an attribute of the swarm S^o and z is the number of attributes specific to the swarm.

Typical attributes of a swarm could be the maximum amount of time available in order to complete a task. The swarm itself may have certain attributes that relate to the goal of the swarm, such as searching for a target or recovering an object. Therefore, the goal of a swarm can be an attribute of a swarm.

It is also worth noting that different swarms may interpret the attributes based on their particular design. That is, a location attribute might be a latitude and longitude measurement, or it might be distances relative to an arbitrary point or datum.

Similarly, time could be based on the Universal Time Constant (UTC) or seconds elapsed since the swarm was released.

The attributes for a *swarm entity*, s_i^o, are defined as:

$$RA^o = \{ra_0^o, ra_1^o, \ldots, ra_y^o\} \tag{3.5}$$
$$= \{ra_i^o\} \tag{3.6}$$

Where: $i = 0, \ldots, y$

Where ra_i^o is an attribute for all the swarm entities s_i^o and y is the number of attributes specific to the swarm entities.

Typical attributes of the swarm entities could be the goals of the individual swarm entities, such as maintain communications with neighbouring swarm entities and locate a specific object, or physical attributes of the swarm entities, such as current location, velocity and energy levels.

It should be noted that the attributes of a swarm entity, RA^o, are not necessarily the same attributes as the overall swarm itself, SA^o. This model also allows for consideration of similar properties, that could have different interpretations. A typical example of this could be a time goal. That is, a swarm's goal could be to arrive at a location by a particular time. However, the swarm entities might have the goal to arrive at a location in the most efficient manner, that can be perceived by the swarm entities, as deduced by the "swarm's intelligence" [34, 37, 85, 96, 112, 117, 161, 188]. However, the swarm, S^o, might not be able to arrive at the location by the required time, such as due to environmental effects, but will arrive there eventually. Therefore, both the swarm goal and the swarm entities goal are related to time, but have different contexts.

3.2.2 Modelling a Malicious Swarm

We define a malicious swarm as a number of swarm entities that will attempt to effect the behaviour, or performance, of the original swarm.

As the original swarm has a goal, the malicious swarm also has a goal. For example, the original swarm might have the goal of clearing a minefield and the malicious swarm could have the goal of attempting to prevent, or hinder, this from happening. Another example could be that an original swarm might have a goal to travel to a specific location. The malicious swarm might have the goal of either preventing this from being achieved, or to slow the original swarm down, in order to effect the efficiency of the original swarm.

The *malicious swarm* is defined as a set of entities S^m, such that:

$$S^m = \{s_0^m, s_1^m, \ldots, s_w^m\} \tag{3.7}$$
$$= \{s_i^m\} \tag{3.8}$$

Where: $i = 0, \ldots, w$

Where w is the number of malicious entities within the malicious swarm.

In a similar fashion to the original swarm, the malicious swarm has a number of attributes.

The *attributes* for the *malicious swarm* are defined as:

$$SA^m = \{sa_0^m, sa_1^m, \ldots, sa_v^m\} \tag{3.9}$$
$$= \{sa_i^m\} \tag{3.10}$$

Where: $i = 0, \ldots, v$

Where sa_i^m is an attribute of the malicious swarm S^m and v is the number of attributes specific to the malicious swarm.

The *attributes* for a *malicious swarm entity*, s_i^m, are defined by:

$$RA^m = \{ra_0^m, ra_1^m, \ldots, ra_l^m\} \tag{3.11}$$
$$= \{ra_i^m\} \tag{3.12}$$

Where: $i = 0, \ldots, l$

Where ra_i^m is an attribute of the malicious swarm entity and l is the number of attributes specific to the malicious swarm entity.

The attributes for the malicious swarm will be dependent upon its design. Both swarms could have the same attributes but set to different values. For example, both the original swarm, S^o, and the malicious swarm, S^m, could have a maximum velocity limit. However, the malicious swarm might have a greater maximum velocity limit, in order to be able to catch and affect the original swarm.

It is also possible for an attacker to capture and modify an original swarm entity, in which case the attributes of the original swarm entity and the malicious swarm entity would be the same. However, the attacker may be able to modify the captured original swarm entity attribute values, in order to allow this to become a malicious swarm entity and attempt to undertake an attack.

An example of a swarm entity's attribute that could be modified might include the permitted time allowed to undertake a task, where the original swarm entity might have a predefined time on a task, in order to allow it to return to a location before its energy is depleted. The malicious swarm entity might be altered to ignore this attribute and just keep working until its energy is depleted and it fails. This would be to enable the malicious swarm entity to have the maximum possible amount of time attempting to affect the original swarm, before its available energy is exhausted.

The capture and subsequent reuse of modified swarm entities might assist an attacker in realising the threat of masquerade. That is, because the attacker is utilising original swarm entities for the attack, the original swarm entities would perceive the malicious swarm entities as legitimate members of the original swarm and would interact with them accordingly. This is further discussed in Section 4.2.2.

3.2.3 Modelling Swarm Interaction

In order for multiple swarms to be able to interact, their swarm entities will need to share common attributes. For instance, in order for the malicious swarm S^m to be able to interact with the original swarm S^o, then entities or S^m will have to share common attributes with the entities of S^o.

That is, in order for different swarms to interact, it is the swarms' entities that interact. In order for this to happen, the swarm entities will need to share some common attributes. In the example of S^o interacting with S^m, some of the attributes RA^o will be shared with RA^m.

This can be seen in Figure 3.1, where RA^o and RA^m share certain attributes.

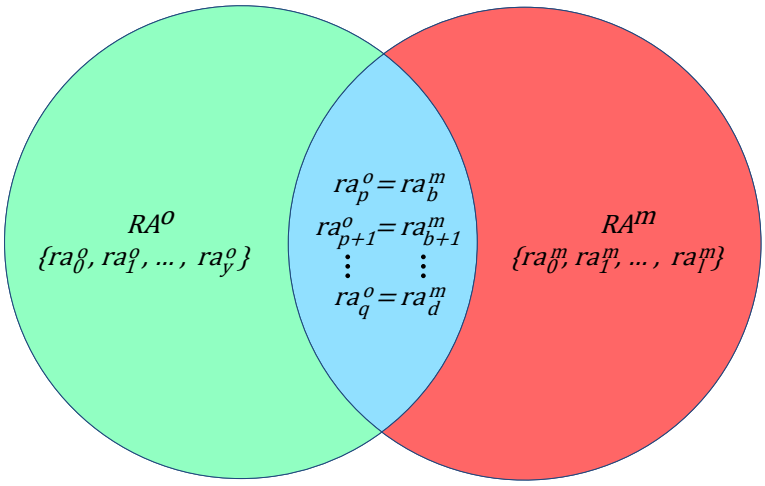


Figure 3.1: Venn Diagram Showing Relationship between the Attributes of the Original Swarm Entities and the Attributes of the Malicious Swarm Entities

In this example, the common attributes of the two swarms are shown in the intersection of $RA^o \cap RA^m$ of the Venn diagram:

$$\{ra^o_p, ra^o_{p+1}, \ldots, ra^o_q\} = \{ra^m_b, ra^m_{b+1}, \ldots, ra^m_d\} \tag{3.13}$$

The aim of the malicious swarm, S^m, is to alter the behaviour of the original swarm, S^o, in the most efficient manner. This could be by the malicious swarm entities utilising as many common attributes as possible, in order to affect the original swarm, or by utilising common attributes that have the greatest effect.

Another consideration is the size of the malicious swarm that is required to alter the behaviour of the original swarm. The ideal situation for the malicious swarm would be that the malicious swarm has significantly fewer swarm entities than the original swarm.

3.2.4 Modelling the Operating Environment

The environment in which the swarm operates needs to be defined, in order to allow the effect of the environment to be included in the modelling of a swarm. This will provide the ability to model environmental effects and interactions of the swarm with its operating environment. This can including swarms that communicate utilising the environment, such as by the use of stigmergy, where modifications are made to the local environment in order to communicate. The following section provides a model for the environment in which a swarm, S^o, and its entities, s^o_i, operate within.

The environment, E, in which a swarm operates within is defined as:

$$E = \{e_0, e_1, \ldots, e_d\} \tag{3.14}$$
$$= \{e_i\} \tag{3.15}$$

Where: $i = 0, \ldots, d$

Where e_i represents individual elements of the operating environment, such as the current area of operation, and d denotes the scope of the operating environment that is being modelled.

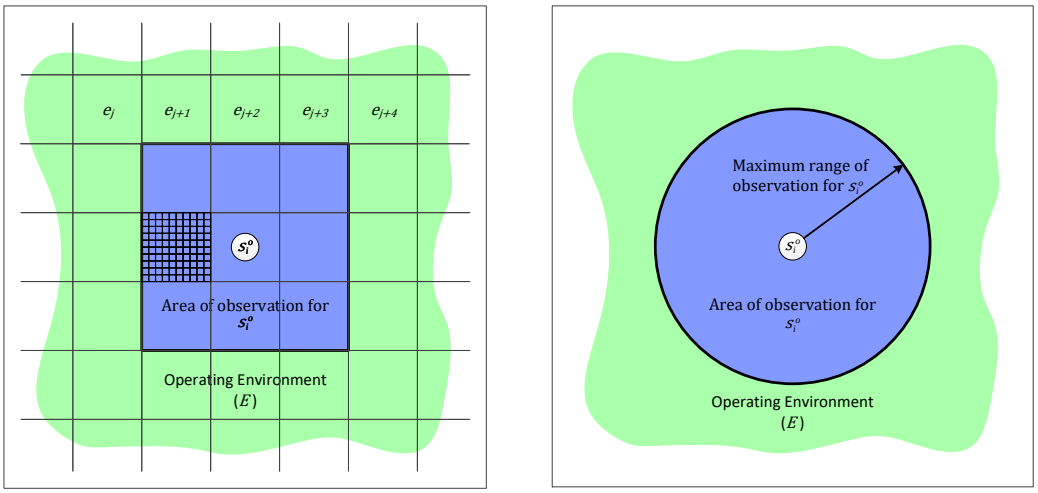
The attributes for the operating environment e_i are defined by:

$$EA = \{ea_0, ea_1, \ldots, ea_g\} \tag{3.16}$$
$$= \{ea_i\} \tag{3.17}$$

Where: $i = 0, \ldots, g$

Where ea_i is an attribute of an operating environment element e_i and g is the number of attributes specific to the operating environment.

Several methods of implementing a swarm model can be undertaken, such as modelling the environment, E, as a series of squares, as shown in Figure 3.2a, or as vectors and distances from an entity, as shown in Figure 3.2b.



(a) Modelling as Squares

(b) Modelling by Distances

Figure 3.2: Modelling the Operating Environment

The implementation of the model could also exhibit dynamic qualities. That is, the model might have to react to changing characteristics. To place this into context, a swarm, S^o, is operating within an environment, E. The goal of S^o is to locate and follow a chemical trail, such as pheromones or smoke. If a swarm entity, s_i^o, detects the presence of the chemical, it increases the accuracy of its sensor readings and therefore increases the granularity of its sensing of the local operating environment, e_i. The effect upon the model is shown in Figure 3.2a, where the area of observation for the swarm entity, s_i^o has been further divided into smaller areas.

3.3 Modelling Swarm Behaviour

There are several ways in which the model can be refined, based upon the requirements of the model of a specific swarm implementation.

3.3.1 States Within a Swarm

The state of a swarm can be viewed from several perspectives. It is proposed that this can be best explained if it is broken the question down into a hierarchy of three separate parts:

1. The state of the goal of the swarm.
2. The state of the swarm itself.
3. The state of a swarm entity.

The overall swarm could be considered as operating asynchronously, as the swarm entities are operating independently without a common clock source and their interactions will be based upon many variables, such as the effects of the operating environment. However, it would be fair to assume that the swarm entities themselves should remain stable, so as not to adversely effect the overall swarm. That is, if a swarm entity were to become unstable, it could act in such a way so as to have a detrimental effect and impact upon the overall swarm behaviour.

There are two classic types, or models, of finite state machines. These are the *Mealy Finite State Machine* and the *Moore Finite State Machine*.

Due to the characteristic that a Moore finite state machine is stable, because of the synchronous implementation provided by the clock, it is proposed that swarm entities are viewed from a Moore finite state machine perspective.

Mealy finite state machine outputs can be directly manipulated from the input. The asynchronous nature of the Mealy finite state machine does enable a quicker response time than a Moore finite state machine but this can lead to a more unstable response.

The actual state machine utilised would depend upon the design and implementation of the swarm and could, in theory, be view from either context.

3.3.1.1 The State of the Swarm's Goal

The state of the swarm's goal is the overall state of what the swarm is attempting to achieve. To place this into context, imagine a swarm of bees. In this simple scenario, in order to illustrate the concept, the bee swarm can have one of two goals, either foraging for food or defending the bee hive.

The normal behaviour of the bee swarm is to forage for food. However, if the bee hive comes under attack, the swarm will alter its behaviour to defending the bee hive, until the threat has passed, when it will return to foraging for food. The overall states of the bee swarm can therefore be one of two states, either foraging or defending.

In reality, only the bees that are in the local environment of the bee hive would change their state to defending, due to the local production of pheromones. However, this example is provided for illustrative purposes only.

The state of a goal for a swarm will be dependent upon the objective of the swarm and its implementation.

If the objective of the swarm is to locate a target, say a missing person in a search and rescue scenario, the swarm states could be:

- **Searching** Attempt to locate the target.
- **Indication** Inform an external party that the target has been located.
- **Recovery** Return to a location for recovery.

The changes between states would depend upon the original configuration of the swarm. That is, if a swarm was in a state of indication, because it had located a target, it could subsequently either change to a recovery state, as there was only one target, or it could change to a searching state, as there are multiple targets. Similarly, the swarm could alter its state from searching to recovery after a set period of time, in order to replenish energy stores, or it could continue until it either changed state to indication, if the swarm located a target, or the swarm eventually fails and stops working, as its energy is depleted.

3.3.1.2 The State of the Overall Swarm

It is proposed that the state of the overall swarm is the current states of all the individual swarm entities attributes.

Although, in practice, this will be a finite number, the actual number will be too large and the problem essentially becomes intractable.

That is, say there is a swarm of only 10 entities, each swarm entity has 4 variable attributes and the attributes can only be binary in nature, i.e. 0 or 1. This would give rise to $\approx 1.1 \times 10^{12}$ possible states for the entire swarm.

Essentially, each entity has 2^4 possible states, which equals 16 possible states. As there are 10 entities this produces 16^{10} possible states, which equates to $\approx 1.1 \times 10^{12}$ possible states for the entire swarm

Adding one extra entity to the swarm raises this number to 16^{11} possible states, which equates to $\approx 1.76 \times 10^{13}$ possible states for the entire swarm.

Now let us assume that the swarm entity's attributes are not binary in nature, such as direction of travel might be in degrees referenced from North, i.e. 0 to 359 degrees, or position could be a GPS 10 digit grid reference. It becomes apparent that, based upon the actual implementation of a swarm entity, that the actual finite state of an entire swarm becomes an intangible problem to resolve.

Threshold techniques could be utilised to reduce the possible number of states but this would still produce a large number of possible states for the entire swarm. For example, the direction of travel could be reduced from 0° to 359° down to 4 possible states: North, East, South and West. However, this would still lead to a large possible number of states for the attributes when considering the entities of a swarm as a whole.

When considering the binary example of just two possible states per attribute for each swarm entity, if the number of swarm entities were to be increased to 1000, then this would equal 16^{1000} possible states, which equates to $\approx 1.3^{1204}$ possible states for the entire swarm. Altering just one of these states from binary to 4 possible states increases the possible number of states for the entire swarm to $(2 \times 2 \times 2 \times 4)^{1000}$ possible states, which equates to $\approx 1.4^{1505}$ possible states for the entire swarm.

3.3.1.3 The State of a Swarm Entity

The state of a swarm entity is the current state of all of an entity's attributes.

The swarm entity's attributes, and therefore its state, will change based on both internal and external influences.

A typical external influence could be detecting the completion of the entity's goal, such as locating a target. This would alter the entity's goal accordingly, such as from Searching to Indication of the located target.

A typical example of an internal influence for a swarm entity could be remaining energy levels. This could simply be the percentage of energy available to the individual swarm entity. When the energy levels reach a certain point, the state of swarm entity, with the goal of locating a target, could change from Searching to Recovery. This would allow the swarm entities to be recovered and then replenish their energy levels.

An example of a state of a swarm entity, that could be influenced by both internal and external factors, might be the location and movement information of the entity. The direction of travel of the swarm entity could be influenced by both environmental effects and the swarm entity's goal. For example, the goal of the swarm entity is to travel North to a pre-determined location. However, due to an obstacle within the environment, the swarm entity has to alter its course, in order to navigate the obstacle. Once the obstacle has been navigated, the entity then returns to a course that ensures that the swarm entity travels in a direction, with the goal of arriving at the pre-determined destination. Throughout this procedure, the location of the swarm entity has also been changing, based on the actions of the swarm entity in response to the environmental conditions.

3.3.1.4 Discussion on States Within a Swarm

It was proposed that, due to the complexity of defining the state of the whole swarm with a finite state machine, as the problem is essentially intractable due to the potentially large number of states that the investigations into swarm behaviours were undertaken utilising simulations, as opposed to mathematical methods.

The states of a swarm entity being represented by their attributes and the state of the swarm being realised by analysing and interpreting the results of the simulations in achieving its goal.

3.4 Modelling the States within Swarms

This section describes proposals for how the states within swarms can be modelled.

3.4.1 Modelling the States of Swarm Entities

The state of the attributes of swarm entities, at a particular moment in time, can be set, and modified, based on conditions within the model.

To place this into context for the reader, the following example considers one attribute of the swarm entity, specifically the attribute that relates to the goal of a swarm entity.

Therefore, let us assume that ra_0^o contains the state of the goal of a swarm entity, s_i^o, of the swarm, S^o.

Assuming that ra_i^o can have multiple goals, such as search for a target and report target located, ra_i^o can therefore have multiple states:

$$ra_0^o = \{g_1, g_2, \ldots, g_u\} \tag{3.18}$$
$$= \{g_i\} \tag{3.19}$$

Where: $i = 0, \ldots, u$

Where g_i is one of the possible states of ra_0^o, the goals of the swarm entity, of which there are u possible states.

Therefore, for a swarm entity, s_i^o, the current state of it goal, ra_0^o, can be determined by:

$$ra_0^o = \begin{cases} g_1 & if \quad \text{condition 1} \\ g_2 & if \quad \text{condition 2} \\ \vdots & \vdots \quad \vdots \\ g_u & else \quad \text{condition } u \end{cases} \tag{3.20}$$

Where each condition, c_i, relates to the goal, g_i, and evaluates to either *true* or *false*.

That is:

$$c_i = \{ \ TRUE, \quad FALSE \ \} \tag{3.21}$$

and

$$c_i = f(\ \text{“}Interactions \quad with \quad neighbours\text{”}\ ) \tag{3.22}$$

That is, c_i is calculated from a function that is based on the swarm entity's interactions with its neighbouring swarm entities, that are within its local sensing capabilities and therefore have an influence upon the swarm entity.

If s_i^o can interact with m other entities, then S_m^o is the subset of S^o that s_i^o can interact with.

Therefore, the function that is used to calculate the conditions can be further refined to:

$$c_i = f(RA_i^o) + \bigcup_{j=0}^{m} f(RA_j^o) \tag{3.23}$$

That is, c_i is calculated from a function that takes into account the swarm entity's own attributes, RA_i^o, and then takes into account the attributes of the neighbouring swarm entities, the swarm entities s_0^o to s_m^o.

The function within the model could be implemented such that weighting factors are utilised to enhance the model. That is, a swarm entity that is further away from the original swarm entity under consideration, s_i^o, could have a lesser effect than a swarm entity that is closer.

The function is finally modified to take into account any effects from the local environment that can have an influence on s_i^o:

$$c_x = f(RA_i^o) + \bigcup_{j=0}^{m} f(RA_j^o) + \bigcup_{k=0}^{l} f(EA_k) \tag{3.24}$$

Where l represents the amount of environment that can have an effect on the swarm entity, s_i^o.

3.4.2 Modelling the State of a Swarm

Following from the modelling of the current states of individual swarm entities, the model can be further refined in order to model the state of the overall swarm.

As the current state of a swarm entity is a combination of a swarm entity's attributes, the current state of a swarm entity s_i^o, at time t, can be considered as RA_i^{ot}.

That is:

$$State_i^{ot} = RA_i^{ot} \tag{3.25}$$

The next state, at $t+1$, is therefore a combination of the effects of all the swarm entities within the swarm, S^o.

$$RA_i^{ot} \xrightarrow{f} RA_i^{o(t+1)} \tag{3.26}$$

The effects that the swarm entities have on each other is represented by the state transition function, f.

As the attributes of the swarm entities take into account the effects of their local environment, any local environmental that affects the state of s_i^o at time t are therefore considered within RA_i^{ot}.

Assuming n entities within S^o, this can be shown as:

$$State_i^{o(t+1)} = RA_i^{o(t+1)} = \bigcup_{i=0}^{n} f(RA_i^{ot}) \tag{3.27}$$

The function f takes into account the fact that certain entities might be positioned such that the entities do not have a direct influencing effect on each other. That is, if entities are not within a distance such that they can have an effect on each other, the function f will take this into account, such as by applying a weighting function that discounts any influence between these entities.

This allows the model to take into account effects upon a swarm entity that are beyond its scope of influence. This is shown in Figure 3.3, where swarm entity A is not directly influenced by swarm entity D. However, swarm entity D does have influences upon swarm entities B and C, which also have an influence on swarm entity A. Therefore, the behaviour of D does have an indirect influence on the behaviour of A, via B and C.

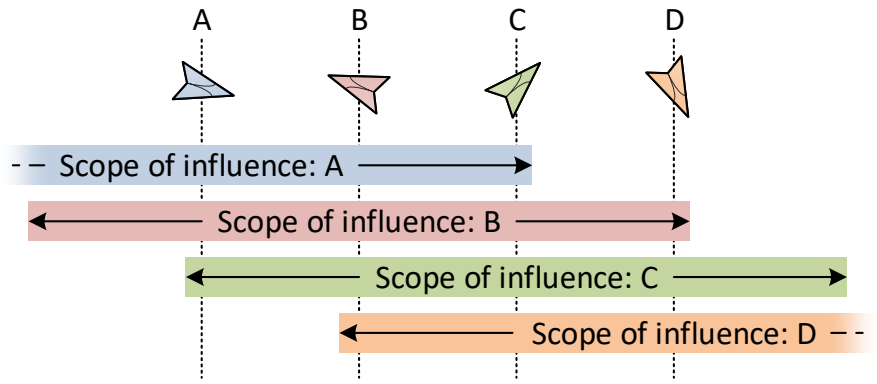


Figure 3.3: Influence of Swarm Entities

3.5 Typical Use Cases for Swarm Applications

The following examples provide a introduction as to how the models could be implemented, for typical swarm scenarios.

The following swarm scenarios presented continue through the thesis:

- **Use-Case 1 - Cooperative Navigation**
  A swarm is attempting to realise cooperative navigation by utilising local communications within the swarm. The example provided is based on previous research that attempts to locate a target within the operating environment.
- **Use-Case 2 - Foraging and Local Recruitment**
  A swarm is attempting to forage for targets and then utilise local recruitment methods to assist with a task. The example provided is based upon previous research that proposed these techniques to locate and make safe landmines within an operational area.

3.5.1 Case 1 - Hunting Example

In some applications it is proposed that there is a requirement for a swarm to hunt out and locate a target object over the shortest path, by utilising efficient navigation techniques. This example will be considered as *Use-Case 1* within this thesis.

In this case, the original swarm, S^o, attempts to find a target, T. The swarm entities, s_i^o, utilise the attributes distance to target and age of information data, in order to assist them in locating the target. A malicious swarm, S^m, could attempt to prevent S^o from locating T, or make the process of locating T less efficient.

The malicious swarm entities, s_i^m, could well have exactly the same attributes as s_i^o. However, the swarm entities within S^m modify the communicated values, in order to effect the overall efficiency of S^o. That is, $RA^o = RA^m$, however, attributes within RA^m communicate false information, such as ra_1^m could report false distance data and ra_2^m might report false information age data.

3.5.2 Case 2 - Landmine Example

In the following example, the goal of a swarm is to attempt to locate and then recruit other swarm entities, in order to make the landmines safe within a given operational area [34, 112]. The goal of the malicious swarm is to attempt to make the discovery of landmines an inefficient process. This example will be considered as *Use-Case 2* within this thesis.

Therefore, the original swarm attempting to locate the landmines, with say 500 members, can be shown as:

$$S^o = \{s_0^o, s_1^o, \ldots, s_{499}^o\} \tag{3.28}$$

This swarm, S^o, attempts to locate and make safe landmines by conducting a search within a pre-defined operating area, E. The attributes for the overall swarm:

$$SA^o = \{sa_0^o, sa_1^o, \ldots, sa_z^o\} \tag{3.29}$$

The swarm attributes could be: Search for landmines, make safe landmines and undertake the task within a particular time period.

The individual swarm entities, s_i^o, could have the following attributes:

$$RA^o = \{ra_0^o, ra_1^o, \ldots, ra_y^o\} \tag{3.30}$$

Where ra_i^o is the attribute of a swarm entity, from the original swarm. The swarm entity attributes could be: location; current velocity; maximum velocity; maximum communications range; and the remaining power levels.

In this case the goal of the malicious swarm, S^m, is to alter the behaviour of the original swarm, S^o, in the most efficient manner. Such that S^o will either fail, or reduce its efficiency, in realising its original goal of locating and making safe the landmines. That is, the goals of S^o and S^m differ, in that the goal of S^o is attempting to locate and make safe landmines, where as the goal of S^m to prevent S^o from efficiently undertaking this task.

The malicious swarm, containing say 100 entities, is defined as a set of entities S^m, such that:

$$S^m = \{s_0^m, s_1^m, \ldots, s_{99}^m\} \tag{3.31}$$

Where 100 malicious swarm entities are introduced as a malicious swarm.

In a similar fashion to the original swarm, the malicious swarm has a number of attributes. The attributes for a malicious swarm entity are defined by:

$$RA^m = \{ra_0^m, ra_1^m, \ldots, ra_j^m\} \tag{3.32}$$

There could be common attributes between $RA^o$and RA^m, the attributes for the swarm entities s_i^o and s_i^m, such as: location; current velocity; maximum velocity; and maximum communications range. However, the values of these attributes could be different, that is, s_i^m could have a greater communications range than s_i^o, in order to influence larger areas of E, that S^o is operating within. It might also have a greater maximum velocity, in order to be able to infiltrate S^o as efficiently as possible. Certain attributes between RA^o and RA^m will be different, as S^m will not want to detect mines but will want to prevent S^o from undertaking this. Therefore, s_i^m might not monitor its energy levels, as there is no reason to remove itself from the minefield.

3.5.3 Review of Case Examples

To assist in understanding, the examples are summarised in Table 3.1 and Table 3.2, and the attributes of the various swarm entities defined in Section 3.2.1 are described.

	Hunting (Use-Case 1)	Landmines (Use-Case 2)
S^o	Efficiently locate targets	Locate landmines Make landmines safe
S^m	Increase time to locate targets	Hinder and prevent landmine location

Table 3.1: Typical Attributes of a Swarm and a Malicious Intruder

	Hunting (Use-Case 1)	Landmines (Use-Case 2)
RA^o	Distance to target Age of information	Location Power Velocity
RA^m	Random distance to target Lower age of information	Location Power 2 x Velocity

Table 3.2: Typical Attributes of the Swarm Entities

As can be seen, when comparing the attributes of an original swarm entity s_i^o, RA_i^o, to the attributes of malicious swarm entity s_i^m, RA_i^m, the attributes can be either the same, modified or different. The actual attributes being dependent upon the implementations of the swarms. It can therefore be seen how the generic model can be applied to swarm implementations and applications. The modelling information can also be used to identify specific attributes that could be utilised, in order to attack S^o.

3.6 Summary

In summary, this chapter provides a generic model that allows the researcher to consider how an original swarm and a malicious swarm can be modelled, along with their operating environment. The chapter identifies how various swarm attributes could be manipulated to cause an effect, taking into consideration the effects of the operating environment for the swarms. The chapter also suggests how an external hostile swarm, specifically with a malicious intent, could affect an original swarm and how this could be modelled. The generic modelling technique allows for further simulations and analysis of the various swarm implementations, and possible attacks against these swarms, to be undertaken.

4 Taxonomy of Threats

4.1 Introduction

This chapter provides a taxonomy of threats to a swarm, based on a literature review of papers and standards. It is used within this research to aid in the understanding of threats posed against swarms when applied to the models presented in Chapter 3.

The majority of the literature describes threats and attacks against networks and services that are delivered via networks. This taxonomy will present these threats and attacks, detailing how they could be undertaken against a swarm and provide a description of how these threats and attacks can be modelled. It should be noted that the various threats described can be either a *primary threat*, such as gaining information or disrupting a process, or an *enabling threat* that assists in attaining the primary threat to a system.

The possible attacks towards and against an actual swarm will vary, based upon the swarm's implementation and its operating environment.

4.2 Threat Modelling

This section provides a taxonomy of threats that has been tailored for swarm implementations. That is, a review of potential threats has been undertaken and a discussion of how these threats might affect a swarm implementation has been provided.

Within the literature, there are several similar definitions of the term "threat". To aid in the understanding of the term threat, the definitions are as follows:

- BS ISO/IEC 13335 [28] defines threat as "a potential cause of an incident that may result in harm to an asset, system or organisation. This harm can occur from an attack on the information being handled by a system or service, on a system itself or on other resources, such as by causing unauthorised destruction, disclosure, modification, corruption, unavailability or loss". It also recognises that "a threat needs to exploit a vulnerability of the asset in order to harm the asset, threats may be environmental or human in origin and, with regards to the human, may be either accidental or deliberate".
- BS ISO/IEC 27005 [91] defines a threat as "a potential cause of an incident, that may result in harm of systems and organisation".

- The Federal Information Processing Standards (FIPS) 200, Minimum Security Requirements for Federal Information and Information Systems [63] defines threat as "any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service". Also, "the potential for a threat-source to successfully exploit a particular information system vulnerability".

A weakness of an asset, or a group of assets, that can be exploited by one or more threats is known as a vulnerability [28].

The following threats have been tailored for swarms from several threat taxonomies [64, 200] and reference the modelling work undertaken in Chapter 3, in order to place the threats into the context of the swarm models.

It is worth noting that the various threat types can be either, or both, a primary threat or an enabling threat. Where an enabling threat is utilised, to allow the primary threat to be realised.

The following describes the various threats to which a swarm could potentially be susceptible [166].

4.2.1 Denial of Service

The threat of a Denial of Service (DoS) is a threat "to prevent the availability, normal functionality and associated services of a system for their intended use by legitimate users" [64, 182, 190, 200]. DoS threatens the resources that prevent legitimate users from using, or depleting, those resources. Typical resources that could be subject to the threat of DoS are bandwidth, memory or processor resources and deny the services by causing devices to incorrectly function or corrupt information required for the correct operation of a system [107, 173].

The DoS threat to a swarm is dependent up the swarm's implementation and the type of DoS attack that is undertaken.

A swarm, S^o, can be subject to a DoS threat posed by a malicious swarm, S^m, if the goal of S^m is to prevent, or reduce, the availability of S^o, its services or its resources such as to effect the overall behaviour or efficiency of the S^o in fulfilling its goal.

A swarm S^o can also be subject to the DoS threat by external influences, from within its operating environmental, E. An external influence can be either malicious, such as man-made barriers or physical destruction, or benign, such as debris fields that was unknown at the time of the swarm release, but all have the effect of threatening a DoS on S^o.

4.2.2 Masquerade

Masquerade is the threat that an intruder or system entity illegitimately poses as, or assumes the identity of, another entity [64, 162]. This is in order to attempt unauthorised users to gain access to confidential data or greater access privileges, while pretending to be legitimate user [136].

It has been suggested that the threat from a masquerade attack is probably one of the most dangerous attacks on system integrity [146]. The attacker, by assuming the identity of fully legitimate user, is able to access the system and other significant user data and therefore cause significant amounts of damage. It is also recognised [64] that the threat from masquerade is such that it can be used as an enabling threat for other primary threats.

From the perspective of swarms, a swarm S^o is subject to the threat of masquerade, if a malicious swarm entities s_i^m are able to be considered as a legitimate members of the original swarm, S^o. As a swarm can be deployed in hostile locations that are not within the control of the swarm's releasing authority, a swarm entity could be recovered by an attacker and subsequently inspected. This could allow an attacker to gain knowledge of how swarm entities interact, in order to replicate characteristics of the swarm entity, or to modify and subsequently reintroduce the swarm entity to the original swarm. This would allow an attacker to operate within the original swarm, such that the original swarm entities believe the malicious swarm entities to be legitimate members of the original swarm.

4.2.3 System Penetration

The threat of system penetration is the unauthorised access to a system to modify system or application files, steal information or illegitimately utilise resources [64].

Within swarms, system penetration can be undertaken from within an entire swarm perspective or from an individual swarm entity level. The primary threat of system penetration can be enabled by the threat of masquerade.

Essentially, a swarm S^o is penetrated to enable unauthorised access by a malicious swarm S^m.

4.2.4 Authorisation Violation

Authorisation violation is the use of a system by an authorised user for unauthorised uses and possibly gain privileged access to system resources [64]. This can be undertaken by either an insider or external threat actor.

Within the theoretical implementation of a swarm, this threat may not be able to be realised, as there is no hierarchy within a swarm.

However, in a practical swarm implementation, a hierarchical structure might be implemented, or swarm entities with enhanced capabilities might be present, such as path finders to enable navigation. Also, in the case of "non-uniform" swarm elements, certain swarm entities might be more susceptible to the threat of authorisation violation, due to their interactions with other swarm entities and their operational environment.

The threat of authorisation violation is essentially that S^m would try and gain privileged access to the swarm in order to attempt to manipulate the swarm S^o in an unauthorised manner.

4.2.5 Planting

Planting is the threat that a malicious capability is planted within a system to perpetrate, aid or enable future attacks [64]. Variations of the planting threats are trojan horses and viruses.

Within the context of swarms, planting could take several forms. Typical examples of the threat of planting within swarms are the removal of a swarm entity that is taken from the original swarm S^o, which is then modified and subsequently reintroduced to the original swarm, but operates as an entity of a malicious swarm, S^m. Also, depending upon the influence and capabilities of an attacker, elements of the original swarm S^o could be modified during the production cycle or supply chain. Therefore, when a swarm S^o is deployed, it already contains a malicious swarm S^m, due to the supply chain attack undertaken against S^o.

Attacks could be introduced, planted, into an original swarm S^o, so as S^m behaves in a non-malicious way and interacts with S^o in such a way that S^o is unaware of S^m. That is, S^o believes the members of S^m to be legitimate members of S^o. However, when required, S^m can be triggered to act in a malicious way, such as by a command by an malicious attacker, a pre-determined physical location or a time based event, such as elapsed time.

4.2.6 Communications Monitoring (Eavesdropping)

The threat from communications monitoring is where an attacker, or intruder, obtains information by reading data sent between system participants, without penetrating the victims' participating entities [64]. An attacker can access the information by either monitoring the information transmitted between the system participants, or by compromising member participants of the network [6]. The attacker often monitors the data transmissions between devices for particular message content, such as authentication credentials or passwords [167]. The threat is not limited to user data but also extends to command, control and management information.

The threat of eavesdropping is generally perceived as a passive attack, where the eavesdropper conceals their presence from the participants within the system and uses only the broadcast medium to eavesdrop on all messages. However, active attacks have been suggested, such as to determine information by sending queries to system participants [6] and active RF attacks [33].

The threat of monitoring all the communications, within a swarm S^o, could take several forms. Examples could be a malicious swarm S^m follows, observes and monitors the communications of S^o, or swarm entities of S^o are modified to to become a malicious swarm S^m, in order to monitor the communications within S^o. The threat of communication monitoring could also be realised by an external entity or system that is not a swarm but has the capability to monitor the communications of S^o.

4.2.7 Modification of Data in Transit

The threat of modification of data in transit can be realised by actively modifying the communications, or the communications process, when sending data between authorised participants [64]. The threat could be directed at either, or both, user data and control information [104]. Essentially, the threat is that an attacker can alter a legitimate message by deleting parts, adding extra, changing elements or reordering it [167]. This is effectively compromising the integrity of the data or information being communicated and is conducted without penetrating a victim's system.

Within the context of a swarm, a malicious swarm S^m is interacting with an original swarm S^o, such that S^o is unaware of S^m. As S^m is able to monitor the communications of S^o, S^m has the ability to determine which messages to modify, in order to effect the actions of S^o. That is, if S^m detects a particular message, such as "target located", it could forward on a message of "no target located", or just not forward on the message. This principle can be seen in Figure 4.1 where a malicious swarm entity has positioned itself at an obstacle, in this example at a bridge over a river. The river acts as an obstacle for the free movement of the original swarm entities and therefore the communications are routed via a "choke point" across the bridge.

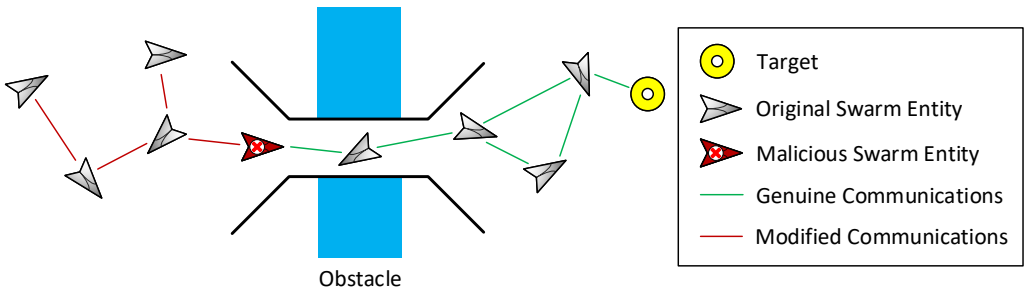


Figure 4.1: Modification of Communications at a Choke Point

However, if there is not a choke point, due to either the amount of original swarm entities or there was no obstacle, then the messaging would be communicated around the malicious entity. This is shown in Figure 4.2, where there is no obstacle affecting the communications paths.

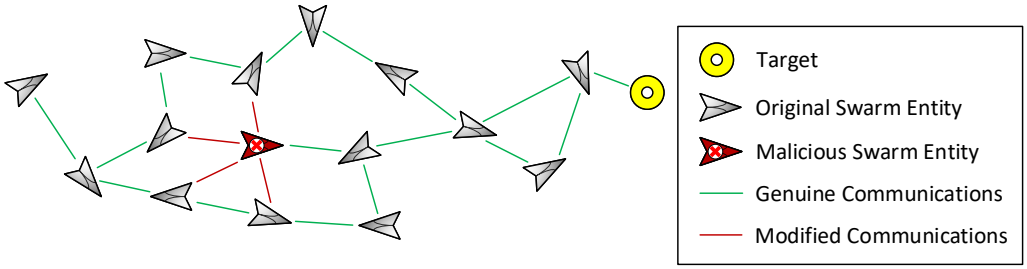


Figure 4.2: No Modification of Communications

Another method of modifying the communications data in transit is to provide information that appears to be "better" than the actual information. To place this into context, suppose the goal of the original swarm is attempting to locate a target, as shown in Figure 4.3.

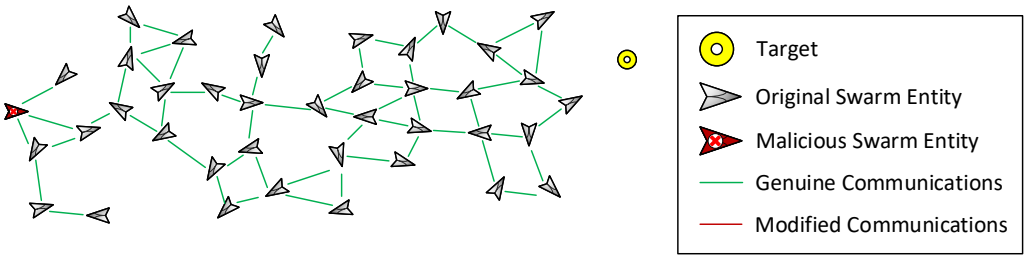


Figure 4.3: Swarm Searching for a Target

When the swarm locates a target, this is communicated throughout the swarm and information is provided to direct a search party to the target's location, as shown in Figure 4.4.

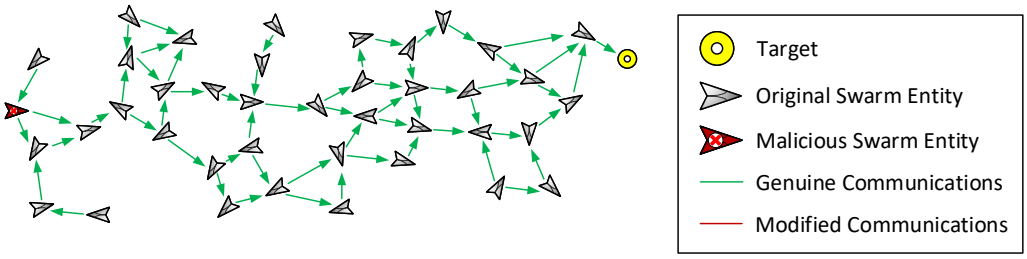


Figure 4.4: Target Located - Swarm Indicating Direction

However, a malicious swarm is monitoring the original swarm communications and when it detects that a target has been located the malicious swarm entities attempt to misguide the search party by presenting what looks like better information regarding the target's location. This information is communicated through the original swarm and the search party is guided to the incorrect target location, as shown in Figure 4.5.

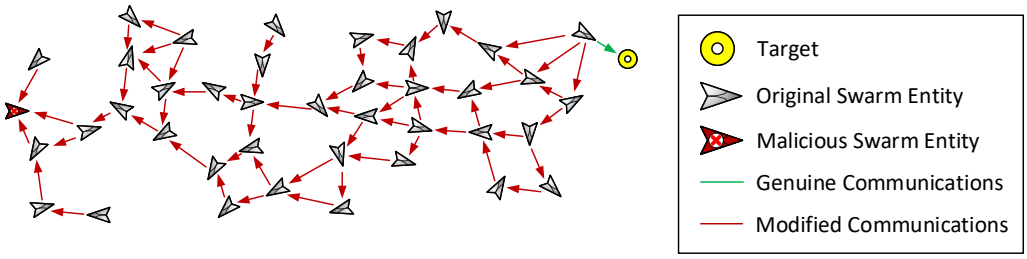


Figure 4.5: Malicious Entity Presents False Information to Original Swarm

4.2.8 Misappropriation

An attacker steals or makes unauthorised use of a service for which the service was initially unintended [167]. This is similar to authorisation validation but is not concerned with the increase in access to system resources.

Within a swarm implementation, misappropriation would be the action of making a swarm undertake a task for which it is capable of performing but was not originally intended to undertake. For example, a swarm might be tasked with undertaking a survey in attempt to locate minerals for a company. A rival company alters the swarm's behaviour so as to make the swarm undertake the same tasking but in a different location, to the benefit of the rival company. A malicious swarm, S^m, interacts with an original swarm, S^o, so as to alter the tasking of S^o.

4.3 The Adversary

An adversary wishing to attack a swarm can be viewed from several perspectives. From HMG Information Standard No 1 [32], a Threat Source is defined as a person, or organisation, that has the desire to breach security and that will ultimately benefit from the breach in some way. It also defines a Threat Actor as the person who actually performs the attack, the attacker, or in the case of accidents, will cause the accident.

The motivation of an attacker will vary, based on the objective of the swarm, and the approach and mind-set of an attacker. There is also an attacker group that will attempt to affect the swarm, just simply for the challenge to prove it can be done, this is analogous to the defacement of websites.

It has been suggested that the motivation of an attacker varies between: Very Low (Indifferent); Low (Curious); Medium (Interested); High (Committed); and Very High (Focused) [32].

An attacker's motivation will vary based on the attacker's desired outcome of an attack. That is, an attacker could have the goal of preventing a swarm from undertaking, or reaching, its goal, or an attacker might only need to delay the swarm.

Also, an attacker's motivation might change over time. To place this into context, say a swarm is located at a position that is distant from an area of interest for an attacker. The attacker therefore only commits minimal resources, in order to attack the swarm. The attackers motivation could be considered Low, as the swarm is currently not near to the attackers area of interest. However, as time progresses, the swarm moves closer to the attacker's area of interest. As the swarm becomes closer to the area of interest of the attacker, the attacker's motivation rises and the attacker commits extra resources to attacking the swarm. If the swarm manages to locate itself within the attacker's area of interest, then the attacker's motivation could be classed High or Very High and the attacker commits their entire resources into attacking the swarm, in order to protect the attacker's area of interest.

They also suggest that the capability of the attacker will vary between: Very Little; Little; Limited; Significant; and Formidable. The attacker's motivation and capability will depend upon the aim of an attacker and the resources available to them. Resources not only includes technical resources but also other resources, such as the time that an attacker is willing to expend in researching and undertaking an attack.

An attacker's capabilities and resources might well dictate how an attacker undertakes an attack. That is, if an attacker is well funded with significant resources attacks might be technical in nature and efficient. However, an attacker, regardless of resources, might be bounded by necessity, such as time, and use any means that are available, such as preventing the progress of a swarm within a limited time frame. The resources and assets available to an attacker can be varied, such as whether the attack is from within or is external to the swarm being attacked, and if the attacking force is fixed or mobile and able to move with the swarm that is being attacked. An attacker's resources could be either a single resource or multiple in quantity. If there are multiple resources, an attacker might be able to co-ordinate these resources in order to carry out an attack in either a more efficient manner, or a manner in which the attack changes as it progresses.

The actions of an attacker could also vary, depending on whether the attacker has an ability to influence the supply chain for a swarm or has control of the environment in which the swarm is operating.

Similarities can be considered between a formidable attacker, with significant resources, to that of the Dolev Yao model, within their paper regarding the security of public key protocols [54]. In their paper, an attacker is located within a network, and is essentially able to eavesdrop on messages, is able to alter, synthesise, delay and reply to any message, initiate protocols and masquerade as any party at any time [156].

4.3.1 Manner of Attack

This research proposes that the attack methods described could be conducted against a swarm in methods that can be either stealthy or non-stealthy in their undertaking, where stealthy and non-stealthy attacks are defined as:

- **Stealthy Attack**
  A stealthy attack is defined as an attack that is undertaken against a swarm that affects the desired behaviour of the swarm and is conducted in a way so as to be difficult to detect by either the swarm itself, or by an external agent monitoring the swarm, such as the authority that initially released the swarm.
- **Non-Stealthy Attack**
  A non-stealthy attack is defined as an attack that can be easily observed or detected by either the swarm or an external agent to the swarm.

As was discussed when describing the motivation of an adversary, the adversary's motivation can change over time. Therefore, an attacker might initially attempt to undertake an attack in a stealthy manner, in an attempt to prevent the detection of the attack.

However, as time progresses, the adversary might alter their attack to a non-stealthy attack, if the adversary deemed this appropriate. This could because the non-stealthy attacks had not been successful and a more direct approach of attack was required.

4.3.2 Attacker's Knowledge of a Swarm

An attacker's method of attack could be based on the amount of knowledge that the attacker has on the swarm that they are attacking. An attacker will either have knowledge or have no knowledge of the swarm's implementation. It is suggested that the main difference between the two attacks would be the resulting efficiency, or success, of an attack. Typical differences between the two degrees of knowledge of an attacker:

- **Detailed Knowledge Of Swarm Implementation**
  An attacker has detailed knowledge of a swarm's implementation, which will allow the attacker to undertake tailored attacks against a swarm. The knowledge could be gained by several methods. This could be through capturing a swarm entity, undertaking an analysis of the swarm entity, such as by using reverse engineering techniques. The attacker can then generating their own malicious swarm, or countermeasures, to interact with the original swarm. Similarly, an attacker could capturing a number of swarm entities and modifying their implementation, in order to conduct an attack upon the original swarm. An attacker might also have the ability to influence the supply chain during design, development, manufacture, distribution or maintenance activities of the swarm, which they could utilise in order to gain knowledge of the swarm's implementation.

- **No Knowledge Of Swarm Implementation**
  An attacker has no prior knowledge of a swarm, that they wish to attack. Certain attacks that can be undertaken with no knowledge might be very efficient and effective. However, the attacks might not be very subtle in there undertaking, such as the physically destruction of the swarm entities.

There could be various degrees of knowledge for an attacker, between a detailed knowledge through to no knowledge of a swarm's implementation. A limited knowledge could be obtained by such means as simply observing a swarm's behaviour within its operating environment and observing how a swarm reacts to various interactions and stimulations.

4.4 Attack Modelling

In order for an attacker to have an effect on the emergent behaviour of a swarm, the attacker conducts the attack against the entities within the swarm. That is, the behaviours of the swarm entities are altered by an attack, which has the subsequent effect of altering the overall emergent behaviour of the swarm, as a whole.

An attack has been defined as "an attempt to gain unauthorised access to system services, resources, information or an attempt to compromise system integrity, with this definition being extended to the intentional act of attempting to bypass one or more security services or controls of an information system" [139]. It has also been defined as "any kind of malicious activity that attempts to collect, disrupt, deny, degrade, destroy information system resources or the information itself" [41].

An attack has similarly been defined within ISO 27000 as "an attempt to destroy, expose, alter, disable, steal, gain unauthorised access or make unauthorised use of an asset". An attack is the realisation of a threat [64].

An attack on a swarm is essentially an action with the potential to prevent a swarm from achieving its goal, or to reduce the overall efficiency of the swarm from achieving its goal. Within this research, an attack upon a swarm can include any, or all, of the elements of the attack definitions. The attack has the objective of altering the emergent behaviour, efficiency or ability of the swarm to achieve, or realise, its intended goal.

A swarm is an autonomous body which, in many cases, is often proposed to operate beyond the control and visibility of the authority that deployed it. It is also not inconceivable that swarms may operate in hostile environments, where an attacker may detect the swarm and attempt to manipulate its behaviour.

4.4.1 Attacking Swarm Entities

An attacker can attempt to attack the entities of a swarm using several possible methods, which are detailed in Section 4.5. These attacks can effect the swarm entities, s_i^o, the interaction between the swarm entities and malicious swarm entities, s_i^m, or the local environment in which the swarm entity is operating within, e_i.

An attack on a swarm entity, s_i^o, will attempt to influence or manipulate its attributes, RA_i^o, in order to make s_i^o react in a way so as to alter the behaviour of s_i^o. The aim of the attacker is to manipulate the behaviour of s_i^o, or as many original swarm entities as is required, such that it alters the emergent behaviour of the original swarm S^o, so as to adversely effect the goal of S^o.

4.4.2 Attacking the Swarm

If the attribute sa_0^o of swarm S^o was the goal of S^o, the objective of an attacker undertaking an attack against RA_i^o of s_i^o, is ultimately to effect sa_0^o in a way that is of benefit to the attacker. Such as ensuring that S^o never achieves sa_0^o, or ensuring that S^o achieves sa_0^o in an inefficient manner.

It is proposed that a swarm can be attacked by four different methods, regardless of a swarm's implementation:

- **Manipulate a swarm entity's goal**
  If ra_0^o is the goal of a swarm entity, manipulating the goal can effect the goal of the swarm, sa_0^o. For example, the goal of a swarm, sa_0^o, could be to locate survivors within a disaster area and the goal of a swarm entity, ra_0^o, is either to search for survivors or return to the release point. An attack on a swarm entity could be to attempt to force the entity to change its state. That is, if the attacker could force a swarm entity to change the state of ra_0^o from "searching for survivors" to "return to release point", the efficiency of sa_0^o, the goal to locate survivors would be reduced.
  To help place this into context, imagine ra_0^o of a bee is either collecting food or defending the bee hive. The current state of a bee's ra_o^o is to collect food. A malicious attacker fools the bee into believing that the bee hive is under attack, forcing the bee to return to the bee hive in order to defend the bee hive. This action prevents the bee from efficiently collecting food for the bee hive, compared to being able to carrying out its normal food collecting, without believing that the bee hive is under attack. The efficiency of a swarm goal of collecting food for the bee hive is therefore reduced.

- **Manipulate a swarm entity's behaviour**
  A swarm entity, s_i^o, has several attributes, RA_i^o. If an attacker is aware of the effects various attributes have to the behaviour of s_i^o, then an attacker can attempt to manipulate these attributes accordingly. For example, if an attacker knew that ra_3^o of s_i^o required s_i^o to maintain a minimum separation from its neighbours, an attacker could release a malicious swarm, S^m, with swarm entities s_i^m, where s_i^m would attempt to position themselves closer to s_i^o than is permitted by ra_3^o. This would have the effect of forcing s_i^o to move away from s_i^m, which could have the effect of altering the emergent behaviour of S^o.

- **Manipulate the physical structure of the environment**
  If a swarm, S^o, is operating within an environment, E, an attacker could alter elements of E, e_i, such that it is of a hindrance to the swarm entities, s_i^o, of S^o.
  The most obvious techniques would be physical barriers to block the progress of s_i^o, or guide the swarm entities in a way that is of benefit to an attacker. Typical examples would be the construction of high walls and positioning of obstacles or moats and ditches around areas that are required to be defended from a ground based swarms.

- **Manipulate the physical communications provided via the environment**
  If an attacker has knowledge of the attributes, RA_i^o, of an entity, s_i^o, from a swarm, S^o, that interact with the local environment, e_i, then an attacker can manipulate e_i to the attacker's advantage.
  Typical examples of how this attack could be achieved could be: if S^o utilised stigmergy to communicate between its swarm entities, then chemicals could be added or removed from the environment; or if S^o communicated via RF, the RF operating frequencies could be jammed.

All four attacks methods are able to be conducted either stealthily or non-stealthily and with or without any knowledge of a swarm's implementation.

4.5 Attack Methods

The following section provides details regarding potential methods of attacks upon the entities of a swarm, s_i^o, and provides examples of how these attacks could be conducted to effect a swarm. The taxonomy of attack methods is an amalgamation of typical attack methods from literature, which could be undertaken against a swarm or its entities. It should be noted that the various attacks can carried out individually, several attacks undertaken together or attacks can be conducted so that the attacks interact with each other, in order for an attack to be more efficient, or for an attacker to have greater success in their attack upon a swarm.

As suggested in Section 4.1, potential attacks on a swarm will vary, according to a swarm's implementation and the swarm's operating environment.

Other factors that could determine an attack method against a swarm might be the attackers' motivation, capability, intent and knowledge of the swarms' implementation.

The following taxonomy of attacks is an amalgamation of typical attack types from the literature that could be relevant to a swarm.

4.5.1 Physical Tampering

If a device that interacts with sensitive information, as defined by the information owner, is used in an hostile environment, or an environment where it is prone to capture, then the device should attempt to ensure a certain level of physical security for the storage, forwarding and processing of the sensitive information. The term sensitive information will differ, depending upon the purpose, design and implementation of a device. Typical examples of sensitive information are cryptographic keys, messages in transit, plain text messaging, firmware and software implementations.

It is informally suggested that secure hardware assumptions encompasses two different components [73, 116]: Read-Proof hardware and Tamper-Proof hardware. Where read-proof hardware prevents an attacker from reading anything about the data stored within it and tamper-proof hardware prevents an attacker from changing anything in the data stored within it. This has been further extended to include the use of the term of Tamper-Resistant Hardware, "as a relaxed term of Tamper-Proof Hardware" [116]. That is, the hardware is resistant to physical tampering to a certain extent. The actual extent of Tamper-Resistance required would be dependent upon the requirements, design and implementation of the system.

In reality, if an attacker has sufficient resources, such as time, equipment and expertise, a device will never be truly tamper-proof or read-proof. A device can only be made tamper-resistant, with the level of resistance being offered varying on implementation.

It has been suggested that active implementation attacks can be classified as fault analysis, physical manipulations and modifications [116]. They propose that fault analysis aims to cause an interference with the physical implementation, in order to cause an erroneous behaviour that can result in a vulnerability, or failure, of a security service. The terms manipulation and modification stem from definitions of physical security, such as from ISO-13491-1 [89], and address similar attacks. They suggest that physical manipulation aims at changing the processing of the physical implementation, so that it deviates from the specification, and physical modification is an active invasive attack, targeting the internal construction of the device.

In order to gain an appreciation of the diverse physical attacks that a secure device could be subject to, ISO-13491-2 [90] provides the following typical examples: chemical attacks, such as by the use of solvents; scanning attacks, by using equipment such as scanning electron microscopes; mechanical attacks, for instance by drilling, cutting or probing the device; thermal attacks, by subjecting the device to extremes of high and low temperatures; radiation attacks, such as x-raying the devices; exploiting information leakage through covert (side) channels, such as through the monitoring of power supplies and timing characteristics; and failure attacks, where a device is forced into a failure mode, or condition, where the failure characteristics can be used by an attacker to the attacker's advantage.

It is assumed that a swarm can operate within a hostile environment and that the entities within a swarm are operating within an environment, which makes a swarm entity prone to capture. This is different to the traditional sense of computing, where the physical protection for the computing assets can generally be achieved by physically preventing access to the computers, or by physically protecting the information stored on a computer's hardware, in case the computer is lost or stolen.

Therefore, it could be possible for an attacker to extract information that will be to their advantage, such as gaining an understanding of a swarm entity's software, firmware and possible cryptographic details, such as cryptographic primitives and key variables.

A physically tamper attack on a device can lead to the realisation of several threats, depending upon the goal of the attacker, such as: masquerade, planting, communications monitoring and data modification.

Also, the characteristics of a swarm is that it is designed to be resilient to failures within the swarm, swarm characteristic 7, and the swarm can alter its size, swarm characteristic 8, so as to be resilient to the loss of swarm entities, such as through the failure of swarm entities. Therefore, if an attacker were to remove a swarm entity from a swarm, the swarm would operate as normal, allowing an attacker to investigate the captured swarm entities without constraint. An attacker would therefore be able to characterise the swarm entity, including by physically tampering with the swarm entity, in order to gain information.

From Equation 3.1, within the generic model, the original swarm S^o is represented as:

$$S^o = \{s_0^o, s_1^o, \ldots, s_n^o\} \tag{4.1}$$

Where n is the number of entities within the original swarm.

The attributes of the original swarm are represented as:

$$SA^o = \{sa_0^o, sa_1^o, \ldots, sa_z^o\} \tag{4.2}$$

Therefore, if a attacker removes v swarm entities from the original swarm and assuming no failed swarm entities, the original swarm becomes:

$$S^o = \{s_0^o, s_1^o, \ldots, s_{n-v}^o\} \tag{4.3}$$

The attacker could undertake several attacks against the removed swarm entities. If the attacker undertook an attack that modified the removed swarm entities and then returned them to the original swarm, the malicious swarm S^m would be:

$$S^m = \{s_0^m, s_1^m, \ldots, s_v^m\} \tag{4.4}$$

With the attributes of S^m being:

$$SA^m = \{sa_0^m, sa_1^m, \ldots, sa_w^m\} \tag{4.5}$$

The attributes of a swarm entity, s_i^o, are:

$$RA^o = \{ra_0^o, ra_1^o, \ldots, ra_c^o\} \tag{4.6}$$

and a malicious swarm entity s_i^m are:

$$RA^m = \{ra_0^m, ra_1^m, \ldots, ra_d^m\} \tag{4.7}$$

However, as the attacker removed a swarm entity from the original swarm, modified the swarm entity and then return the modified swarm entity to the original swarm, the original swarm entities and modified swarm entities could share certain attributes, these would be characterised by $RA^o \cap RA^m$.

An attacker could remove swarm entities from a swarm, in order to tamper with the swarm entities in order to gain an understanding of how the swarm entities operated, such as how they interacted with other swarm entities or the operating environment. The attacker could then use this knowledge to produce their own malicious swarm, from the details gained. Then the number of malicious swarm entities that the attacker could release, would not be bounded by the number of swarm entities that had been removed from the original swarm. That is, an attacker is not limited by the amount of swarm entities removed and could introduce as many malicious swarm entities as they desired.

That is:

$$S^m = \{s_0^m, s_1^m, \ldots, s_u^m\} \tag{4.8}$$

Where u is the amount of malicious swarm entities built and then released into the original swarm, by an attacker.

The attributes of the malicious swarm and the malicious swarm's entities could again be totally different, or they might share certain attributes, such as location and velocity.

The rationale behind an attacker wishing to tamper with a swarm entity, s_i^o, would be to gain an understanding of the operation of the swarm entity, in order to gain an understanding into the overall operation of the swarm. The attacker might also wish to tamper with swarm entities, in order to modify them and then reintroduce them into the original swarm, within the objective of modifying the original swarm's behaviour.

If an attacker attempted to tamper with original swarm entities s_i^o, in order to produce malicious swarm entities, s_i^m, an attack might be to alter the attributes of s_i^o. An example of this could be to reduce the minimum separation distances between swarm entities, such that s_i^m attempts to force the movement of s_i^o, in a way that was not previously considered within the original operation of S^o.

It is worth noting that an attacker does not necessarily have to remove swarm entities from S^o, but could also collect failed swarm entities from the operating environment, as these entities might contain information relevant to an attacker. Also, from an attackers' perspective, the failed swarm entities might be easier, or safer, to obtain from the operational environment.

4.5.2 Software Attacks

Software-based attacks are concerned with modifying the software code, and exploiting known software vulnerabilities [155]. A well known example of this type of attack is the buffer overflow attack. This is where the overall goal of a buffer overflow attack is to subvert the function of a privileged program, in order for an attacker to take control of that program and, if the program is sufficiently privileged, and then take control the host [44].

Software errors or a bug in the software running on wireless sensor nodes, or on the wireless sensor network base stations, can give rise to attacks which can be easily automated and these attacks can be mounted on a very large number of nodes in a very short amount of time [17].

If an attacker can gain physical access to a node, by a physical tamper attack, it is suggested that attacks can be conducted by attacking system elements, such as JTAG, bootstrap loaders and external flash memory [17].

There are suggestions that Wireless Sensor Networks may be susceptible to malicious code attacks, such as viruses, using the network to propagate a malicious code attack [186]. If a software error is found within the software used in the nodes of a Wireless Sensor Network, and assuming that all the nodes are running the same software, finding an appropriate bug would allow an adversary to control the whole Wireless Sensor Network.

When considering software attacks, such as viruses and trojans, being undertaken against swarms, the attack could be achieved in several ways. An example could be the realisations of the threat of planting, as an enabling threat within the supply chain. The swarm might then be open to the realisation of a primary threat, such as Denial of Service, System Penetration or Communications Monitoring. Communications Monitoring could also be realised by an attacker "re-visiting" the swarm, once an attack had been undertaken.

If a swarm entity, s_i^o, is captured and subject to physical tampering, a local attack could be conducted against s_i^o, in order to exploit either known software vulnerabilities, or exploit software vulnerabilities that are discovered during the tampering attack. The software vulnerability findings could then subsequently be utilised against other members of the swarm S^o.

Software vulnerabilities could also lead to behaviours that were not expected by the original swarm designer. That is, if certain conditions are met, the attributes for s_i^o interact in such a way, so as to make s_i^o react and behave in an unexpected manner. This knowledge could be gained through physical tampering, enabling an understanding of the attributes of s_i^o, their thresholds and interactions, or by observation of entities within S^o. An attacker could then utilise this information in an attempt to maliciously manipulate S^o.

4.5.3 Attacks on Communications

Attacks on communications can take several forms and the method of attack will be particular to the implementation of a swarm. The method of attack will also be determined by the objective and the knowledge of an attacker.

An attack on the communications of a swarm could be undertaken within various locations, which are involved with the communications process within the swarm and its swarm entities. This section will provide examples of typical attacks to the communications within a swarm.

Communications within a swarm can take several forms and can be distinct from other communications architectures. Depending upon a swarm's implementation, the communications can be either direct or indirect. Direct communications between swarm entities are real time one-to-one or one-to-many communications, such as by RF or optical implementations. Indirect communications are achieved by using stigmergy and changes to the operating environment, such as the use of pheromones or the placing beacons or markers.

4.5.3.1 Replay

A replay attack is attempting to exploit a vulnerability within a system, that realises the threat from eavesdropping, in order to realise the threat of masquerading. That is, by eavesdropping, an attacker captures legitimate traffic between two entities within a system and then replays the traffic to the intended victim at another time.

The attacker does not necessarily know the content of the messages, within the captured traffic, but relies on the fact that the attacker believes that these were once legitimate messages, between the communicating entities. The attacker is therefore masquerading as a legitimatise user within the system.

This masquerade can then lead to subsequent threats being realised, such as a flooding attack being conducted, where the victim believes the messages to be genuine and the effect on the system is that the threat of DoS is realised.

There are various forms of replay attack available to an adversary. These could be to fool the system into believing that it is operating as expected, or to cause a system to respond in a known way to a given command. The replay attack is generally taken to mean an attack in which a past message is played back to the same recipient [78]. However, they [78] suggest that this is somewhat misleading, as the most general definition of replay is any reuse, possibly after manipulation, of past messages.

An example of where a system was made to believe that it was operating in an expected manner, when actually it was not, was the Stuxnet attack. The Stuxnet attack monitored control systems in order to gain knowledge of the normal operating characteristics of an industrial plant. It then replayed these characteristics to the control system, as it then carried out an attack. The control system was effectively deceived into reporting that the system was operating as expected, while the industrial plant actually began operating beyond its design limits and destroying itself [115].

In order to make a system believe it is operating in the expected manner, the attacker records sensor outputs from the system, when it is operating normally, and then replays the captured information whilst attacking the system [127]. This is often portrayed as an attacker capturing a video feed from a security camera and then replaying it to the system, as the attacker perpetrates an attack. Although the perpetrator is still observed by the camera, the system is presenting the captured video images, prior to the reply attack.

Another type of replay attack would be to observe a system's behaviour based on its received commands. The attacker does not need to be able to interpret the content of a command message, just capture the message and observe the system behaviour based on the captured message. An example of this could be to consider the control of an Unmanned Air Vehicle (UAV). If an attacker was able to capture the command and control information transmitted to the UAV and observe the subsequent actions carried out by the UAV, the attacker could attempt to reproduce the actions undertaken by the UAV.

That is, if the UAV descended by a distance, the attacker could replay the message that the attacker believed commanded the UAV to descend, in order to attempt to make the UAV continue to descend and possibly crash. The attacker would not necessarily need to know the actual format, content or security mechanisms protecting the message, such as encryption schemes. The attacker just needs to have the ability to capture and replay messages.

Within the context of swarms, replay of communications can take several forms. These being the replay of communications between swarm entities, or the reproduction of environmental effects that are perceived by the swarm entities. An example of reproducing environmental effects, in order to realise a reply attack, could be undertaken against a swarm that communicated using stigmergy. The attack could be achieved by an attacker collecting pheromones from a swarm's operating environment and then, subsequently, re-introducing the pheromones to the environment, at a later time.

Within the swarm, the swarm elements, s_i^o, will have an attributes that could be attacked. Attributes, RA^o, could be:

ra_r^o = received message, which s_i^o receives, reads, interprets and then act appropriately.

An attacker, through eavesdropping, has captured traffic that makes s_i^o behave in a manner, that has been observed by the attacker. The behaviour is advantageous to the attacker and therefore the attacker replays the message at an appropriate time.

Similarly:

ra_s^o = action to undertake based on a detected pheromone level.

An attacker, through the capture of a pheromone and observation of the effect of the pheromone on a swarm element, s_i^o, releases the pheromone in order to alter the behaviour of the swarm elements and therefore the behaviour of the swarm, S^o. The attribute ra_s^o might be dependent upon thresholds, such that the action that s_i^o will undertake, based on the concentration level of the sensed pheromone. This knowledge could be of benefit to an attacker, so as to conserve the attacker's supplies of pheromone, for subsequent attacks on elements of S^o.

4.5.3.2 Misinformation on Communications

Misinformation is essentially the realisation of the threat of data modification, which would be undertaken by the threat of masquerade. That is, an attacker masquerades as a legitimate member of the swarm and provides incorrect information, either modified information that had been received from within the swarm, or false data that was sourced by the attacker.

For an attacker to be able to receive, store and then forward misinformation within the swarm, the attacker has realised the masquerade threat, as an enabling threat and allowing the attacker to be trusted. The attacker then achieves the threat of communications monitoring, to eavesdrop on the communications, as another enabling threat.

This then enables the primary threat of data modification to be achieved, in order to prosecute a data misinformation attack. A more sophisticated variant of this attack would be to only provide selective misinformation, based on the information observed whilst undertaking the communications monitoring. An attacker would only alter specific information in order to achieve their goal.

A swarm entity, s_i^o, perceives an attacker, s_i^m, as a member of its own swarm S^o. In an attack, s_i^m monitors the communications and is looking for a trigger, in order to alter a specific message. All other messages are forwarded as received, thus enabling the attacker to maintain its trust within S^o. Details of the messages to be modified would be modelled as an attribute, ra_i^m, of s_i^m. This could be considered a man-in-the-middle attack [35, 111, 185].

For an attacker to undertake misinformation on indirect communications, the attacker can either modify a previously placed messages, such as by removing pheromone trails or beacons, or an attacker can add false messages, such as by adding extra pheromone trails or beacons.

In the indirect communications attack, the attributes,$\{ea_0, ea_1, \ldots, ea_z\}$, of the local environment, e_i, that are monitored by swarm entities, s_i^o, have been modified. This could be carried out by either a malicious swarm, S^m, or just by physical intervention by other means from an attacker, such as by washing away pheromone trails.

4.5.3.3 Interfere or Prevent Communications

In this attack, the communications between swarm entities are interfered with, such that the communications between the swarm entities is degraded. This is a realisation of the Denial of Service (DoS) threat to the swarm.

If a swarm was communicating by RF on a single fixed frequency, one of the most basic attacks would be to deny the communications within an area by jamming the communications on the single operating frequency. Although not subtle, it can very effective form of attack.

If the swarm was communicating by RF but utilised multiple frequencies, by utilising techniques such as frequency hoping or spread spectrum, then the jamming becomes a little more involved. The most basic, least subtle and potentially inefficient attack is to barrage jam the entire RF operating spectrum of the swarm. A slightly more subtle, and more efficient from the perspective of jamming energy required, but needing more processing capability would be to jam specific frequencies in use by the swarm. This would need to including follower jammers, if the swarm utilises frequency agile techniques to alter the frequency it is using. A swarm might utilise frequency agile techniques to aid with spectrum deconfliction, or to try and lower its probability of intercept or jamming. However, this might take up more processing power within a swarm entity and also use more power, from what is already a limited resource.

The attacker can make the jamming attack more subtle, if the attacker is able to realise the threat to the swarm of communications monitoring. If the malicious swarm entities, s_i^m, are able to eavesdrop on the communications, the attack can be tailored to only jam specific messages, making the attack more efficient. The eavesdropping threat would exist, regardless of whether S^m had achieved the threat masquerade or not, and would act as an enabler to prosecute the primary threat of DoS.

This is in order to prevent communications or enable interference to degrade communications. For example, if the attacker knows that the first part of a message, within a transmission, is always a synchronisation header for the message and that is required by the receiver to enable reception of the message, then the attacker might only jam the synchronisation portion of the message. Similarly, an attacker can jam control channels used for timing information. Techniques have been proposed that can jam Long Term Evolution (LTE) wireless networks that utilise "smart jammers" [11, 98]. The advantage for the attacker is that the attacker uses less RF energy and only transmits for limited amounts of time, therefore making itself harder to locate for counter measures. The disadvantage is that it takes more processing power to achieve.

To enable the jamming of swarm communications, the jammer could be either part of a malicious swarm that moves with the original swarm, or it can be a fixed capability that is external to the swarm, such as when attempting to defend a fixed area. The advantage of a fixed jammer is that it can often have access to more resources, such as power, and it might possibly have larger jamming ranges. However, with being fixed it is only capable of jamming a specific area and can also make itself an easier target to counter measures.

Within the swarm, S^o, malicious swarm entities, s_i^m, could act as a mobile jammers, which move with S^o. The malicious entities that form the malicious swarm S^m can be either believed to be legitimate members by the swarm S^o, in which case S^m has also realised the masquerade threat, or it can be totally separate to S^o, where S^o might be unaware of the existence of S^m. The advantage to the attacker of S^m moving with S^o is that the attacker could tailor an attack, such as at particular locations or time, and could disperse the malicious swarm entities, s_i^m, to produce the maximum effect. This could be at communications choke points, or provide the ability to jam the communications, such that the jamming effect extends beyond the reception range of swarm entities s_i^o. Therefore, even if the message has started to be forwarded within a swarm, via multiple routes, there is still the ability of preventing its onward transmission.

The swarm could also be subject to a DoS threat, by the prevention or interference of communications, if S^m were able to masquerade within S^o. The masquerade would enable the threat of eavesdropping. The malicious swarm entities, s_i^m, within S^m would then monitor the communications that it is expected to forward on. If the communications are of no interest to s_i^m, then s_i^m forwards the communications on, as if it were a legitimate swarm entity.

However, if s_i^m detects a message of interest, then s_i^m does not forward the message on. If a malicious swarm entity, s_i^m, drop all messages that it received, this entity would effectively act as a "sink-hole" attack within the swarm, otherwise it could be categorised as a selective forwarding attack.

The malicious swarm entities, s_i^m, can then undertake a jamming attack, in an attempt to prevent messages from being communicated via other routes. In Figure 4.2 the other routes prevented the malicious entity from successfully attacking the communications.

However, if the malicious entity was also able to jam the communications, then the attack could be improved, as shown in Figure 4.6.

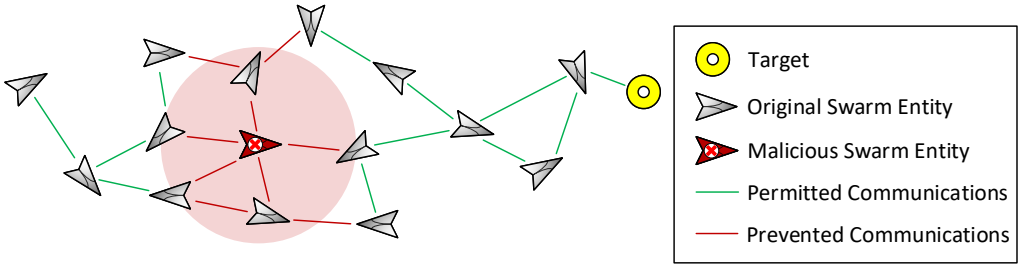


Figure 4.6: Single Jammer Attack on Communications

The effect of the jamming would be dependent upon the size of the jamming effect and the size of $S^o$and S^m. That is, if there were only one jammer, the communications would probably be routed via a different path. However, an attacker could deploy multiple malicious entities, as shown in Figure 4.7.

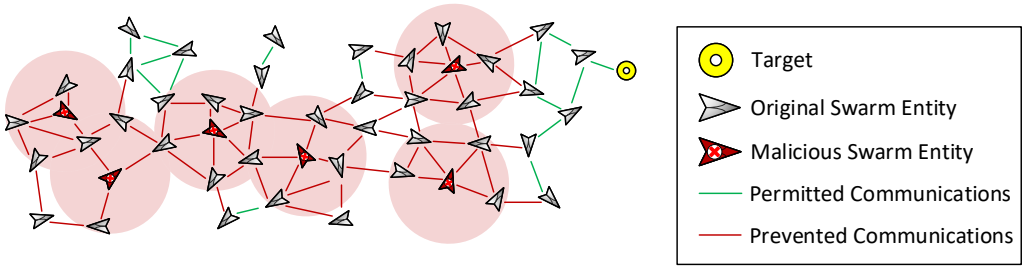


Figure 4.7: Multiple Jammers Attacking Communications

4.5.3.4 Collisions

Another method of the prevention of communications is to essentially flood the communications with false messages, which the swarm entities within S^o believe to be genuine and therefore attempt to process. This has the effect of using processing time on the swarm entities, s_i^o, and also false message essentially "colliding" with genuine messages.

The collisions can possibly corrupt the genuine messages or preventing their delivery, as s_i^o is already attempting to receive and process a false message. The false messages can be transmitted from either fixed locations or by malicious swarm entities, s_i^m, in a similar fashion to the undertaking of a jamming attack.

4.5.3.5 Exhaustion

Within an exhaustion attack, the attacker is attempting to exhaust the resources on a victim's system, such as CPU clock cycles, network applications and protocols, memory and hard drive space.

An example of an exhaustion attack is the SYN flood attack, which attacks an exploit in the Transmission Control Protocol's (TCP) three way handshake that is used to establish a connection [72, 197]. A connection process consists of the client sending the server a SYN (Synchronise) request, which is responded to by a SYN-ACK (Synchronise-Acknowledgement) message. The server then waits for the client to respond with a corresponding ACK (Acknowledgement) and a connection is established. This can be seen in Figure 4.8.

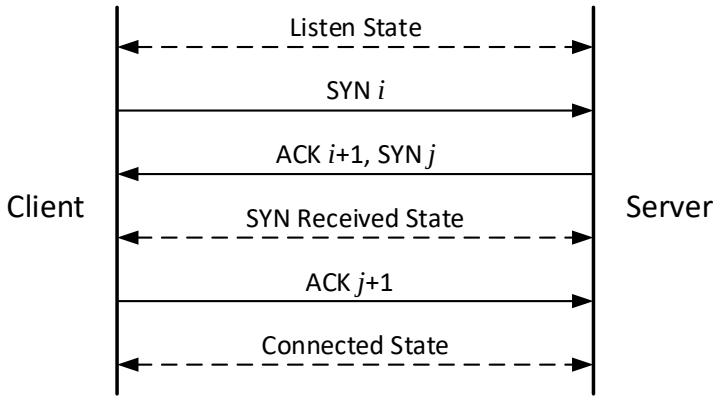


Figure 4.8: TCP Three Way Handshake [72]

In an attack, an attacker sends multiple SYN requests to the server, often from fake IP addresses. The server responds with a SYN-ACK message but does not receive the corresponding ACK. While waiting for the ACK message, the server cannot close down the connection until a time period has elapsed. Prior to the time period elapsing, more attack SYN requests are received and the server attempts to fulfil the requests. This has the effect of filling the server's connection table with half-open requests and eventually exhausts the servers resources. This can be seen in Figure 4.9.

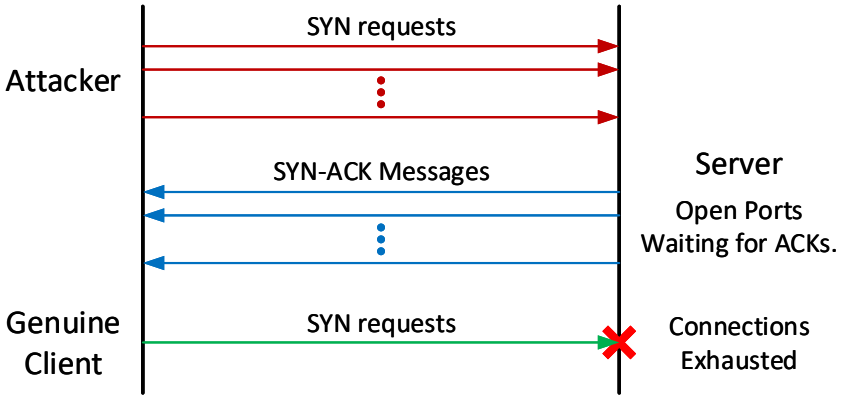


Figure 4.9: SYN Attack

In the context of a swarm, the attacker, S^m, causes the swarm entities, s_i^o, to retransmit information, or to make more transmissions than s_i^o would normally undertake. This has the effect of causing more transmissions from s_i^o than was expected, from the swarm design implementation, and causes the power supplies of s_i^o to become exhausted earlier than expected.

4.5.4 Incorrect Information Display

There are proposed attacks where an attacker could maliciously alter a display, which is presented to a recipient of the information. One area where this is of concern is within medical implants. The concern is that an attacker could attack a device, so that it displays an incorrect reading or measurement. For example, an implantable dynamic tattoo is programmed to display blood glucose levels. If the device is compromised, it might display inaccurate data or unauthorised text [82]. This could consequently result in an inappropriate insulin injection being administered to a patient.

Within a swarm, a similar attack might be to attempt to make swarm entities display incorrect data, such as landmine found, when a landmine is not present, or not display when a landmine actually is located. Therefore, entities within S^o could be coerced into displaying incorrect information. That is, an attacker, S^m, has realised the threat of system penetration, in order to modify the information with the swarm's systems. This could be achieved by s_i^m displaying incorrect information, or the swarm entities within S^m causing the swarm entities s_i^o to display the incorrect information, such as by realising the threat of the modification of data in transit, to enable the primary threat of system penetration.

4.5.5 Reconnaissance

In order to assist in carrying out an attack, an adversary may undertake a reconnaissance of a swarm, in order to gain intelligence. The method of reconnaissance used will vary, depending on the capabilities of an attacker.

4.5.5.1 Traffic Analysis

An attacker can gain intelligence by utilising traffic analysis, sometimes referred to as traffic flow analysis, which is undertaken by the monitoring of transmissions for patterns of communication. There can be a considerable amount of information contained in the flow of messages between communicating parties [167]. An attacker passively monitors transmissions, in order to identify communication patterns within a network. An attacker can typically obtain information by monitoring which parties communicate and at what times, in order to determining the content or content type of communications messages.

It is suggested that traffic analysis attacks try to deduce the context information of nodes, by analysing the traffic pattern from eavesdropping on communications [118, 155].

Traffic analysis is essentially an attacker realising the threat of communications monitoring, by eavesdropping on the victim's communications. That is, the communications with the swarm, S^o.

The traffic analysis does not necessarily need to observe the content of the messages and is not just confined to the content of data collected. Traffic analysis also utilises other messaging, such as command, control and management messages.

Traffic analysis is the study of communications patterns, not the content of the message but their characteristics [170]. It has often been reported that the pizza deliveries to the Pentagon increased by one hundredfold, in the hours preceding the U.S. bombing Iraq in 1991 [170]. An observer would not be aware of what type of pizzas were being ordered, or what was happening within the Pentagon. However, an observer would suspect that something unusual was happening, as the amount of pizzas being delivered was uncharacteristic, when compared to the normal activity.

4.5.5.2 Visual Observation

Due to the fact that a swarm operates within an environment, it might be possible for an attacker to observe the swarm and its capabilities. An attacker will be able to observe the swarm and how it reacts to various stimulus and situations, such as how a swarm reacts with various environmental aspects and the interactions between swarm entities, s_i^o. This will provide an insight into how the swarm entities within S^o react, which will aid in the understanding of likely responses to various stimulus, such as by S^m. The observation will be passive but could give an attacker an understanding of how successful, or not, an attack has been, which the attacker could use to refine their future attacks.

4.6 Discussion

This chapter has provided an overview of the various types of threats, including enabling and primary threats, to a swarm. It is also proposed that there are no new threats that are specific to swarms. However, there are new and unique attacks that could be targeted against swarm implementations. Potential attacks that could be conducted against a swarm and its entities have been discussed. Many of the threats and attacks have been mitigated in "non-swarming" systems, using various techniques. However, these mitigations are not necessarily appropriate for certain aspects, that are unique to swarms.

Attacks can be mounted against a swarm that are similar in nature to attacks against other systems, such as eavesdropping within a Wireless Sensor Network. However, it is proposed that the actions and subsequent effects on a swarm will be different. For example, within a Hierarchical Wireless Sensor Network, an attacker that realises the eavesdropping effect might position themselves appropriately within the hierarchy of the network, in order to realise other attacks, such as attacking the network routing, using attacks such as a Selective Forwarding attacks [27, 207], where particular messages are not forwarded and others are forwarded as usual; Sybil attacks [57, 140], where nodes present multiple identities; or Wormhole attacks [87, 94, 198], where low latency links tunnel information to different parts of a network.

However, as a swarm is not a hierarchical network architecture, the attacks that could be successful against a Wireless Sensor Network, would not be effective against a swarm. Similarly, both Hierarchical Wireless Sensor Networks and Distributed Wireless Sensor Networks use command and control mechanisms and report to a data sink point. However, as a swarm has a decentralised control mechanism, as detailed in swarm characteristic 2, and a resulting collective emergent behaviour, swarm characteristic 4, then the potential attacks to Wireless Sensor Networks and Distributed Wireless Sensor Networks would not be suitable to attack a swarm implementation.

It can be seen that, although some systems appear to be similar to swarms, there are important difference in the detail of implementations and operations. These differences arise from the unique characteristics of a swarm, as described in Section 1.2 and Section 2.4. Consequently, many attacks on "swarm-like" systems do not actually apply to swarms and therefore, by a similar rationale, not all mitigation techniques that apply to "swarm-like" systems apply to swarms.

In particular, the unique characteristics of swarms, such as the use of stigmergy as a communications medium and the emergent behaviour of a swarm, present a new challenge to attackers and researchers.

The details in this chapter, in conjunction with Chapter 3, have provided an understanding of the various threats and attack mechanisms which could be carried out against swarms and how these relate to the swarm model. This information as then utilised in the simulation of attacks against robotic swarms, which is presented in Chapter 5.

4.6.1 Case 1 - Cooperative Navigation (Hunting)

In use-case 1, the objective of a swarm, S^o, is to hunt for and locate a target, T. The swarm might be further constrained, depending upon the implementation or actual operation of the swarm. That is, the swarm might also have other constraints, such as the maximum amount of time allowed to locate the target, say $sa^o_{maxTime}$. The swarm will therefore achieve its goal if it successfully locates the target and it does so within any constraints.

The role of an attacker is to attempt either prevent S^o from locating T, or to attempt to reduce the efficiency of S^o, such that the time taken to locate a target exceeds $maxTime$.

An attacker would release a malicious swarm, S^m, in order to undertake an attack against S^o. The actual attack against S^o would be conducted by the swarm entities of S^m against the swarm entities of S^o. Therefore, s^m_i will attempt to influence s^o_i.

As an example within use-case 1, attacks undertaken by s^m_i could be attacks on the communications between s^o_i, as described in Section 4.5.3 . For example, s^m_i might attempt to jam the communications between s^o_i, or, by realising the threat of masquerade, s^m_i could monitor the communications between the s^o_i swarm entities. s^m_i would then present modified information to s^o_i, such as when the messages between s^o_i report target located, in order to attempt to maliciously manipulate the actions of S^o.

4.6.2 Case 2 - Foraging and Local Recruitment (Landmine Locating)

In use-case 2, the objective of a swarm, S^o, is to make safe all the landmines, LM, within an operating environment, E. The goal of s^o_i is to locate a landmine, lm_i, and then locally recruit other swarm entities of S^o, such that there are enough swarm entities present at the landmine in order to make the landmine safe. Upon which, the swarm entities will then return to locating and making safe other landmines, within E.

Let us say that when swarm entities, s^o_i, locate a landmine, they undertake local recruitment by the release of a chemical into the environment and that the chemical's density reduces over distance, such that it is not detectable after a particular distance, $rangeChem$. Upon detection of the recruitment chemical, s^o_i will follow an increasing chemical density, until it reaches a chemical density maximum, upon where s^o_i would locate a landmine.

The objective of the attacker is to prevent S^o from successfully making all the landmines safe within E.

As an example for use-case 2, attacks undertaken against S^o could be manipulate the physical communications provided by the environment, as described in Section 4.5.3.

An attacker could attempt to manipulate the communications within the environment by providing false information, such as by the attacker releasing a false recruitment chemical, manipulating the actions of s_i^o so that s_i^o follows the false chemical scent within E. An attacker could defend their landmines by placing the false chemical scent generators strategically within E and, assuming unlimited resources, could deploy the false chemical scent such that its effective range was greater than that of *rangeChem*.

5 Simulating Attacks on Swarms

5.1 Introduction

This chapter describes how the previous work regarding threat taxonomies, in Chapter 4, was then taken forward to simulate attacks against swarms.

The aim of simulating robotic swarms was to gain an understanding of if, and how, malicious attacks could be conducted against these swarms [165]. The simulations were conducted against several types of swarms and utilised various attack techniques. The objective of the simulations was to observe and record the effect of the attacks upon the overall behaviours of the swarms.

In order to conduct the swarm simulations, a literature review was carried out to gain an understanding of other researchers' proposals, for swarm implementations. The review also considered the various types of swarms, as detailed in Section 2.5 regarding swarm taxonomies.

Following the literature review, two different types of swarm implementations were chosen to be simulated. These proposed swarm implementations had been developed within different areas of swarm research, with different goals for the swarms.

The previous researchers had undertaken work regarding the application of swarms for cooperative navigation [59] and the utilisation of swarms with foraging techniques combined with local recruitment schemes, with a proposed use within landmine detection [34, 112, 113, 114].

The simulations undertaken within this research for cooperative navigation are a continuation of use-case 1, where the goal of the swarm utilising cooperative navigation is to locate a target. The simulations undertaken within this research for the foraging techniques combined with local recruitment schemes are a continuation of use-case 2, with the goal of the swarm being to locate and make safe landmines.

Initial simulations within this research were conducted on swarm implementations within benign environments, which did not involve any malicious activity or attacks. These initial simulations were carried out, in order to ensure that the results obtained within this research matched the results presented by the other authors' within their research. This research work then attempted to maliciously manipulate these different swarm types, conducting various types of attacks, as detailed in Section 4.5, against the swarms.

Once initial operating baselines had been established, the effects on the swarm from simulating malicious activity could then be observed.

5.2 Case 1 - Cooperative Navigation

This section describes work undertaken on simulations that demonstrate use-case 1, where the overall objective of the swarm is to locate a target by the use of cooperative navigation.

The following work was based on research initially carried out by Ducatelle, et al., regarding their proposal to utilise swarms for the provision of cooperative navigation [58, 59]. Entities within a swarm forward navigation information to other swarm entities, that are within communications range of each other. This is in order to attempt to cooperatively provide the most efficient way of navigating to a target location. The entities within the swarm can be either swarm entities that are attempting to locate the target, essentially searchers that are hunting for targets, or they can be external to the swarm that is searching for a target. This could be independent swarm entities that are carrying out other tasking, but are able to store, process and forward navigational information. Ducatelle et al. initially conducted simulations of between 1 and 90 simulated robots for their proposal and subsequently undertook experiments utilising real robot entities, using between 1 and 10 actual robots. As Ducatelle et al. reproduced the same results for the simulations when compared to using real robots, it was decided to only consider simulations within this research. The results obtained by Ducatelle et al. demonstrated the validity of the simulations and as this research was concerned with understanding attacks on swarms, and the subsequent effects, it was decided that undertaking simulations would be a valid way of conducting the research. Simulations also provided a repeatable way of testing different approaches to various attack methods, in order to observe any effects, and allowed the researcher to monitor actions from the perspectives of both the overall swarm and the individual swarm entities.

5.2.1 Taxonomies

The cooperative navigation implementation of a swarm may be considered from several perspectives when considering taxonomies, as detailed in Section 2.5.

When considering the *Method Based* taxonomy proposed by Brambing et al. [24], this representation would be considered a *finite state machine* within *Behaviour Based* design methods. Within the *Collective Behaviour* based taxonomy, Brambing et al. would consider this as *Collective Exploration* within *Navigational Behaviours*.

The *Robotic Swarm Behaviour* taxonomy proposed by Cao et al. [31] would classify this activity as a *Communications Structure* within a *Group Architecture*. It could be considered a *Geometric Problem*, in that the swarm is assisting in path planning. However, the collective entities are only passing on navigational information to assist in path planning and not actually planning the path to take themselves.

5.2.2 Overview of the Previous Research

In the research undertaken by Ducatelle, et al., a proposal was presented for a new algorithm that enabled a collision free path to be found between either a searcher and a target, such as for a search and rescue scenario, or back and forth between two locations, such as between an insect nest and a food location [58]. They refer to these two scenarios as *single robot navigation* and *collective navigation*. The messages sent between the swarm entities leads to an emergent behaviour within the swarm, which, as there has not been instruction or influence by an external controlling source, is decentralised. They refer to this as *cooperative navigation* within the swarm.

Their proposals utilise decentralised control, swarm characteristic 2, and local communications between the swarm entities, swarm characteristic 6, in order to achieve a collective emergent behaviour, swarm characteristic 4.

The work by Ducatelle, et al. considered the use of direct communications between local swarm entities, within a confined operational area. An event is advertised, such as a target is found, and this event is then periodically advertised and then relayed throughout the swarm. This process is also carried out via the other entities of the swarm, which are not necessarily involved with the navigation task. A searcher entity, or entities, will then utilise the relayed information, in order to locate the target.

The algorithm proposed by Ducatelle, et al. is relatively simple. Each swarm entity contains a *navigation table* that stores the *information age* of the information received and a *distance to target value.* The information age is represented by a sequence number that provides a relative age of the information. The distance to target value is an estimate of the navigation distance to the target. Prior to deployment, the target's information age is set to zero and the swarm entities navigational tables are initialised, so as to receive information. The target generates an increasing count, which is used as the reference for the age of the information.

At a predetermined set period of time, the target increases the count. Any entities from the swarm, which are within communications range of the target, update their knowledge of their own distance to the target and the target's current sequence count number. This information is then rebroadcast, in order to forward the information on and distribute the information through the swarm, via swarm members that are in range of each other.

Ducatelle et al. provided Algorithm 5.1 to describe this function.

Algorithm 5.1 Communication-based navigation: the actions executed at each control step by each robot A

```
1: /* Update local distance estimates */
2: for (Each target T in navigation table) do
3:     Update distance information d(A,T) for T based on A's moved distance
4: end for
5: /* Process received messages */
6: for (Each received message from a neighbour robot B) do
7:     for (Each target T in the message) do
8:         Receive distance d'(B,T) and sequence number s'(T) from B
9:         Compute d'(A,T) := d(A,B) + d'(B,T)
10:        /* Update navigation tables if new information is better */
11:        if ((s'(T) > s(T) OR ((s'(T) == s(T)) AND (d'(A,T) < d(A,T))))) then
12:            Replace information for T in table: s(T) and d(A,T) := d'(A,T)
13:        end if
14:        /* Update navigational behaviour if new information is better */
15:        if (A is searching for target T) then
16:            if ((s'(T) > s*(T)) OR ((s'(T) == s*(T)) AND (d'(B,T) < d*(T))))
    then
17:                Replace current navigation information:
18:                s*(T) := s'(T) and d*(T) := d'(B,T)
19:                Move towards B's position
20:            end if
21:        end if
22:    end for
23: end for
24: /* Send message */
25: if (Time to send update) then
26:    if (The local robot A is a target) then
27:        Increase sequence number s(A) for target A in navigation table
28:    end if
29:    for (Each target T (subset of) table) do
30:        Add information s(T) and d(A,T) to message
31:    end for
32:    Broadcast message
33: end if
```

This can be seen in Figure 5.1. The receiving swarm entities compare the received information to the information that they already hold within their navigation tables. If the received navigational information is "better" than the information currently stored by the swarm entity, the receiving swarm entity will update its navigational table accordingly.

As the information age is generated by an increasing count at the target and subsequently communicated within the swarm, there is no need to synchronise the swarm entities across the entire swarm. The count number of the information age at the swarm entities is based on the last time they received the best navigational information.

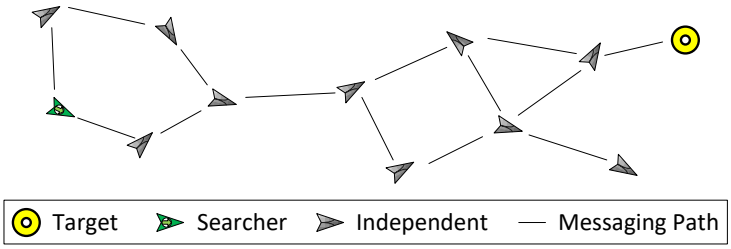


Figure 5.1: Messaging Within a Swarm

Received navigational information can be better for one of two reasons:

1. The received information is fresher than its stored information, identified by the received information age count being higher than the stored information age count.
2. If the received information age is currently the same as the stored information's age but the estimated distance to the target is lower.

If either reason is realised, the receiving swarm entity updates its navigation tables. If the received information is fresher than its stored information within its navigational table, the receiving swarm entity updates its stored information age value with the newly received value and updates the value for the estimated distance to the target. If the information age value received is the same as that already held but the estimated distance to target value is lower, just the distance to target value is modified.

The estimated distance to the target value is calculated by taking the received distance to the target value and adding on the distance between the receiving swarm entity and the swarm entity that transmitted the better navigational information. Other than a searcher swarm entity, which is searching for a target, the other swarm entities do not act on this information, other than to store-and-forward the information on to other swarm entities.

These other swarm entities were simulated such that they moved in a random manner, as if carrying out other tasks. That is, they would randomly calculate a heading to travel in and a random distance to move, that was bounded to a maximum distance, and they would then move the calculated distance. Upon travelling the calculated distance, the swarm entity would then repeat the task of randomly choosing a new direction to travel in and a random distance to travel.

It should be noted that Ducatelle et al. did not describe how they calculated the distance between the entities within their research.

A searcher swarm entity is also able to receive the navigational information, of the distance to the target and the information's age. If the information received by a searcher swarm entity is better than its stored navigational information, the searcher swarm entity will also update its navigational table.

The searcher swarm entity will then face the direction of the swarm entity that provided the better navigational information and travel forwards in this direction.

This process continues until the searcher entities locate the targets that they have been searching for.

As described, as swarm entities move around the operating environment, they receive information from other swarm entities and, if the information received is better than the information that they currently have stored within their navigation tables, they will update the information within their navigation tables. Likewise, the swarm entities also broadcast their navigation table information to other swarm entities that are within their broadcast ranges. If a swarm entity has moved, prior to broadcasting its navigation table information, it will add on its own distance moved to the estimated distance to the target within its navigation table, in order to attempt to maintain a realistic estimate of the distance value to the target.

Swarm entities that have been out of range of communications will update their navigation tables when they come back into range of swarm entities that have better navigation information. Their proposed swarm implementation is therefore able to update swarm entities that have either been operating away from the other entities within a swarm or they are newly introduced into the swarm, as detailed in swarm characteristic 8. When operating away from other entities within a swarm, where the different sections of the swarm contain different navigational information, one swarm section will contain better navigational information. Examples of this could be when a swarm is first deployed and the swarm is dispersed and there are swarm entities that are geographically removed from the majority of the swarm, or there are swarm entities that are at the physical extremities of the swarm and, because there was no better navigational information available, randomly chose to move in a direction that detached them from the main swarm. When swarm entities are newly introduced to the swarm, they will have yet to receive any navigational information from the already deployed entities if the swarm.

Figure 5.2 and Figure 5.3 show how two separate sections of a swarm can come together and then update the swarm entities navigational tables to the correct information. In Figure 5.2, the entities that are in direct communication with the target are communicating with current up to date information. The entities that do not have a communications path to the target are still communicating with each other but the information is either out of date or the entities have yet to receive any navigational information that they can pass on. The navigational table is populated with the current count figure, for the information age, and the shortest path to the target.

Figure 5.3 shows that, after one event count, two of the entities have moved close enough to each other to be within communications range. The navigational information is therefore updated throughout the swarm.

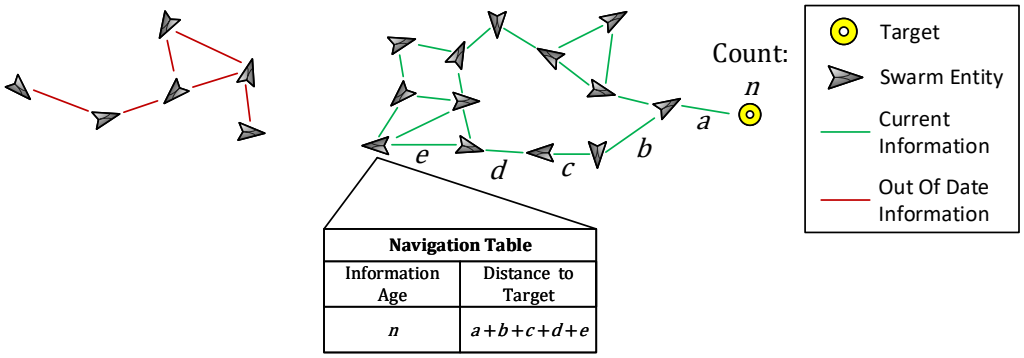


Figure 5.2: Information Distribution Prior to Join Up

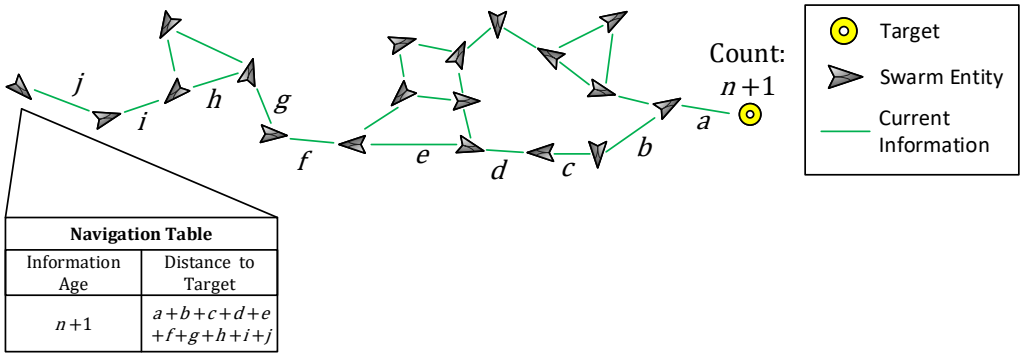


Figure 5.3: Information Distribution After Join Up

It is worth noting that the navigational information received by a searcher swarm entity might not actually be the most efficient route to a target's location. This is because the searcher swarm entity will face the direction of the swarm entity that provided the best navigational information, based on the information received from swarm entities within its communications range. This information regarding the best route to take can also change over time. This can be seen in Figures 5.4 and 5.5.

In Figure 5.4, the searcher is actually moving away from the target, as it believes this to be the best information. At time $t + n$, the swarm entities have moved and a better path is calculated, the searcher therefore changes direction accordingly, as shown in Figure 5.5.

As the searcher swarm entities travel, they continuously monitoring for received navigational information that is better than their stored navigational information. If no better navigational information is received, the searcher swarm entity continues in a straight line until it has travelled the distance it calculated between itself and the transmitting swarm entity it received the navigational information from. If no better navigational information is received, the searcher swarm entity essentially arrives at the location of that the transmitting swarm entity originally transmitted from.

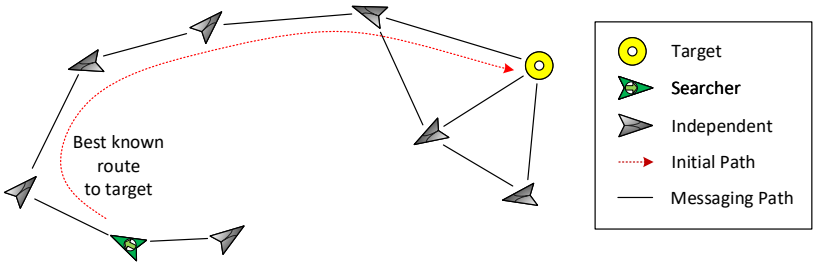


Figure 5.4: Longer Path, Time: t

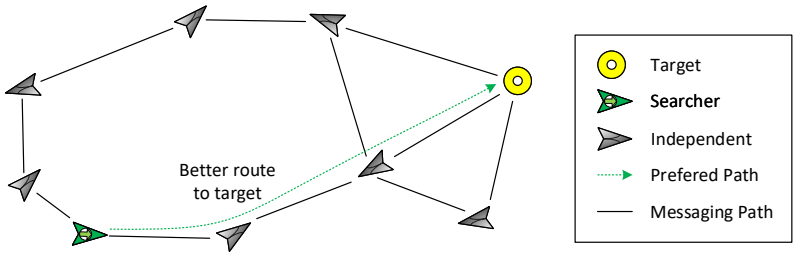


Figure 5.5: More Efficient Path, Time: $t + n$

Their research considered two actions that the searcher swarm entity can be pre-programmed to undertake:

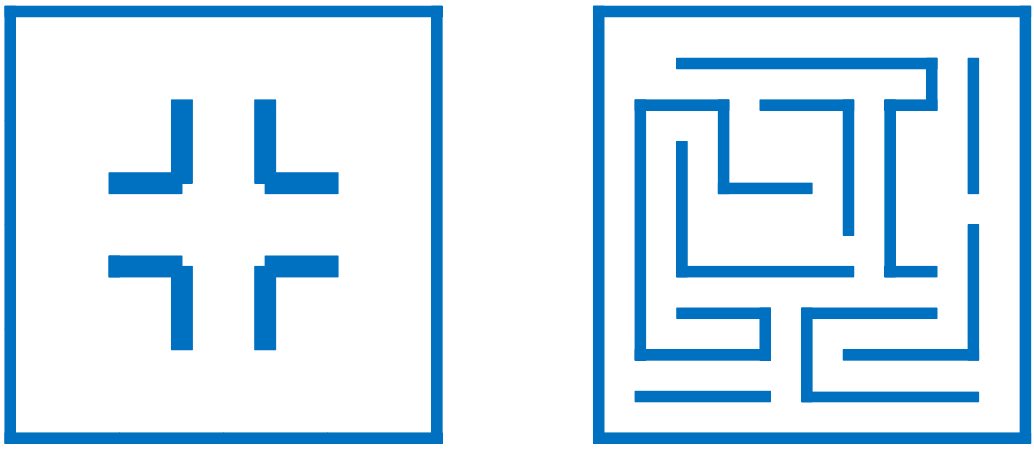
1. The searcher swarm entity stops and remains stationary, awaiting better navigational information.
   Ducatelle et al. referred to this as *Navigation with Stopping* (NwS).
2. Upon reaching the determined location, the searcher swarm entity randomly calculates a new heading and a random distance to travel and moves this calculated distance. The random distance that the swarm entity could calculate to travel being bounded a maximum distance. If no better navigational information is received during the searcher swarm entity's travels, the action is repeated.
   Ducatelle et al. referred to this as *Navigation with Random* (NwR).

Their research showed that a searcher entity was capable of locating a target entity, by utilising the simple algorithm. Also, the greater the number of other swarm entities that assisted the searcher entity, the less time that the task took to complete. This was more apparent when the searcher swarm entity carried out the Navigation with Random actions, if no better navigational information was provided to it.

Simulations where undertaken by Ducatella et al., where a randomly placed single searcher swarm entity had the goal of locating a randomly placed target. The simulations were performed utilising three different operating environments, each of which had a size of 20 by 20 units.

The operational environments simulated were:

- **No Maze** The operating environment was free from any obstacles and the swarm entities were capable of moving freely throughout the operating environment.
- **Simple Maze** The operating environment had the addition of a simple maze, as shown in Figure 5.6a, for the swarm entities to negotiate in order, to complete the task.
- **Complex Maze** The complexity of the maze in which the swarm entities needed to negotiate, in order to each the goal, was increased. This can be seen in Figure 5.6b.



(a) Simple Maze **(b)** Complex Maze

Figure 5.6: Operating Environment Mazes

The actions of the swarm entities that collide with another object within the simulation, such as a boundary or maze wall, was to act as if it were to "reflect" off, in a similar way to an elastic collision. This can be seen in Figure 5.7.

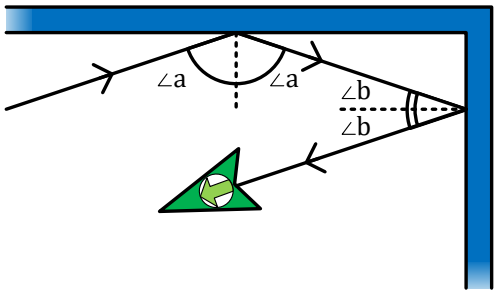


Figure 5.7: Searcher "Reflecting" Off Walls

As would be expected, the average time taken to locate a target increased as the complexity of the operating environment increased.

The goals of the searcher swarm entities were then modified, such that they had the task of alternating between targets to find. That is, a searcher would search for say target T_1 and, upon locating target T_1, would then switch to attempting to locate target T_2. Once target T_2 has been located, the searcher switches the target it is attempting to locate back to target T_1. This can be considered similar to a swarm in nature moving between a food source, represented by T_1 and a nest, represented by T_2.

The simulations of moving swarm entities between two targets where conducted with all the swarm entities being searchers, without the assistance of the other swarm entities.

Ducatelle et al. conducted these simulations in operating environments that did not contain any obstacles. A number of searchers were placed within the operating environment that were initially configured so that half of the searchers were looking for one of the targets and the other half were looking for the other target. Initially the searchers that could not receive navigational information would move in a random manner. Once navigational information was obtained, the searchers began moving towards their pre-defined target. Upon reaching a target, the searcher altered the target which it was searching for and the process repeated. This can be seen in Figure 5.11.

Once Ducatelle et al. had shown that the searchers could successfully navigate between the two targets that were unobstructed, they moved the location of the targets and placed an obstacle in the direct path between the targets. This provided two possible paths that the searchers to navigate between the targets. The operating environment can be seen in Figure 5.12 and it can be seen that the obstruction provides two possible paths between the two targets, that are of different lengths. The simulations by Ducatelle et al. demonstrated that once a simulation had been running for a period of time, the swarm entities would eventually start to utilise the shortest path between the two targets. This was not achieved by any external influence, attempting to inform the swarm entities the shortest path, but by the swarm entities determining the path to take themselves. This demonstrated that the algorithm was designed and achieved the requirement to find the most efficient path between a searcher and a target.

5.2.3 Simulations

I carried out simulations that replicated the original simulations that were conducted by Ducatelle et al. Further simulations were then undertaken within my research, in order to attempt to maliciously interfere with the swarm and to attempt to maliciously manipulate the swarm entities and therefore affect the overall swarm goal. The attacks against the swarm increased in complexity, from manipulating the environment, through to utilising knowledge of the swarm entities communications and utilising this against them. The simulations were undertaken using NetLogo 5.3.1 [141], which is a multi-agent programmable modelling environment. The NetLogo code produced for the simulations is available in GitHub [163].

My initial simulations of attacks, that attempted to realise the threat of Denial of Service by jamming, are included for completeness, in order to show how the research progressed. The majority of this section concentrates on the simulations of attacks that utilise knowledge of the swarm implementation and knowledge of how the swarm entities interact. This knowledge is then utilised by an attacker, in order to attempt modify the swarm's emergent behaviour, in order to reduce the efficiency of the swarm in achieving its goal.

As the simulations in the original research by Ducatelle et al. had not considered how ranging for distance calculations between entities was achieved, it was decided that this research would not consider this either. This was due to several reasons. The actual method of communications between entities would be dependent upon the actual implementation of the swarm, such as by using RF or LEDs. Ducatelle et al. utilised LEDs in their experiments with real robots, although the number of robots used was lower than that which was simulated, with a maximum of 8. They did note that the actual robots did have "noisy range and bearing estimates" but that their results showed that the navigation was successful and that the results demonstrated similar trends between the simulations and the robot experiments.

RF could also be utilised, with ranging being estimated from received signal strength and direction by using Direction Finding techniques. RF communication paths could be simulated, along with effects such as multipath, fading, Doppler shift, constructive and destructive interference and the properties of the transmitting and receiving aerials. However, these effects would be the same for both benign environments and environments that were subject to an attack. Also, any error correction techniques that could be employed to reduce the effects of channel losses and errors could be implemented by both the original swarm and a malicious actor.

The best case scenario for an attacker is for a "perfect" lossless channel. That is, any malicious transmission that is made by an attacker is received, without error, by a victim entity. Therefore, as the purpose of this research is to consider malicious activity upon a swarm all simulations undertaken did not consider any communications channel effects or link budgets.

5.2.3.1 Initial Simulations of Previous Research

Initial simulations were conducted in this research that replicated the original research undertaken by Ducatelle et al. within a benign environment. The results I obtained from the simulations were comparable to the original research and therefore demonstrated that the method of implementing the algorithms within this research, represented that of the original research.

The simulations I undertook were for a searcher to locate a target, utilising navigational information, that is provided by other, independent, swarm entities. This can be seen in Figure 5.1, where swarm entities that are within communications range of each other send messages containing navigational information between each other.

The simulation results I obtained show the average times for a single searcher to achieve its goal of locating a single target, with various operating environments and differing amounts of independent swarm entities that assist in the navigation task. The results shown in Figure 5.8 are for randomly placed searchers and targets, as with the original research. The results shown in Figure 5.9 are for randomly placed targets but with the searchers then placed at random directions but set distances from the target, to allow for repeatability of the experiments and to enable comparisons against subsequent simulations of attacks.

This research concentrates on a single searcher utilising cooperative navigation. This is because the multi-target cooperative navigation implementation followed the same methodology as used in the implementation for the single searcher cooperative navigation but only used searcher entities, as opposed to other entities and a single searcher entity. Successful simulations were carried out on multi-target cooperative navigation implementations, in order to successfully verify the simulations being undertaken in this research were comparable with the results obtained Ducatella et al. The remainder of this section therefore only considers single robot navigation implementations.

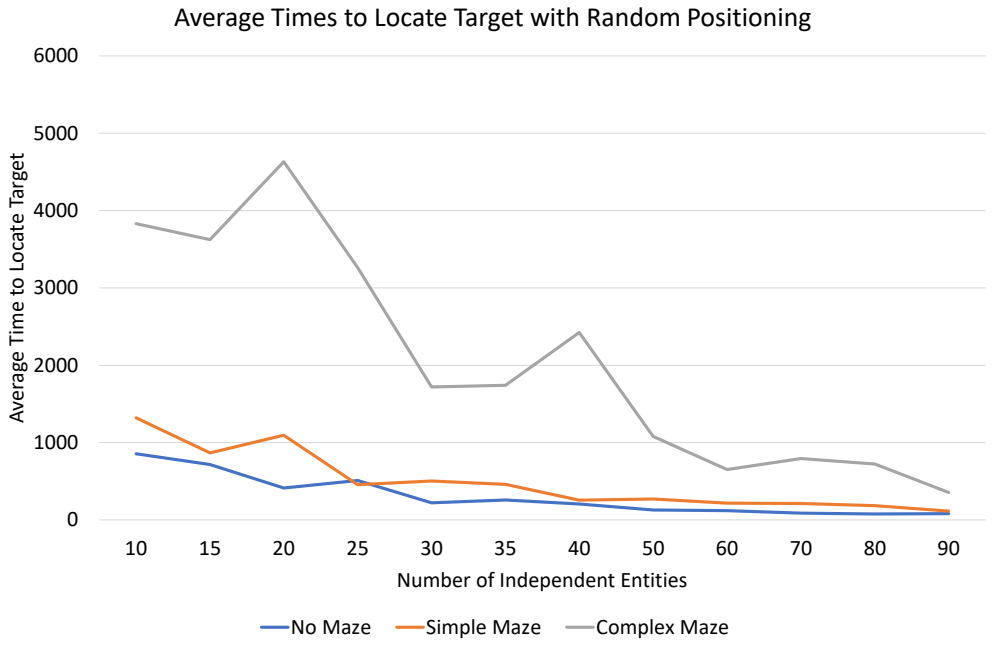


Figure 5.8: Baseline Results of Random Placements of Targets and Searchers

The simulations results, shown in Figure 5.10, verified the findings of Ducatelle et al., in that the Navigation with Random method of implementation enabled the target to be located more quickly when there were fewer assisting swarm entities, when compared with the Navigation with Stopping results. As shown in Figure 5.10, once there are less than approximately 20 independent swarm entities, the efficiency of the two methods start to diverge. When there were more than 25 independent entities, there were generally sufficient independent entities within the operating environment that independent entities would be constantly updating navigational information.

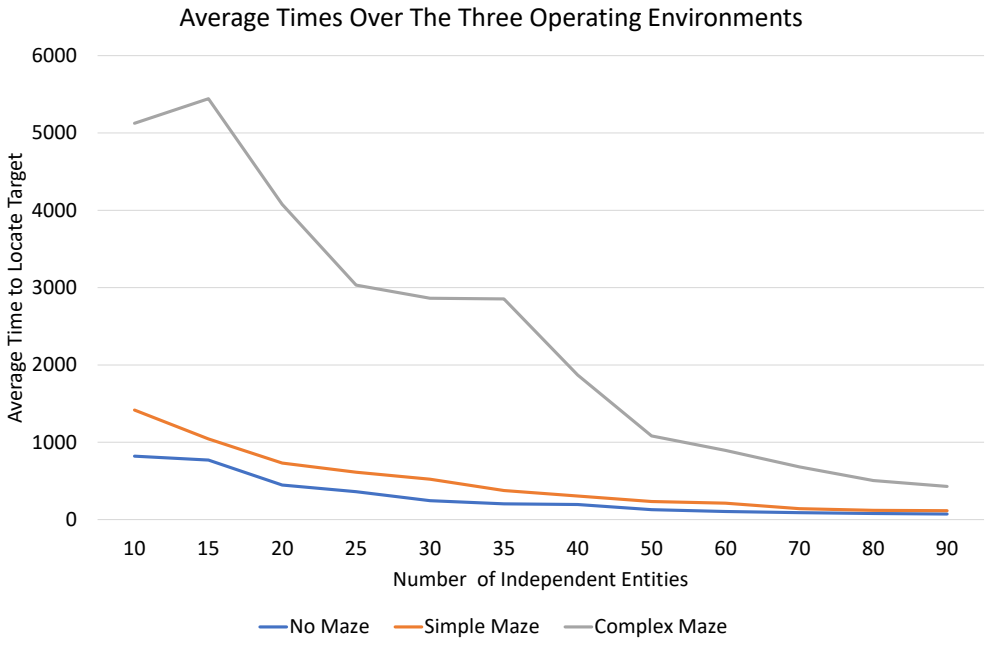


Figure 5.9: Averaged Results for Fixed Separations

In the case of NwS, if an independent entity reached its distance to travel and it had stopped to wait for new navigational information, the amount of other independent entities within the operating environment meant that the wait was generally not long, as another independent entity would pass within communications range of the stationary entity and provide new navigational information. When there were less than 20 independent entities, the NwS implementation had longer periods of time where an independent entity would be stationary and waiting for new navigational information. As the amount of independent entities to decreased, the time taken for a searcher to find the target in an NwS simulation increased. When NwS simulations were undertaken with less than 10 independent entities, the time taken for a searcher to locate the target could be either extremely large, compared to the NwR methodology, or it could fail to reach the target, as the searcher and independent entities had reached their maximum distance of travel and were waiting for new navigational information.

Because of this, it was decided to undertake all future simulations within this research by implementing the Navigation with Random implementations of the swarms. This was because the objective of this research was to attempt to maliciously manipulate the swarms, as opposed to trying to find the most efficient way for a swarm to attaining a goal. It was therefore felt that by utilising the Navigation with Random implementation, the swarm would have a better chance of reaching its goal and therefore attacks against the swarm would, if successful, be more apparent.

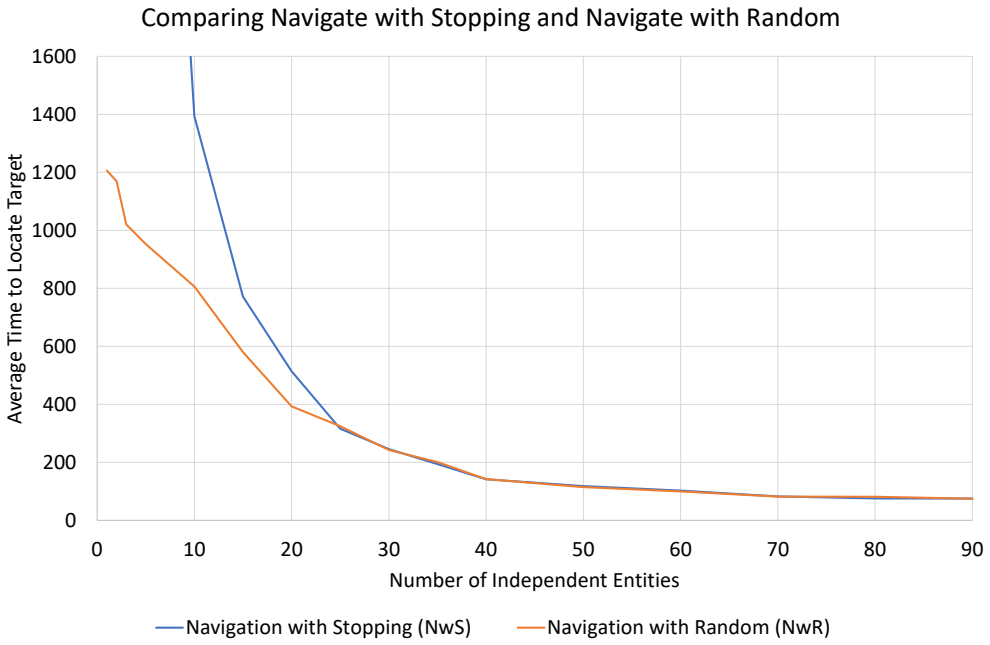


Figure 5.10: Expanded View - Comparing Performance of NwS and NwR

For completeness, the following figures show the searcher swarm entities moving between two targets. Figure 5.11 shows an unobstructed operating environment and Figure 5.12 shows an obstructed operating environment and demonstrates how the swarm entities, after a period of time, utilise the shortest path in order to realise their goal.

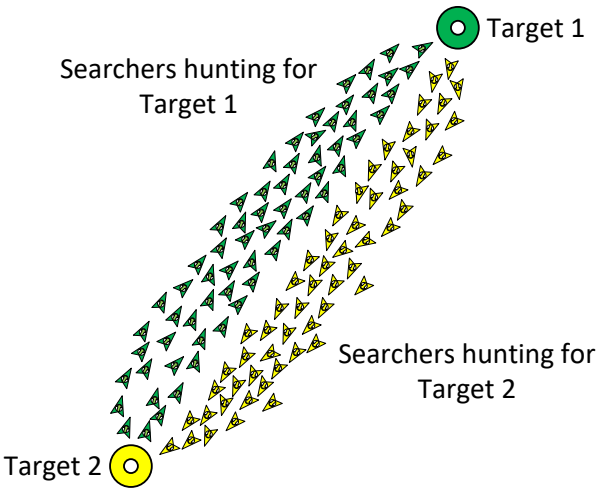


Figure 5.11: Two Targets - No Obstacles

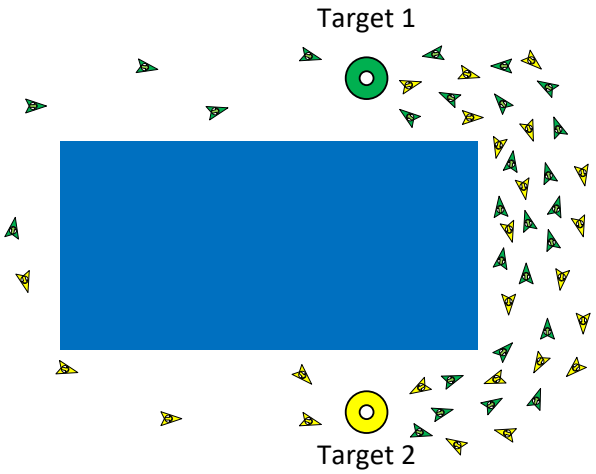


Figure 5.12: Two Targets - Obstacle in Operating Environment

5.2.3.2 Traditional Attacks Against the Swarm

Initial attacks that this research undertook against the swarms were basic jamming attacks, which were undertaken by randomly placed and static jammers within the operating environment.

The objective of the attack was to realise the threat of a Denial of Service against the swarm. The attack was undertaken by manipulating the physical communications within the environment, in order to prevent communications between the swarm entities. The aim of the attack was that a reduction in communications between the swarm entities would prevent navigational information from being effectively communicated and therefore reduce the efficiency of the swarm.

Simulations were undertaken for two attack implementations. The first was where the jammers realised the threat of masquerade and the jammers, when not undertaking jamming, would interact with and act as part of the original swarm.

In the second set of simulations, the jamming entities were stealthy and not visible to the swarm that was being attacked. Although it could be argued that once a jamming entity is actually undertaking jamming, it is detectable. These simulations were undertaken in order to gain an understanding of the effects of jamming on the swarm. However, for this research, it was not that interesting as, although there was a minimal effect upon the swarm, in that it could increase the amount of time taken for a searcher swarm entity to locate the target, it did not utilise any of the unique characteristics of a swarm.

The entities that undertook the jamming within the environment where increased from one to ten in number and the jamming period was fixed, at five time units.The area effected by the jamming was set to be the same as the communications range of the swarm entities and the chances of jamming were varied between 1% and 5%.

Although this is a very basic and conventional type of attack against communications, the swarm was able to cope with the attacks, due to its implementation. If a particular communication path between two entities was being jammed, there was often another path available to support the communication of navigational information. Also, if the communications to the searcher entity where prevented, this would not prevent the searcher entity from continuing to move. The movement would be to travel to the last known location of navigational information or, if no navigational updates were provided and the searcher reached the communications location of the navigational information, the searcher would pick a random direction and distance to travel. Whilst moving along the random direction of travel, the searcher is constantly attempting to update its navigational information.

Simulations undertaken showed that the malicious entities, that where undertaking the jamming, had little, if any, effect upon the swarm. Example of this are shown in Figure 5.13 and Figure 5.14, where the searcher was operating in an environment with a simple maze and is being subjected to an attack by fixed jammers that are not visible to the swarm entities, a 1% chance of jamming and a starting separation of 10 units.

What is shown by the results of the simulations is that the greatest effect on the efficiency of the searcher in completing its goal is the amount of independent entities present within the operating environment, as shown by Figure 5.13. To assist in the understanding of how limited the overall effect the jamming by the malicious entities has to the overall performance of the swarm, Figure 5.15 is a simple linear trend line analysis for Figure 5.14 and demonstrates the fairly stable general trend for the performance of the swarm in locating a target when compared to the number of malicious entities.

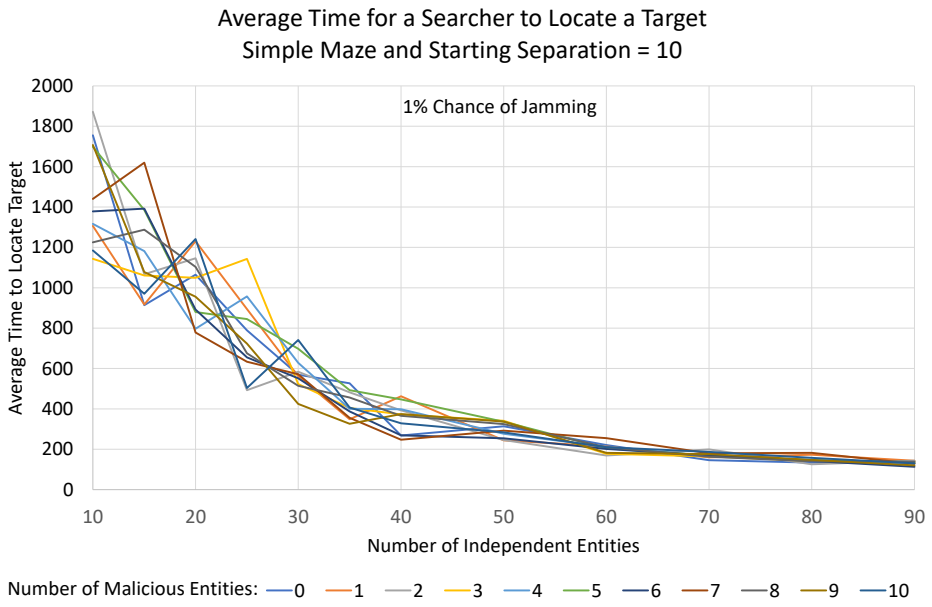


Figure 5.13: The Effects of Jamming on the Average Time to Locate a Target

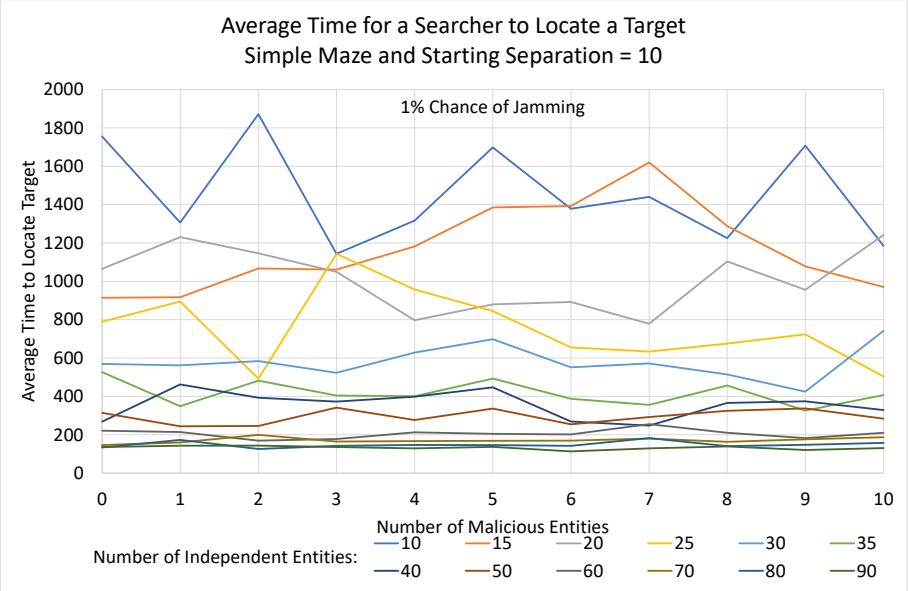


Figure 5.14: The Effects of Jamming by Comparing the Number of Malicious Entities

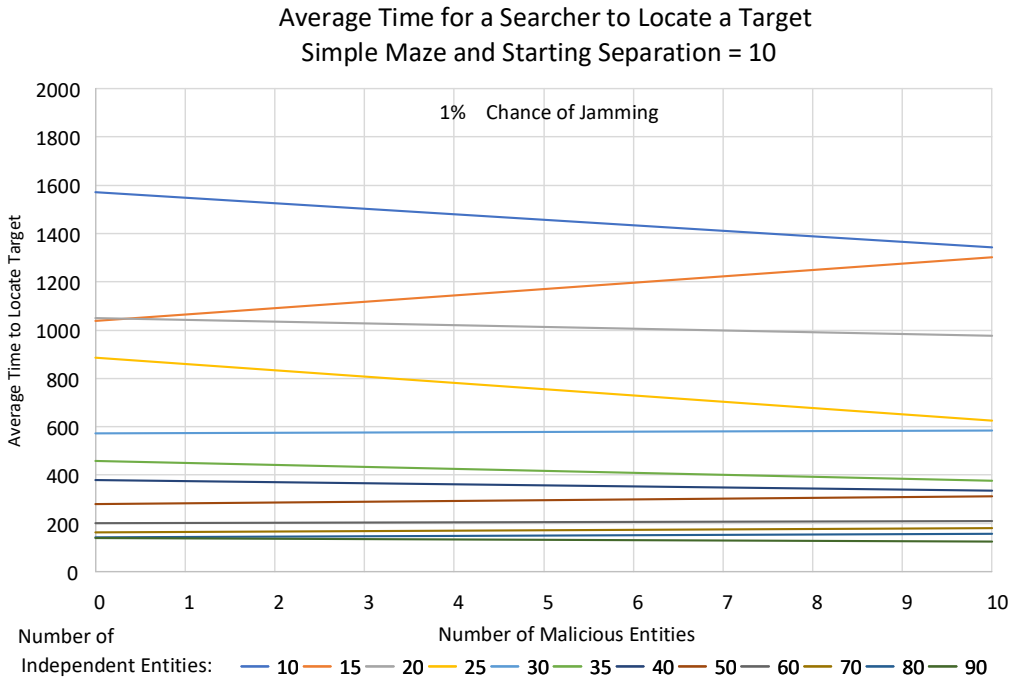


Figure 5.15: Linear Trend Line Showing the Effects of the Malicious Entities

Other results demonstrated that when the malicious entities were visible to the swarm, the efficiency of the searcher in locating the target slightly improved. This was because when the malicious entities were not undertaking jamming, they would contribute to the original swarm by acting as original swarm entities, in order to realise the threat of masquerade.

That is, if there were 10 entities in the original swarm and there were 10 entities in the malicious swarm, if there was no jamming activity being undertaken by any of the entities within the malicious swarm, the overall swarm size is 20 entities. There would therefore be twice as many entities contributing to the task of locating the target. Therefore, the malicious entities actually supplement the independent swarm, aiding the searcher and thus reducing its time to locate the target.

5.2.3.3 Modified Traditional Attacks Utilising Malicious Swarm Entities

In these simulations, the objective was to again realise the threat of Denial of Service. However, the malicious jamming entities were mobile within the operating environment.

Again, simulations were undertaken where the malicious entities were either independent of the original swarm, such that they did not interact with the original swarm and undertook jamming activities, or interacted with the original swarm, realising the threat of masquerade.

The simulations that were undertaken for static jammers were repeated for the mobile jammers and the results obtained were comparable.

As with the static jamming simulations, when a malicious entity interacted with the original swarm entities, the effect was that they assisted in the searcher swarm entity in achieving its goal of locating the target.

5.2.3.4 Review of the Traditional Attacks Against a Swarm

The jamming simulations showed similar results for all the different variations. That is, regardless of the amount of malicious jammers, whether the jammers were stationary or mobile, whether the chance of jamming was for 1% or 5% and for any operating environment, which included no maze, simple maze and complex maze, the results all followed the same profile. The results showed that the main factor that affected the efficiency of the searcher locating the target was the amount independent searchers that would assist in the task and that there were limited effects caused by the jamming. When observing the simulations the jamming effects could be seen, in that jammed communications paths prevented the communication of navigational information. However, there were either other communications paths available or an entity, either the searcher or an independent entity, would just continue to move and would move out of the jammed area.

Due to the results and the simplicity of the jamming attacks, it was decided to concentrate this research on attacks that utilise the unique characteristics of a swarm.

5.2.3.5 Attacks that Manipulate the Behaviour of a Swarm

The initial attack simulations undertaken, that considered jamming, were essentially to understand if the efficiency of a swarm could be reduced by an external and malicious influence.

However, the area of interest for this research is to understand if a malicious attacker can undertake an attack that is specific to a swarm implementation. The attacks will attempt to utilise knowledge of how a swarm entity will interact with another swarm entities and utilise this knowledge to the attacker's advantage, in order to attempt to effect the swarm entities and ultimately the emergent behaviour of the swarm as a whole.

Several attacks were considered. These included:

- **Random Walker** A malicious entity is deployed within the original swarm that randomly moves around the operating environment and broadcasts false navigational information, which appears better than the actual navigational information.
- **Hunter Jammer** A malicious entity attempts to locate the target before the searcher and then jams the communications surrounding the target.
- **Hunter** A malicious entity is deployed that has a slightly longer communications range than the independent swarm entities. This allows the malicious entity to detect and block a target before a searcher can locate the target.
- **Hunter/Runner** A malicious entity is deployed within the original swarm that behaves as though it were a searcher swarm entity. Upon locating a target, the malicious entity moves away from the target, whilst broadcasting better, but false, navigational information.
- **Hunter/Runner with Stop** Similar to the Hunter/Runner, in that upon locating a target it moves away from the target and broadcasts better, but false, navigational information. However, when the malicious entity locates a wall, it stops and remains stationary.

An aim of these proposed attacks was that they should be simple to implement, as they should be able to be carried out by swarm entities and therefore be subject to the same constraints as other swarm entities.

Random Walker The initial simulations that attempted to manipulate the behaviour of a swarm involved randomly placing a malicious entity within the swarm's operating environment. The malicious entity was configured to broadcast navigational information that appeared better than the true navigational information.

Essentially the malicious entity had realised the threat of masquerade, in order to carry out the attack of manipulating a swarm entity's behaviour, by conducting a misinformation attack.

The false navigational information being broadcast by the malicious entity appears better than the true navigational information, that is being transmitted by the target, and this false information is communicated through the swarm by the independent swarm entities. When the false information is received by a searcher swarm entity, the searcher will attempt to follow the false navigational information. The searcher swarm entity is essentially now searching for the malicious entity, as opposed to the target.

Although this was a simple attack, the impact it had was significant. The minimum percentage increase in the average time taken for the searcher to reach its goal of locating the target was 240%, where as the maximum impact was over 21,000%. There was also a general trend that as the number of independent entities increased, the time taken to locate the target also increase. This can be seen in Figure 5.16.

There was also a lower number of searchers that achieved their goal in the simulations. However, this impact was also due to the increase in times taken for searchers to achieve their goals and the simulations timed out.

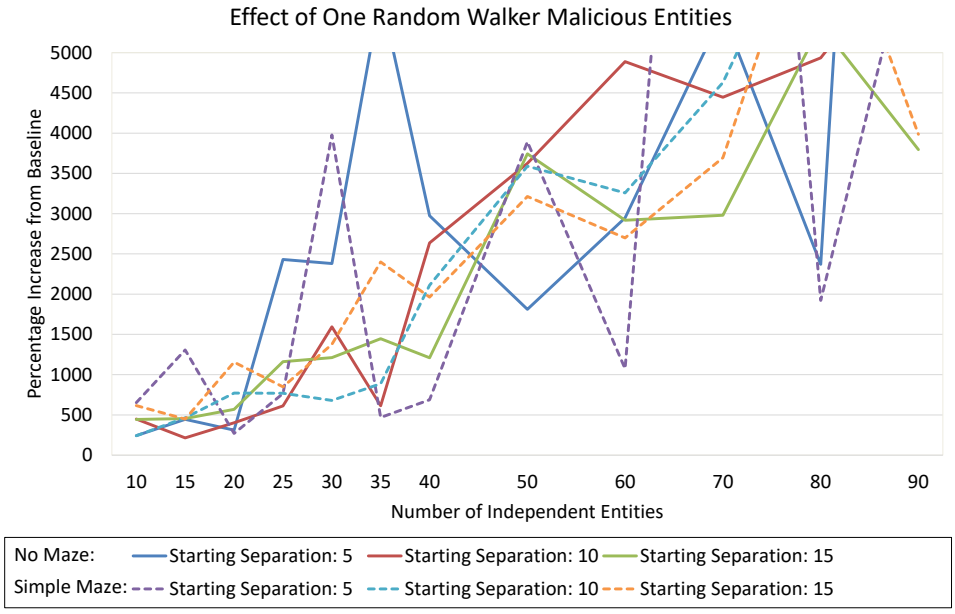


Figure 5.16: Impact on Searcher from One Random Walking Malicious Entity

What can be observed for the simulation results, and from observing the simulations whilst they were being undertaken, is that the time taken for a searcher entity to locate a target is essentially the time taken for the malicious entity, that the searcher swarm entity is following, to pass a target location. That is, the searcher swarm entity will direct itself, based on the received navigational information. The best navigational information will be the false navigational information that has be broadcast by a malicious entity and the searcher swarm entity will therefore attempt to make its way the source of the false navigational information.

The source of the false navigational information, a malicious entity, will continue to move in random manner and therefore the searcher swarm entity will continue to follow the malicious entity, in order to locate the source of what the searcher interprets as the best navigational information. Eventually, the random movements of the malicious entity will cause the malicious entity to pass by an actual target, such that the searcher swarm entity is within range of that target and will therefore locate the target and accomplish its goal. The time taken for the searcher swarm entity to locate a target is therefore essentially the time taken for a randomly moving malicious entity to pass by a target location.

Although the effect on the efficiency of the searcher swarm entity by an attack from an attack by a random walker was apparent, it was not considered to be a subtle or challenging attack. That is, although it utilised knowledge of how the swarm entities interacted, it was not that complex in its approach of how it could attempt to manipulate the swarm's emergent behaviour.

Hunter Jammer An attack was considered where the attacking malicious entities would attempt to locate the target, essentially behaving in the same manner as a searcher, until it located a target. Once at the target, the malicious entity would then begin to prevent communications from the target by jamming all the communications from the target.

When a malicious entity located the target prior to the searcher, then the searcher never achieved its goal of locating the target. This was because once the targets communications had been prevented, no navigational information could be communicated to the swarm and if a searcher passed the target, through the searcher's random movements, the target was not able to inform the searcher that it was the target. The principle of the hunter jammer can be seen in Figure 5.17.

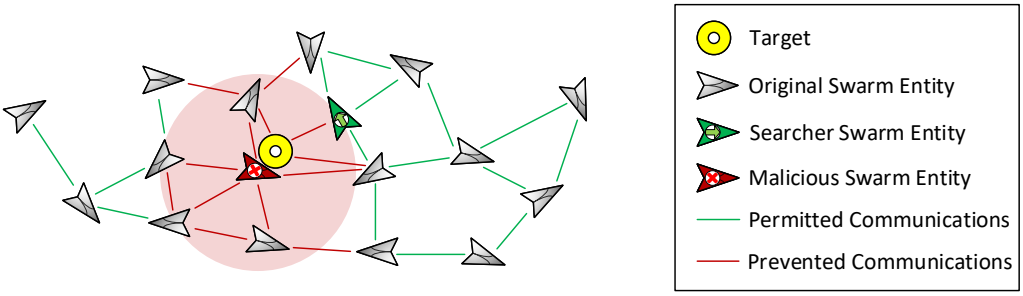


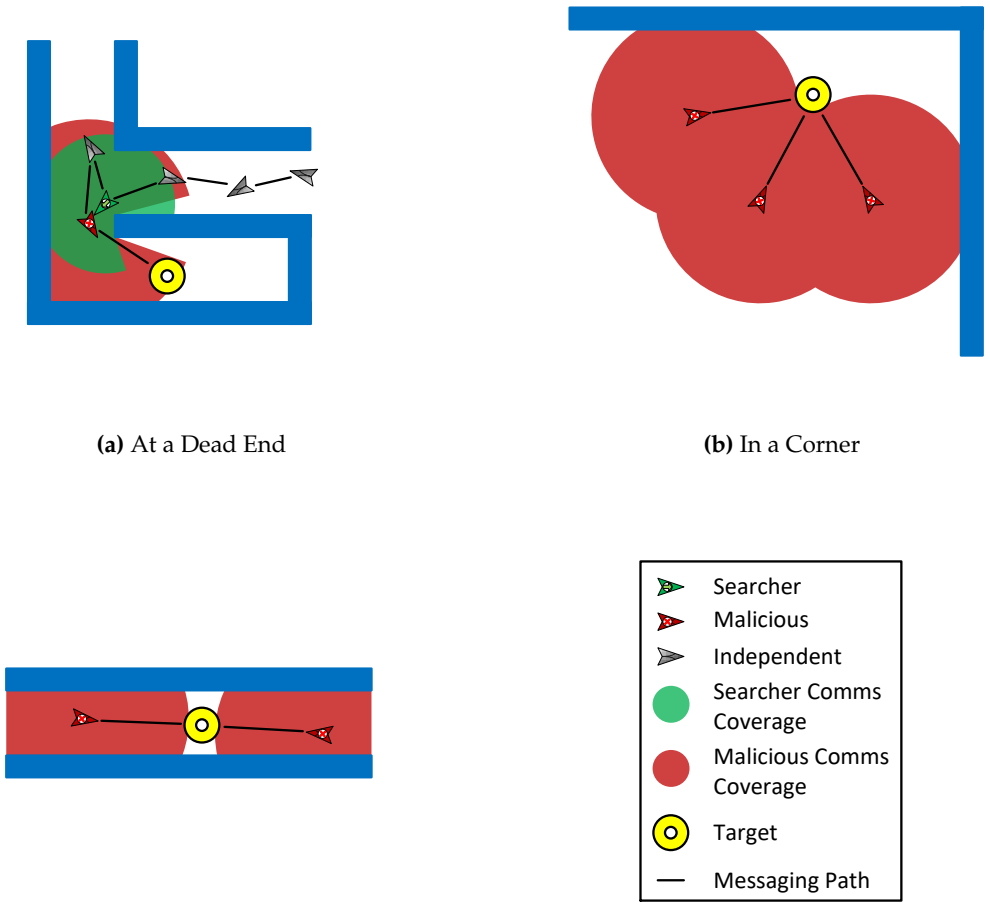
Figure 5.17: Principle of the Hunter Jammer

Although this was a very effective attack, the attack was not subtle and, although it made use of the navigational information provided by the independent entities, the attack do not utilise the unique characteristics of the swarm, such as emergent behaviour, in order to prosecute the attack.

Because of this, it was decided not to continue researching this attack method.

Hunter This attack built upon the Hunter Jammer attack, in that the malicious entities attempted to locate the target before the searcher. In this attack, the malicious entities were modified, such that they had a slightly larger communications range, compared to the searcher and the target. The malicious entities were configured such that initially they were not visible to the other swarm entities, but they would utilise the other swarm entities navigational information broadcasts to assist in locating the target. Once a malicious entity had located a target, it stopped a distance away from the target that was slightly further than the communications ranges of the searcher and the target. At this stand off distance, the malicious entity would make itself visible within the swarm and then broadcast navigational information that was better than that of the target.

This resulted in searchers becoming trapped at the location of the malicious entities, outside of the communications range between the searcher and the target. The effect was greatest nearer to walls, especially were operating environments contained mazes. This effect is shown in Figure 5.18.



(a) At a Dead End

(b) In a Corner

(c) Along a Path

Figure 5.18: Hunters Isolating Targets

Figure 5.18a shows how a malicious entity has trapped a searcher, preventing the searcher from going down the dead end and locating the target. Figure 5.18b shows how several malicious entities have located such that the target cannot be reached, as any searchers would be attracted to the malicious entities and then become trapped and Figure 5.18c shows how malicious entities have positioned themselves either side of a target along a path. Any searcher would locate a malicious entity first and therefore become trapped at the malicious entity.

The effect of the Hunter attack was that fewer searchers managed to complete their goal of locating the target. The greater the number of malicious entities, the less efficient the searcher was at achieving its goal of locating the target. This can be seen in Figure 5.19 and Figure 5.20.

The interesting observation is that although there is a noticeable degradation in efficiency, as the number of malicious entities increase, the malicious attack undertaken on an operating environment with no maze present produces better results for the attacker, when compared to that of the simple maze.

When considering how the numbers of malicious entities can have an effect upon the searcher achieving its goal, it can be seen in Figure 5.21 that in an operating environment that contains no maze, the majority of the impact can be realised by approximately 4 malicious entities. The effects of the numbers of malicious entities, based on the amount of independent entities, tends to be fairly constant.

In an operating environment that contains a simple maze, a reduction in the efficiency of the searcher achieving its goal can be seen in Figure 5.22. The results are tending to show that the effect of the increasing number of malicious entities tends to lessen around seven to eight malicious entities.

However, in both environments, the effect of the malicious attackers is very apparent. In an operating environment with no maze, an attacker only needs to deploy 4 malicious entities and the efficiency of the searcher in achieving its goal drops to less than 20% and possibly as low as 1%. In an operating environment with a simple maze, the deployment of 4 malicious entities would reduce the efficiency of the searcher in achieving its goal to at least 30% and possibly as low as 10%. In order to achieve a reduction of efficiency of the searcher achieving its goal to less than 20%, eight malicious entities would need to be released.

What was also noted in the results was that when a searcher did successfully achieve its goal of locating the target, the average time taken to complete this was less than if there had been no malicious entities within the operating environment. This was due to the malicious entities having larger comminations ranges that would broadcast the navigational information to independent entities, that would not normally have received the navigational information. If a searcher receives this information, it will assume that it is genuine and act accordingly and move towards the source of the navigational information.

This can cause the searcher to be moved towards a targets location earlier than it would have normally and can cause the searcher to then locate the target.

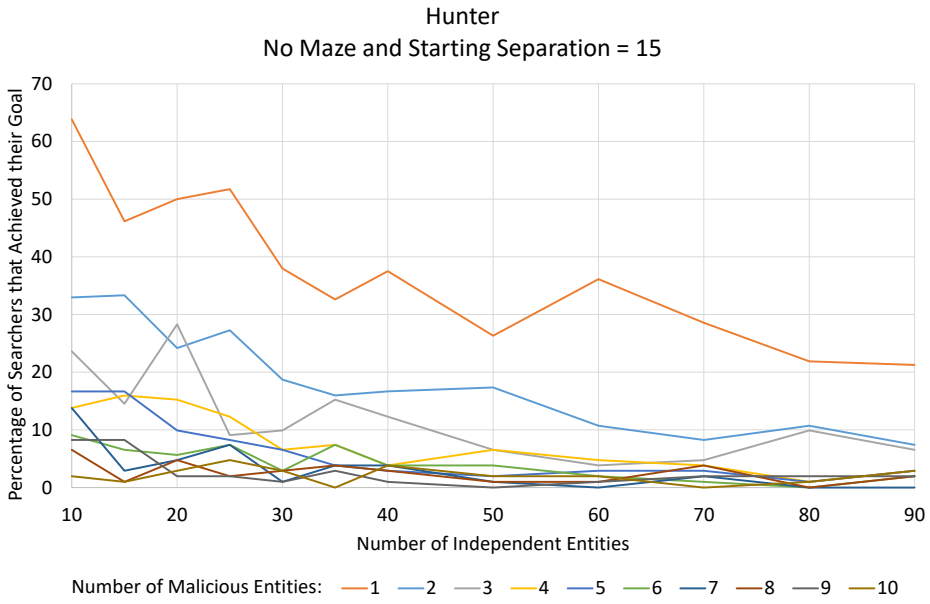


Figure 5.19: Average Amount of Searchers that Achieved their Goal: No Maze

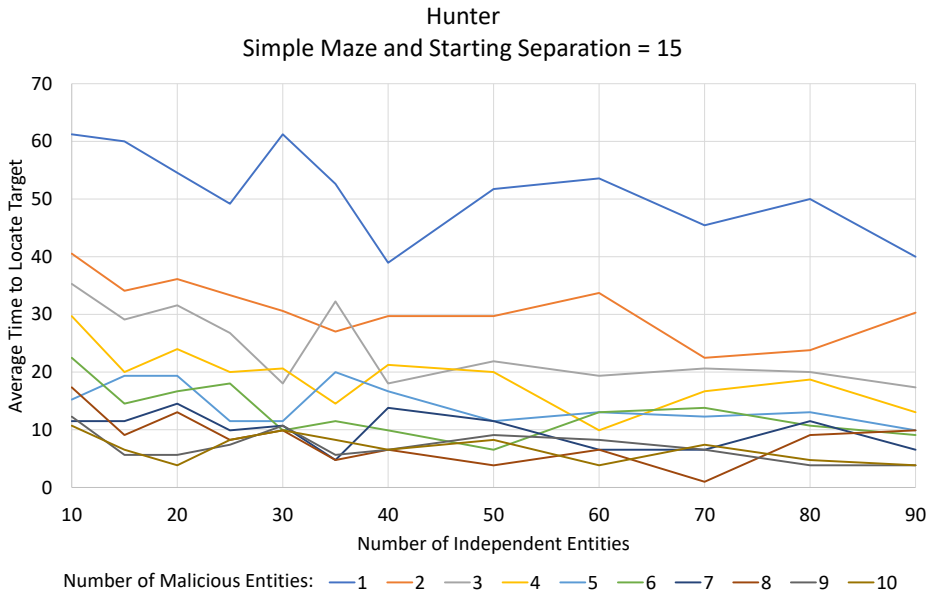


Figure 5.20: Average Amount of Searchers that Achieved their Goal: Simple Maze

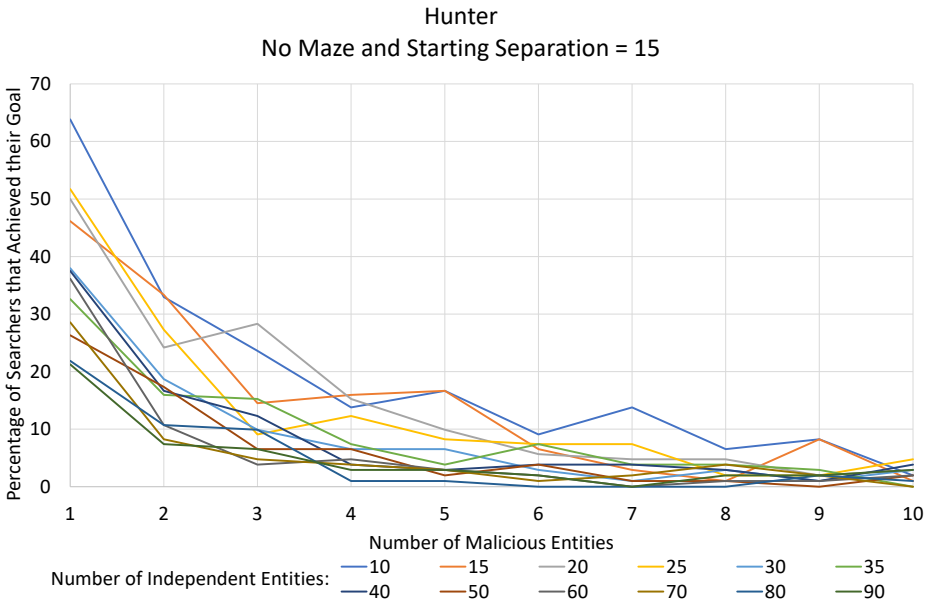


Figure 5.21: No. of Malicious Entities and their Effect on Searcher Efficiency: No Maze

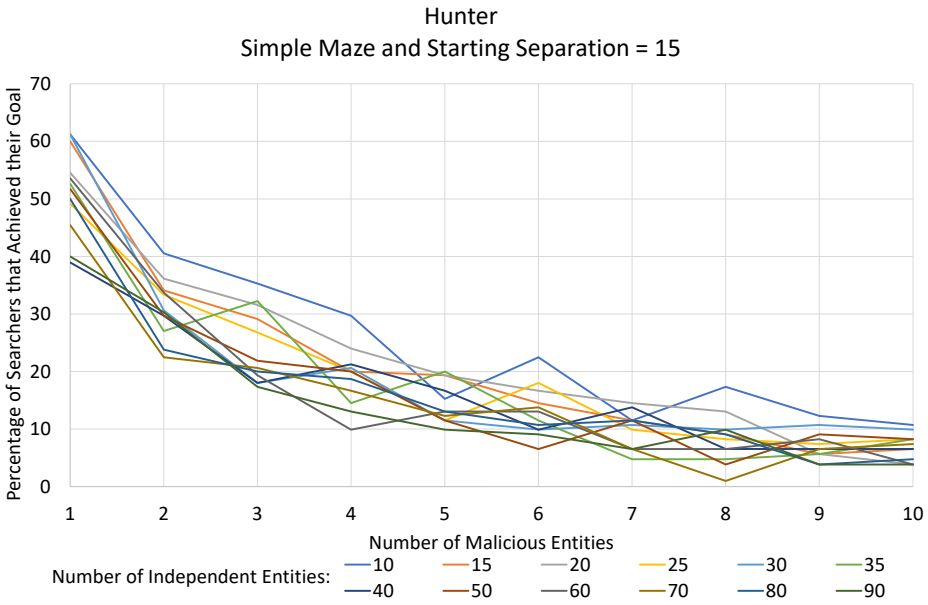


Figure 5.22: No. of Malicious Entities and their Effect on Searcher Efficiency: Simple Maze

However, the overall efficiency of the searcher is still reduced, as there is an impact on the searcher actually achieving its goal.

Hunter/Runner A new attack was considered where the attacking malicious entities would attempt to locate the target, again essentially behaving in the same manner as a searcher, and then attempt to lure searchers away from the target. In this simulation, the communications range of the malicious entity was the same as all other swarm entities. Therefore, unlike the Hunter example, if the malicious entities were captured and modified entities from a previous swarm, the modifications required would be software modifications and not potential hardware modifications, such as altering the communications provision.

The attack is conducted as follows, a malicious entity acts like a searcher, in order to locate a target. This can be seen in Figure 5.23.

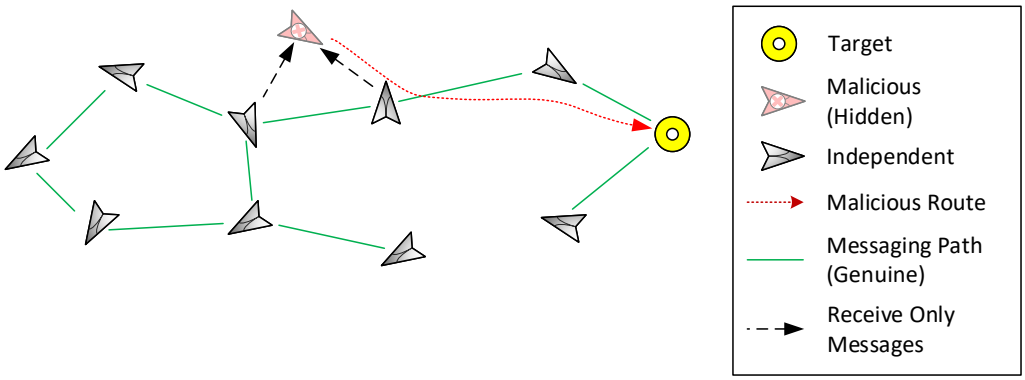


Figure 5.23: Malicious Entity Hunting for a Target

Once the malicious entity locates a target, as shown in Figure 5.24, it turns away from the target and decrease its information age and increase its distance to target value. This change to worse navigational information is to attempt to either not attract a searcher entity or not to pass any useful navigational information in to the swarm itself. The malicious entity is essentially attempting to look "unattractive". The malicious entity then travelled in a straight line until it had moved a distance that was greater than both the malicious entity's and target's communications ranges, as shown in Figure 5.25.

Once the malicious entity had travelled beyond the communications range of the target, the malicious entity then altered its navigational information, such that the information's age appeared greater than the target's information age value and the distance to the target was reduced to a low figure.

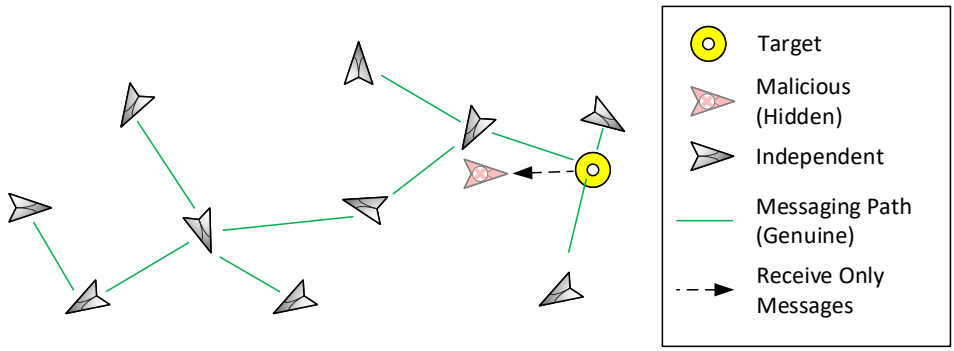


Figure 5.24: Malicious Entity Locates a Target

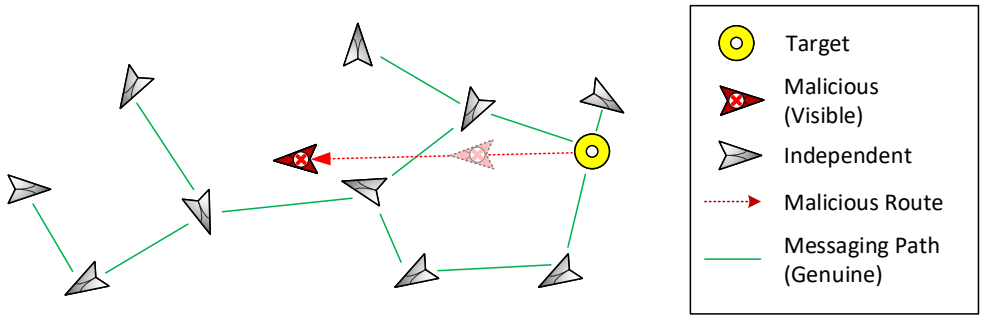


Figure 5.25: Malicious Entity Moves Away from Target

This ensured that the malicious entity would be transmitting false navigational information that contained better navigational information than the actual target that the malicious entity had located. The attacker then continued to move in a straight line, in order to try and make the searcher swarm entities follow the false navigational information and move in a direction that was away from the actual target location. This can be seen in Figure 5.26.

This was again a realisation of the threat from masquerade, in order to carry out the attack of manipulating a swarm entity's behaviour by conducting an attack by misinformation.

The initial phase of the attack, where the malicious entities are behaving like a searcher, simulations were conducted where the malicious entity could operate either independently of the original swarm entities or interacted with the other swarm entities. When the malicious entity acted independently, it received and then acted upon the navigational information that it received, in order to locate a target, but it did not broadcast any navigational information. When it was simulated in a mode of operation where it interacted with the other swarm entities, it did broadcast true navigational information to other swarm entities that were in communications range whilst hunting for the target.

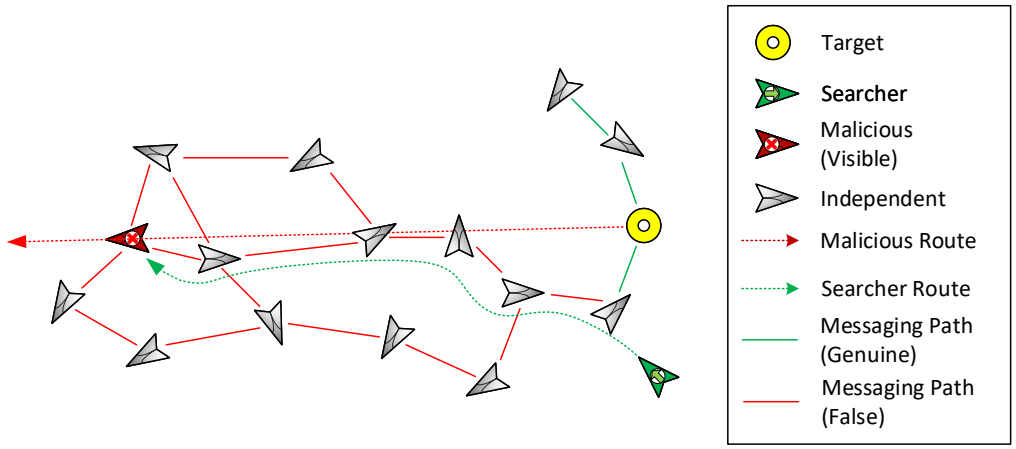


Figure 5.26: Malicious Entity Provides False Information

Obviously, regardless of the mode of operation within the initial phase of the attack, the malicious attacker would broadcast false navigational information in the second phase of the attack, when the malicious entity attempts to lure searching entities way from the target.

From the results obtained from the simulations, it was apparent that the attack had greater success when the malicious entities were hunting whilst not visible to other swarm entities. Therefore, the research concentrated this attack method.

Simulations were conducted within all three operating environments, each of size 20 by 20 units, which contained either no maze, a simple maze or a complex maze. The simulations undertaken considered searcher to target starting separations of 5, 10 and 15 units.

The results showed that the searcher would eventually always reach the target's location. However, the average time required to locate the increased when malicious entities were introduced. As examples, Figure 5.27 and Figure 5.28 show how for the environment with either no maze present or a simple maze present and a starting separation distance of 15 units, the time taken to achieve the goal of locating the target increased, once there were more than 30 independent entities present. The malicious entity's goal of reducing the efficiency of the original swarm was therefore achieved, in this situation.

What is also demonstrated in Figure 5.29 and Figure 5.30, is that in order to achieve a successful attack in these circumstances, there is only need for one or two attacker to be present in the operating environment. The effect of further malicious swarm entities does not a significant effect on the overall performance of the attack. Knowledge of which could be used by an attacker, so an attackers would not deploy more malicious entities than are actually required in order to undertake a successful attack.

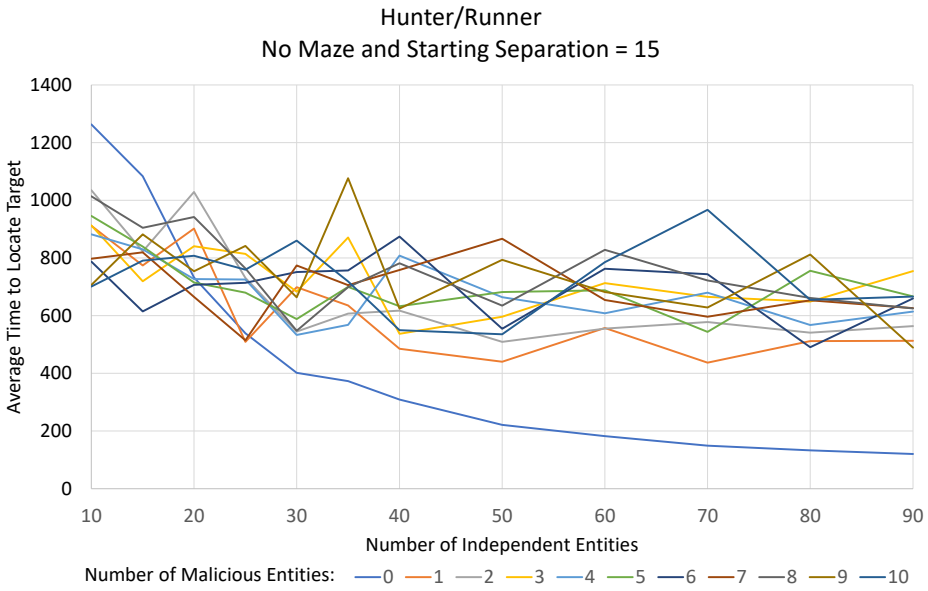


Figure 5.27: Average Time to Locate a Target: No Maze

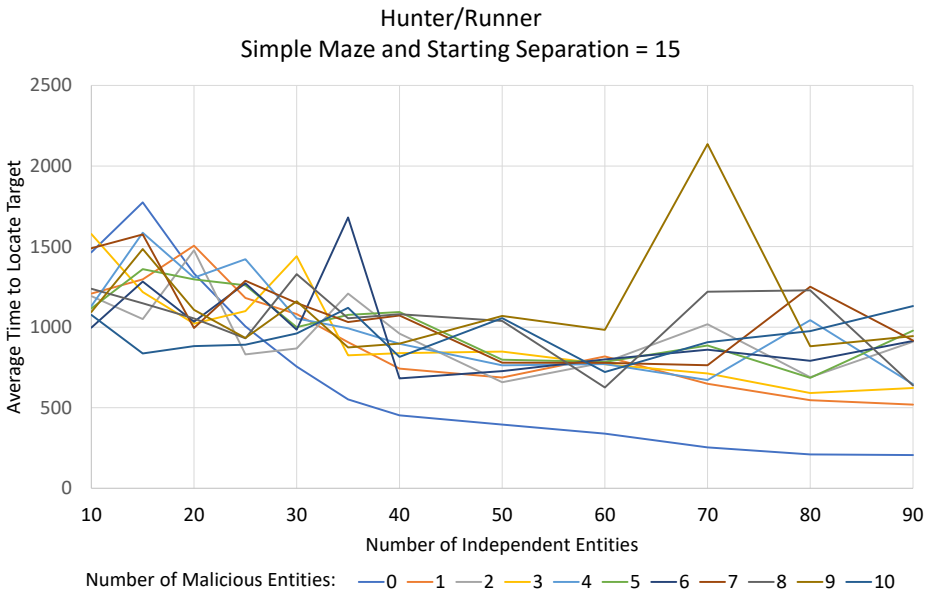


Figure 5.28: Average Time to Locate a Target: Simple Maze

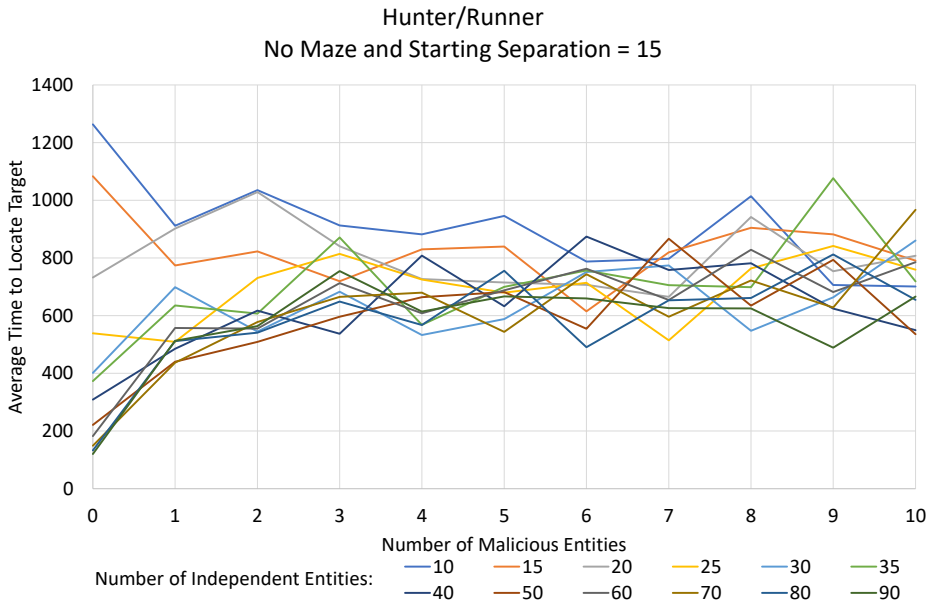


Figure 5.29: Number of Malicious Entities to Effect Efficiency of Searchers: No Maze

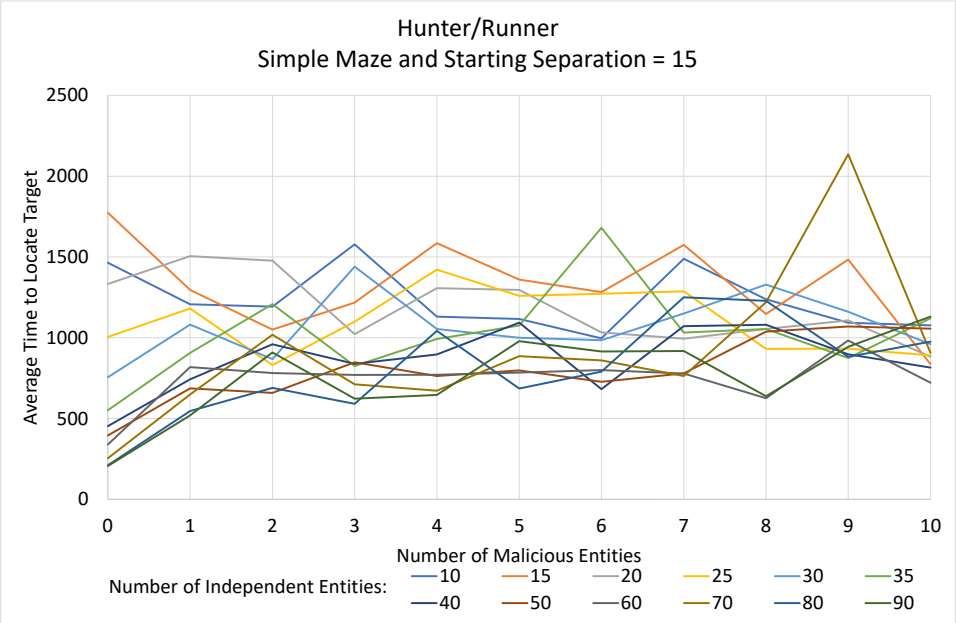


Figure 5.30: Number of Malicious Entities to Effect Efficiency of Searchers: Simple Maze

When there are no obstacles within the operating environment, the Hunter/Runner has less of an effect upon the efficiency of the searcher to locate a target. This was because the simulated operational environment was kept to be the same as that described within the paper by Ducatelle et al., which is essentially a relatively small operating environment. Therefore, if a malicious entity locates a target, it moves away in a straight line. The malicious entity then moves towards a wall, reflects off the wall and often heads back in the general direction of the target's location.

However, the principle of the attack was realised and this attack could therefore be used effectively within larger operating environments.

Hunter/Runner Wall Stop After observing the results for both the Hunter and the Hunter/Runner simulations, it was decided to slightly modify the Hunter/Runner attack. As with the Hunter/Runner implementation, when a malicious entity detects a target, it moves away from the target and, once it is a suitable distance away from the target, broadcasts better navigational information. However, in this new attack, when a malicious entity that is moving away from a detected target reaches a wall, it stops. It then continues to broadcast false navigational information, in a similar fashion to the Hunter simulation, in order to attract the searcher. The communications ranges were not extended, as they were with the Hunter, and were kept to be the same as all the other swarm entities.

The reason for stopping upon the detection of a wall, was that this would then prevent the malicious entity from reflecting off the wall and return in the general direction of the target. The stationary malicious entities would then attract the searchers to their location, as opposed to potentially leading the searchers back to the target.

The result of the simulations for an operating environment with no maze can be seen in Figure 5.31 and what is immediately apparent is that for a low number of malicious entities, as the size of the independent swarm entities increases, the malicious attack becomes more effective, as less searchers reach their goal of locating the target. This is demonstrated by the results for a single malicious entity with 90 independent entities having the same effect on efficiency as 6 malicious entities when there are 20 independent entities.

As the environment has no maze, there are no obstructions to communications, other than the swarm entities being out of range of one another. Therefore, as the number of independent swarm entities increase, the majority of the operating environment has independent entities that are able to communicate navigational information between each other. As soon as an independent entity encounters a malicious entity with false navigational information, this false navigational information is then distributed to cover almost all of the operating environment and any entities that are operating within it, as this purports to be the most efficient navigational information in order to reach the target . The malicious swarm entities are essentially utilising the operating environment and the independent swarm to their advantage.

Figure 5.32 shows how in an operating environment with a simple maze, the effect of the malicious entities is fairly constant, regardless of the number of independent entities. This is because the independent entities are constrained in the amount of entities that they can communicate with, as the maze acts as an obstacle to their communications and therefore reducing the overall communications coverage.

The amount of malicious entities required to produce a reduction in the searcher's efficiency can be seen in Figure 5.33 and Figure 5.34. What is shown is that when the operating environment contains no maze, a high number of independent entities has a greater effect. This is demonstrated in Figure 5.33, when there are 90 independent entities, the swarm fails to achieve its goal when there are 10 malicious entities. Essentially, there will always be a malicious entity with the range of an independent entity and the searcher will always navigate to a malicious entity instead of the target.

When the searcher had to negotiate a simple maze and there was no direct route to follow, when following the navigational information, this could result in an increase in distance to for the searcher to travel and the searcher coming within range of a randomly positioned target, and therefore locating the target. This is why there is a slight increase in efficiency of the searcher in achieving its goal and locating the target when the operational environment contained the simple maze.

However, regardless of the operating environment, increasing the number of malicious entities has the effect of reducing the efficiency of the swarm achieving its goal of locating the target. The number of independent entities has the greatest influence on the success of the attack, which is more apparent when there are few malicious entities and a greater number of independent entities.

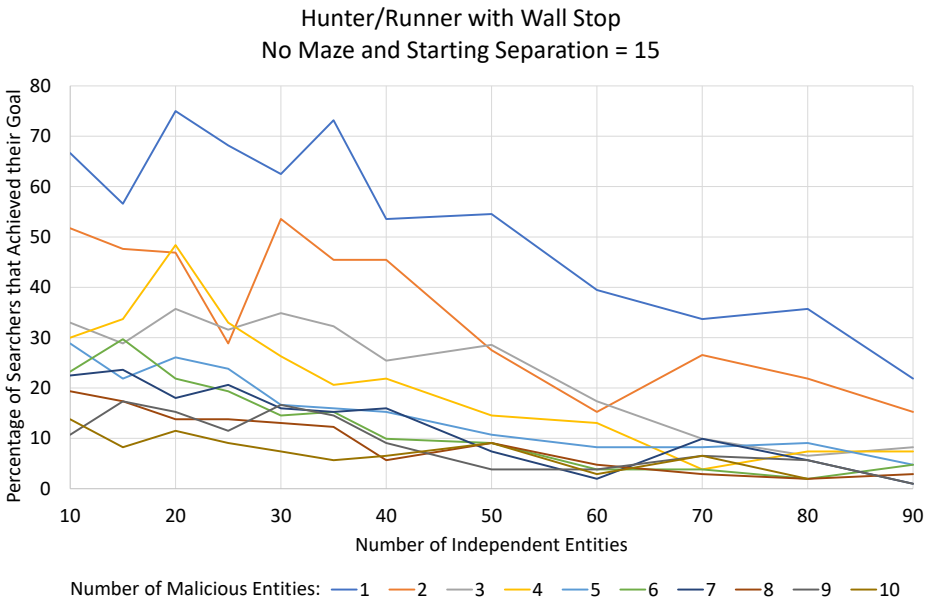


Figure 5.31: Average Percentage of Searchers that Achieved their Goal: No Maze

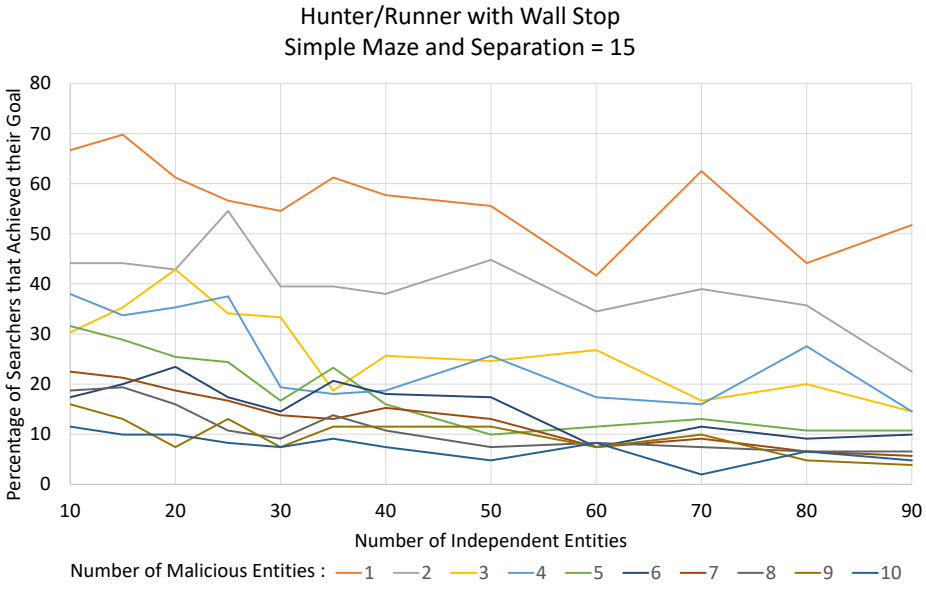


Figure 5.32: Average Percentage of Searchers that Achieved their Goal: Simple Maze

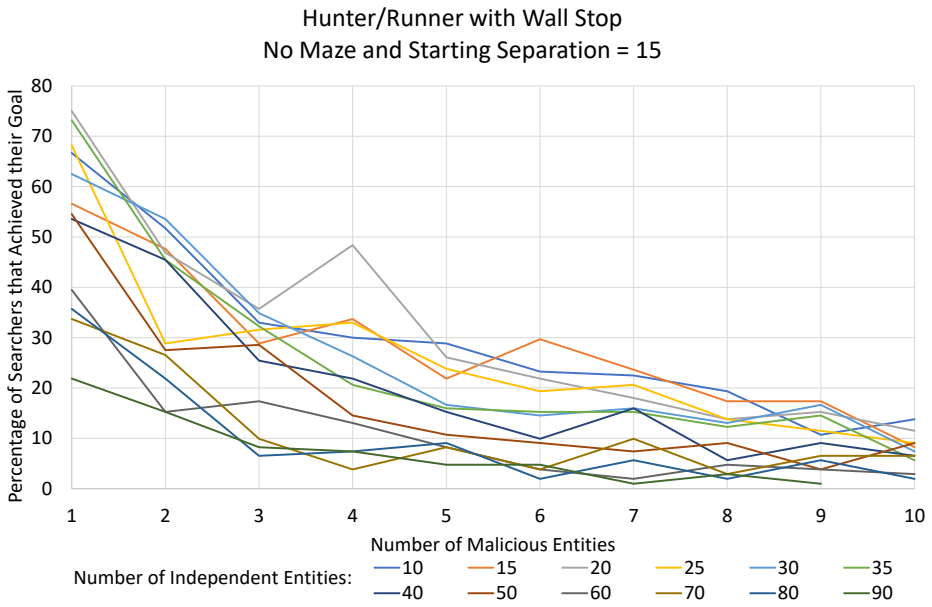


Figure 5.33: Number of Malicious Entities Required to Effect Efficiency of Searchers: No Maze

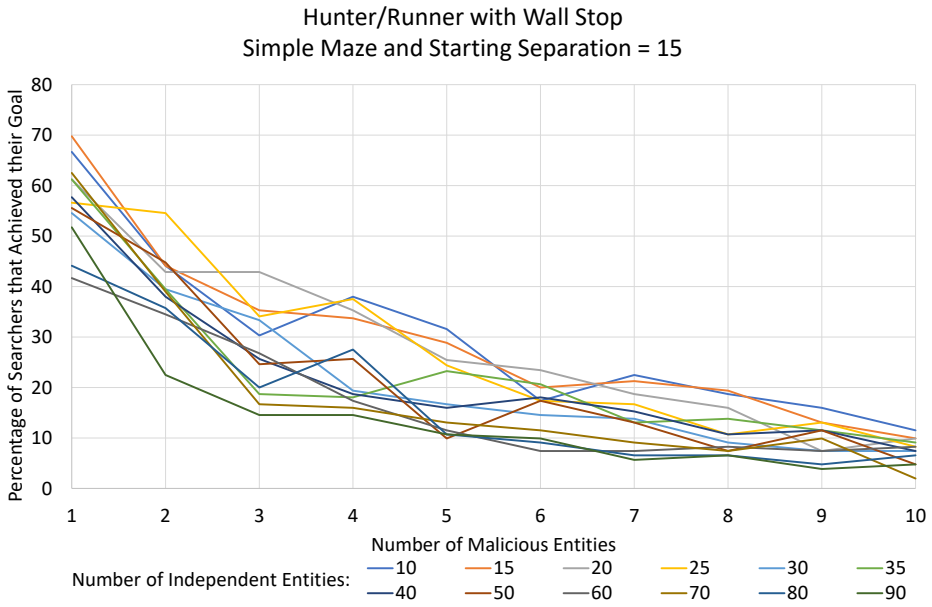


Figure 5.34: Number of Malicious Entities Required to Effect Efficiency of Searchers: Simple Maze

5.2.4 Conclusion of Case 1

It has been demonstrated, by simulations, that it is possible to alter the efficiency of this implementation of a swarm, which is undertaking cooperative navigation techniques to locate a target, by utilising the characteristics of the swarm.

The simulations that I undertook demonstrated that in the case of cooperative navigation, use-case 1, the swarm was able to be successfully manipulated by an attacker.

To assist in summarising the results, averages were taken for the percentage of time a searcher successfully achieved its goal, for each number of malicious entities. This was undertaken across the various operating environments and starting separations. When averaging out the effect of a malicious attacker across each of the number of independent entities within an operating environment, the effects for the different starting separations can be seen in Figures 5.35, 5.36 and 5.37.

What is apparent from the results is that across all of the environments and starting separations, the more malicious attackers that are introduced, the less efficient the searcher is in achieving its goal. Also, the effectiveness of an attack is not directly linked to the amount of malicious entities that are introduced into the operating environment. For example, if the case of a Hunter attack is considered within an environment without a maze, then the attacker might not wish to deploy any more than 4 malicious entities, as the benefit from deploying more could be considered negligible. Similarly, within a Hunter/Runner Wall Stop attack within an environment with a complex maze, the attacker might not wish to deploy more than five malicious entities.

The actual type of attack and amount of entities that an attacker would deploy would depend upon the attackers motivations and capabilities. If an attacker is able to produce their own swarm entities or is able to modify the operating capabilities and characteristics of a swarm entity, then they might wish to undertake a Hunter attack. However, if the attacker utilises the same swarm entities as, say, a searcher, and they can modify the software code on the swarm entity, than they might with to undertake a Hunter/Runner Wall Stop attack.

Another consideration is the operating environment that the swarm, and malicious attack, will be operating within. If the environment is fairly large, with a larger amount of independent entities, then the Hunter/Runner attack might be considered to be more suitable, as once a malicious entity has located a target, it will move away and be assisted with by the independent entities to guide the searcher towards the malicious entities. If the operating area is very large, possibly even unbounded in reality, and free from many obstacles, a malicious entity might never reflect off an obstacle and return towards the target.

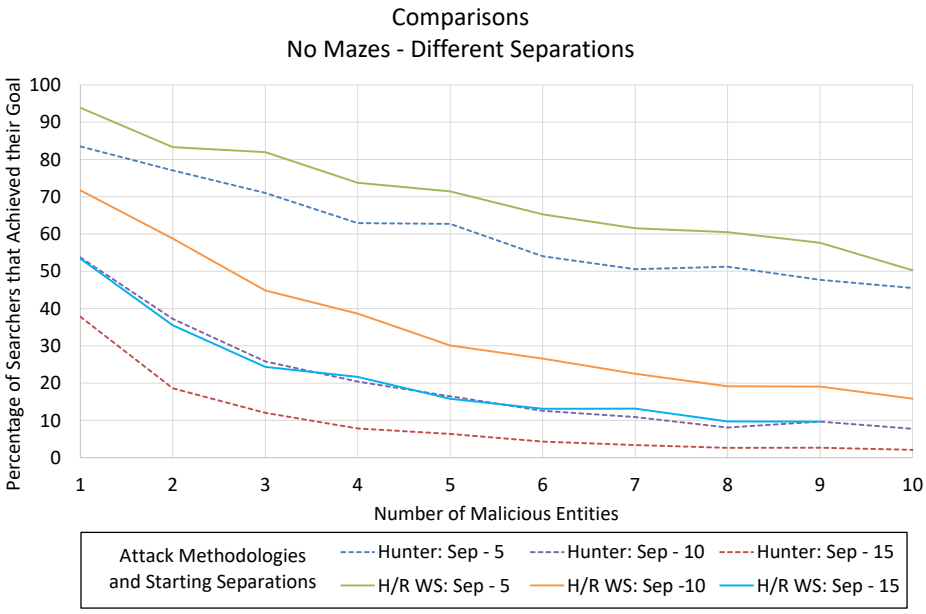


Figure 5.35: Comparing Efficiency of Attack Methods - No Maze

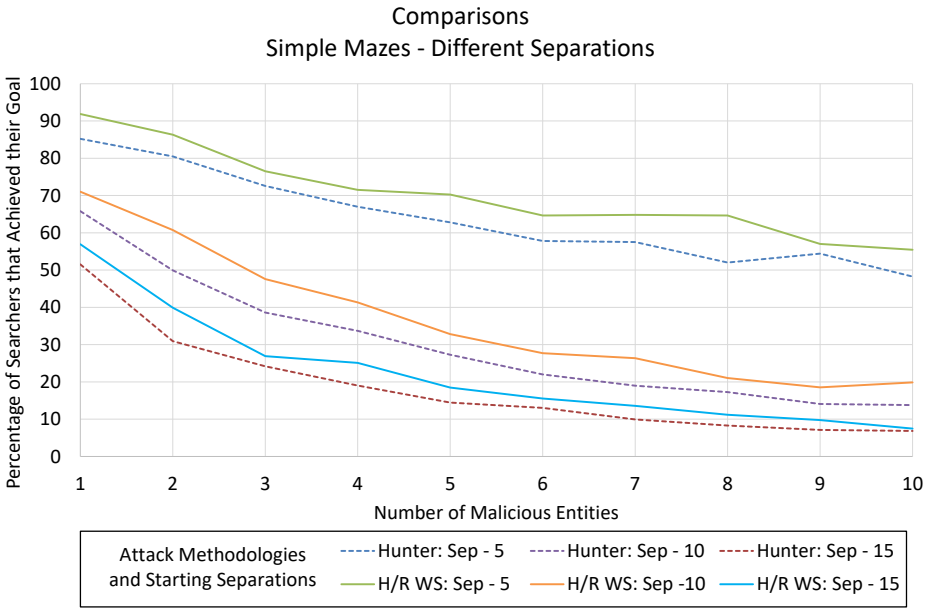


Figure 5.36: Comparing Efficiency of Attack Methods - Simple Maze

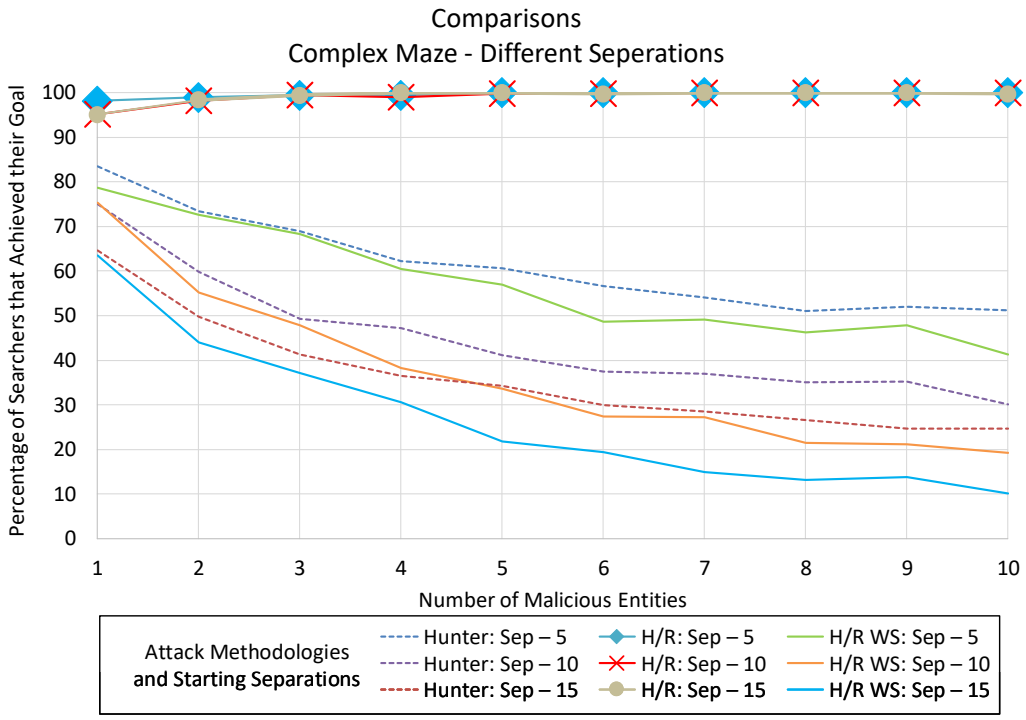


Figure 5.37: Comparing Efficiency of Attack Methods - Complex Maze

5.3 Case 2 - Foraging Techniques and Local Recruitment Schemes

This section describes work undertaken on simulations that demonstrate use-case 2, as the overall goal of the swarm is to make landmines safe by the use of foraging techniques and local recruitment schemes.

The following work was based on research conducted by Kumar, Sahin and Chapman. They proposed the use of swarms to locate and make safe landmines within a defined operational area. Their research was based on how the tree dwelling ants *Polyrhachis Laboriosa* move and forage for food. They then proposed a recruitment mechanism based on the *Novomessor Albisetosis* and *Novomessor Cockerelli* ants, which release pheromones for both short range and long range recruitment schemes, in order to attracting other entities to assist and undertake the required tasking. In their landmine detection research, the research utilised a short range recruitment scheme.

They proposed a foraging technique to locate landmines and then a local recruitment methodology to recruit other swarm entities, in order to assist in the task of making a located landmine safe.

It should be noted that the research conducted by Kumar, Sahin and Chapman was not attempting to provide a practical methodology for locating landmines and then making the landmines safe. They were conducting research on the feasibility of how swarms could be developed that would forage for a target and, upon locating a target, locally recruit other swarm entities to assist with a tasking. They could have considered other scenarios, such as attempting to locate survivors of an avalanche and then recruit other swarm members to provide assistance in casualty recovery.

5.3.1 Taxonomies

The application of swarms utilising foraging and local recruitment schemes can be considered from several perspectives when considering taxonomies, as detailed in Section 2.5.

When considering the *Method Based* taxonomy proposed by Brambing et al. [24], this representation would be considered a *finite state machine* within *Behaviour Based* design methods. Within the *Collective Behaviour* based taxonomy, Brambing et al. would consider this as *Collective Exploration* within *Navigational Behaviours*.

The *Robotic Swarm Behaviour* taxonomy proposed by Cao et al. [31] would classify this activity as *Origins of Cooperation.*

5.3.2 Overview of the Previous Research

The previous research by Kumar, Sahin and Chapman was concerned with a swarm being deployed into a defined operational environment, with the goal of foraging for a target and then locally recruiting other swarm entities to assist in a task. The scenarios that they provided had the goal of locating a landmine and then locally recruiting other swarm entities in order to make the landmine safe.

Swarm entities could move horizontally and vertically within the operating environment and their research stated that a landmine would be classed as disarmed, and removed from the operational environment, if 4 or more swarm entities were located at the same physical location as the landmine. The number of swarm entities required to disarm a landmine is essentially an arbitrary number that they decided upon and the threshold could have been set to any amount.

The previous research proposed that the swarm entity can be in one of three states and can transition between them based on external influences. The three states are:

- Foraging
- Waiting
- Scent Following

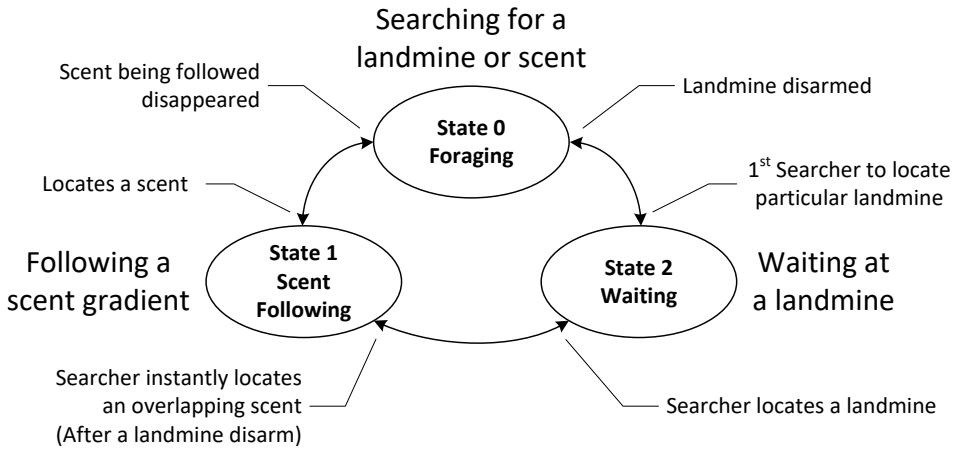


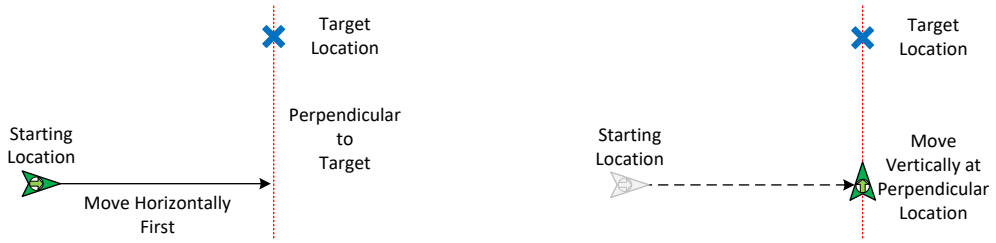
Figure 5.38: Swarm Entity Operating States [112]

5.3.2.1 Foraging

Within the context of Kumar, Sahin and Chapman's research, foraging is the process of the swarm entities moving around the operating environment with the goal of either locating a landmine, that has not previously been located, or the swarm entity moving into an area of the operating environment that contains scent, that has been released by another swarm entity that has successfully located a landmine.

The previous research of Kumar et al. considered different techniques to move swarm entities around an operating environment [114]. Their research suggested that a swarm entity would randomly choose a location within the operating environment and then move to that location, with the movement of the swarm entities being constrained to only move in either horizontal or vertical directions. They suggested that this could be undertaken in one of two methods:

1. **Deterministic Walk** The swarm entity randomly chooses a target location within the operating environment and then moves in a horizontal direction, until it reaches a location that is perpendicular with the target's location. The swarm entity then moves in the appropriate vertical direction, until it reaches the target's location. This can be seen in Figure 5.39.
2. **Random Walk** Again, the swarm entity randomly chooses a target location within the operating environment. The swarm entity then randomly chooses a horizontal or vertical direction to travel along, that is towards the target location, and moves along a distance of one unit of distance. The process is then repeated until either the swarm entity reaches the target's location or the entity reaches a target location limit, where another move in that direction would go beyond the target's location position in a perpendicular direction. If the swarm entity does reach a target location limit, the swarm entity will then only travel in a straight line to the target's location. This method can be seen in Figure 5.40.



(a) Deterministic Horizontal Walk **(b)** Deterministic Vertical Walk

Figure 5.39: Deterministic Foraging Movement

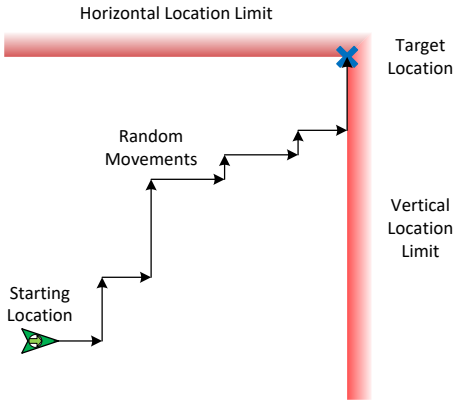


Figure 5.40: Random Foraging Movement

5.3.2.2 Waiting

In Kumar, Sahin and Chapman's research, when a swarm entity locates a landmine, the swarm entity changes state to a waiting state. Within this state, the swarm entity releases a scent and waits at this location. The swarm entity is waiting for three other swarm entities to arrive at its location, that will be locally recruited due the released scent. When this happens, the landmine is classified as disarmed, the landmine is removed from the operational environment and the swarm entities return to either a foraging state or enter a trail following state, if they are within an area of the environment that contains scent from another swarm entity.

The objective of a swarm entity in releasing the scent is to attempt to undertake the local recruitment of other swarm entities, in order to recruit enough swarm entities to that location, in order to disarm the landmine.

The released scent disperses and becomes less dense as the distance increases away from the release point. This causes a scent gradient, that increases the closer the location is to the release point. A cross-section through a scent dispersal can be seen in Figure 5.41 and a 3-Dimentional view can be seen in Figure 5.42. This model assumed that the operating environment had no external influences, such as wind or rain.

Where two or more scents overlap, such as when there are multiple swarm entities at a landmine, the scent density value at a particular location is the addition of all the scent values from the various swarm entities that can contribute to that location. Local scent maximums can also occur when two landmines are discovered that are located close to each other and the scent released from each location is added together, which generates a local scent maximum. An example of this is discussed in Section 5.3.2.3.

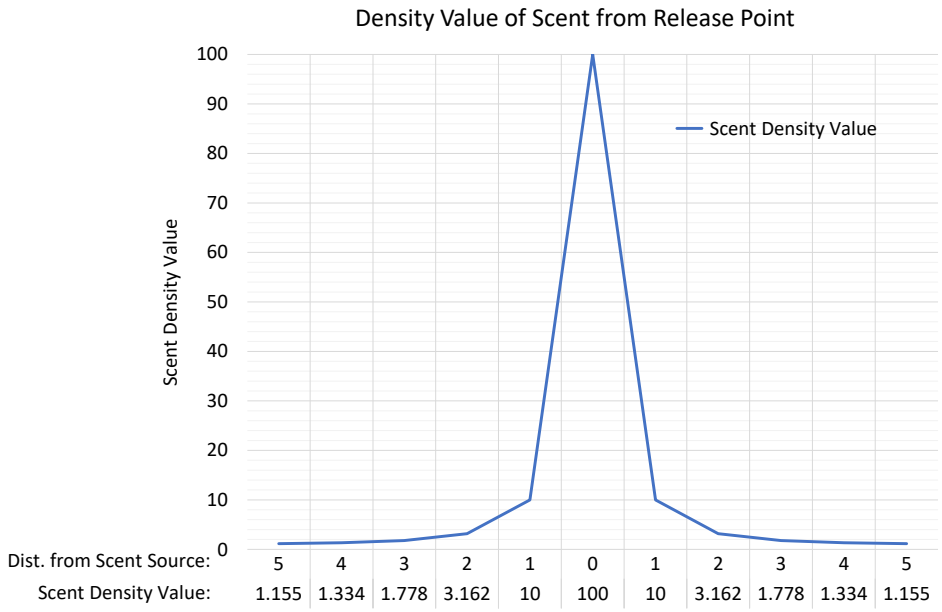


Figure 5.41: Cross-Section Through Scent Gradient

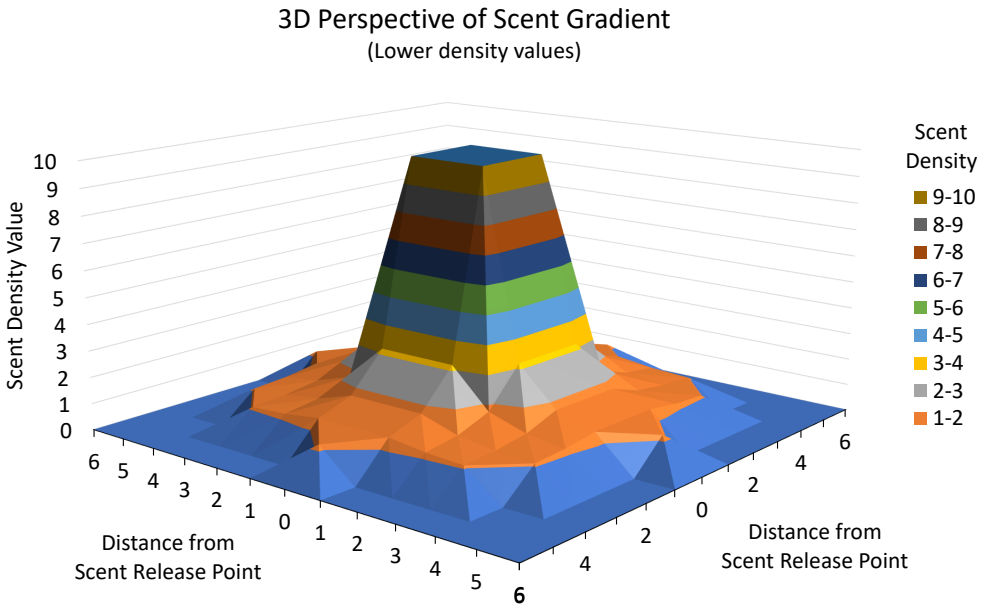


Figure 5.42: 3D Overview of Scent

5.3.2.3 Trail Following

If, whilst moving through the operating environment, a swarm entity moves into an area of the operating environment that contains scent, the swarm entity will change its state to scent following. The swarm entity will then follow the scent, by heading towards an increasing scent gradient. In the scent following function, a swarm entity measures the scent densities directly in front of itself and to both the left and right of itself. The entity will then move in the direction of greatest scent density. In Kumar, Sahin and Chapman's research, if there are equal scent densities, the entity will default to moving forward and if the scent densities to the left and right of the swarm entity are equal and larger than the scent density ahead, the swarm entity will default to moving to the left.

The swarm entity will follow the scent gradient until it reaches either a landmine, where it enters a waiting state, or it reaches the maximum of the scent gradient. It is possible to arrive at the maximum of a scent gradient and not arrive at a landmine. This can happen when two landmines have been discovered and the swarm entities that located the landmines have released their scents. However, the scents cross over each other and where the scents cross, a local scent maximum is produced. A cross-section of this local maximum can been seen in Figure 5.43 and a 3-Dimentional view is shown in Figure 5.44. Kumar, Sahin and Chapman's original research recognised this event and managed this by allowing the swarm entity to wait at the scent maximum until one of the contributing landmines was deactivated from the operating environment. This then removed one of the scents contributing to the local maximum, which removed the local scent maximum and therefore released the previously stuck swarm entity.

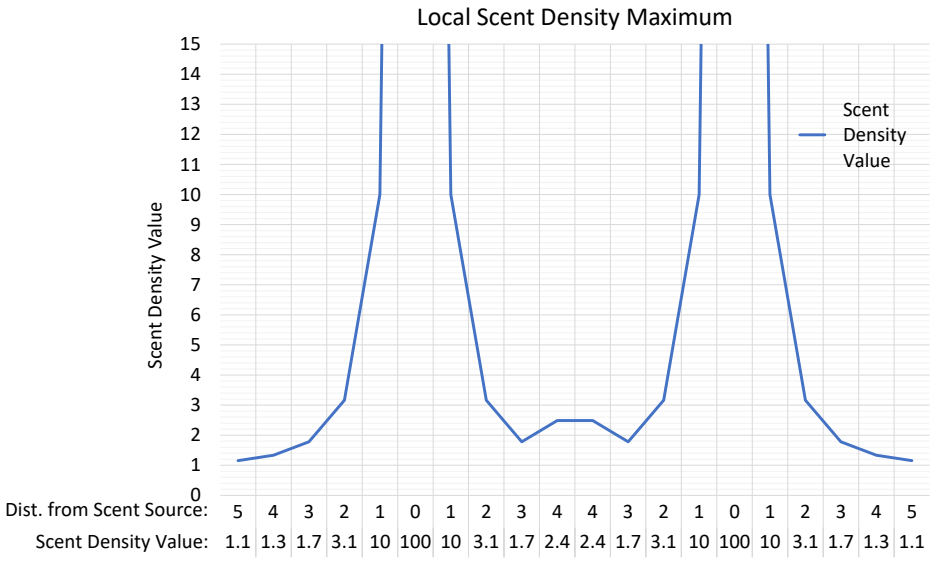


Figure 5.43: Cross-Section Through Local Scent Maximum

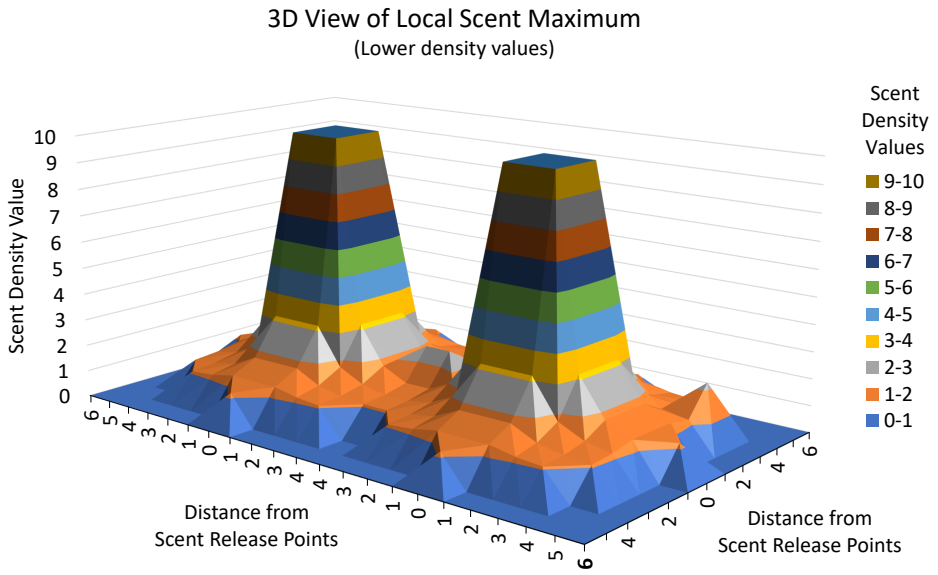


Figure 5.44: 3D Overview of Local Scent Maximum

5.3.3 Simulations

I undertook initial simulations, which were conducted in benign conditions and the results compared to that of the original researchers. This was in order to confirm that the simulations being undertaken were representative of the original research. I then undertook simulations in an attempt to maliciously attack the swarm, which was attempting to locate and disarm the landmines within an operating environment. The objective of the attack was to realise the threat of masquerade by attacking the swarm entities with misinformation, in order to reduce the efficiency of the landmine detection. Essentially, the attacker is attempting to defend the area that contains the landmines. An attack would be deemed successful if there is an increase in the amount of time taken for the swarm to achieve its goal, of clearing all of the landmines within a defined operating environment.

5.3.3.1 Initial Simulations of Previous Research

Initial simulations were conducted in benign environments that repeated the initial research work of Kumar et al. and included both the deterministic walk and random walk methodologies of a swarm entity moving between locations [114].

The operating environment was configured as per the original research by Kumar et al. as an area of 100 x 100 measurement units. A number of landmines were randomly dispersed throughout the operating environment, numbering between 10 and 100 within the simulations, and swarm entities were then released within the operating environment, numbering between 10 and 100. This again followed the research undertaken by of Kumar et al.

To ensure consistency and repeatability, and to make the simulation slightly more realistic, the swarm entities within the new simulations were always released from the same point and from the edge of the operating environment. The chosen location was the lower left position of the operating environment.

The results obtained reflected that of the original research, including the simulations entering what the original researchers called a *frozen* state. A frozen state was realised when all the available swarm entities were in a waiting state, with no other swarm entities available to increase the number of swarm entities at a landmine's location to 4. The number of swarm entities required to make a landmine safe was chosen to be 4, so as to replicate the number required as detailed in the original research be Kumar et al. To place the frozen state into context, assume that there are 10 landmines within the operating environment and 30 swarm entities are deployed, that are attempting to locate and disarm the landmines. It is therefore possible to have 3 swarm entities located at each of the landmines. Therefore, all of the swarm entities will be in a Waiting state, waiting for a fourth swarm entity to arrive, in order to disarm the landmine and release the swarm entities located at that landmine. The swarm entities would therefore wait indefinitely, as no other swarm entities are available to release them.

The results obtained can be seen in Figure 5.45 and Figure 5.46. Both methods were representative of the original research and both methods demonstrated that the average time to locate and disarm all of the landmines was dependent upon the quantity of swarm entities and the amount of landmines present within the operating environment. It was also noted that for both searching methods, when there were only 10 swarm entities, the simulation would time out, as the process was taking too long.

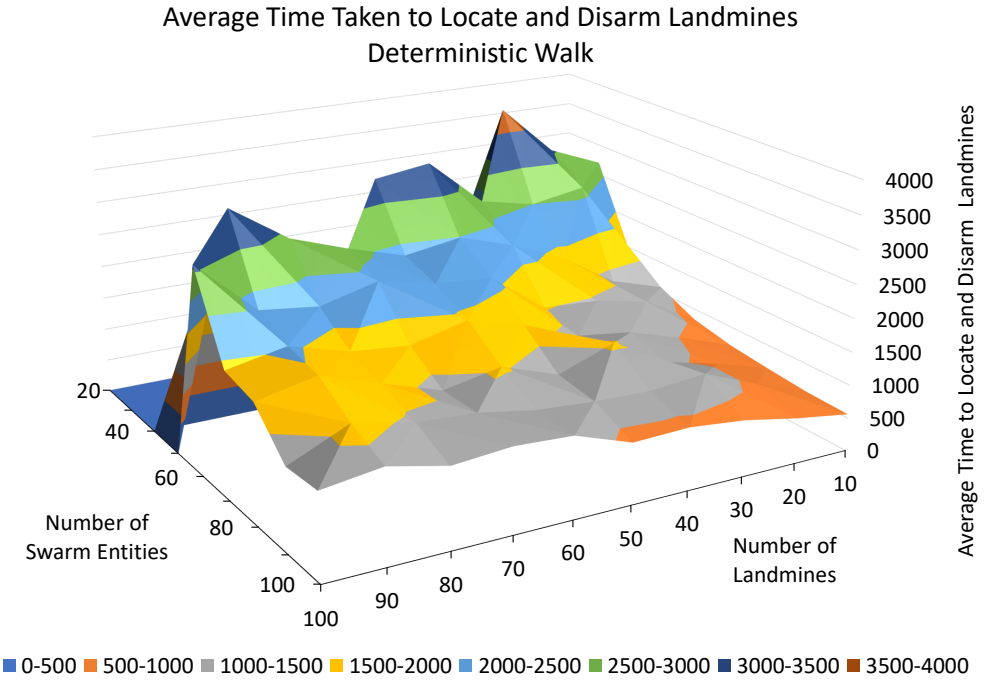


Figure 5.45: Baseline Results of Searching for Landmines: Deterministic Walk

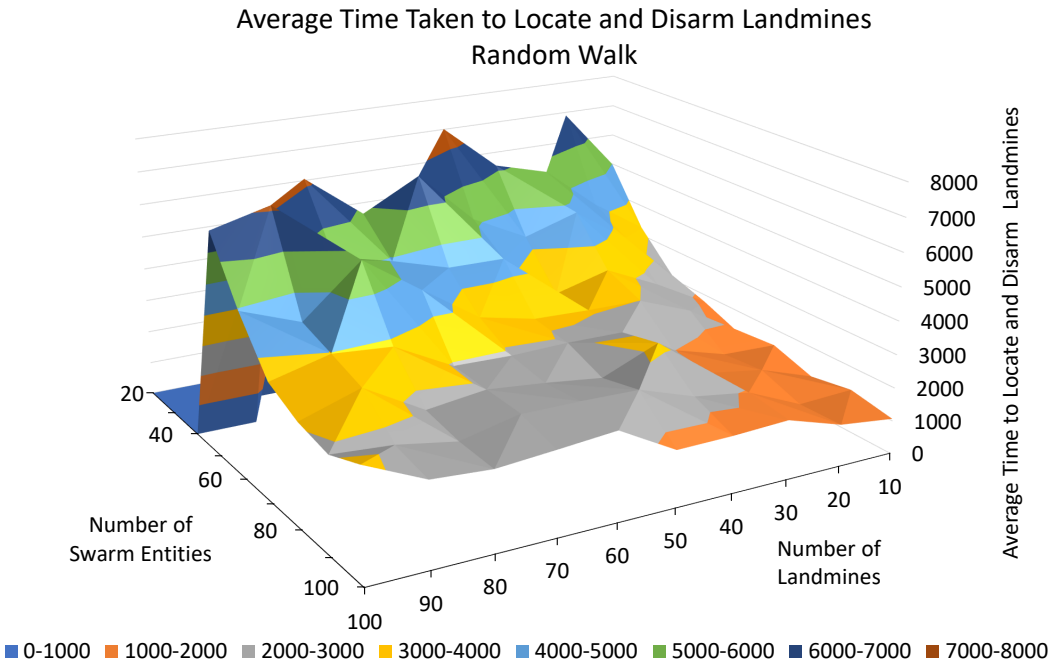


Figure 5.46: Baseline Results of Searching for Landmines: Random Walk

The simulation results demonstrated that utilising the random walk method to search for landmines resulted in a greater number of combinations for the task to be successfully completed. Figure 5.47 and Figure 5.48 show the various combinations of the number of searching robotic entities and number of landmines.

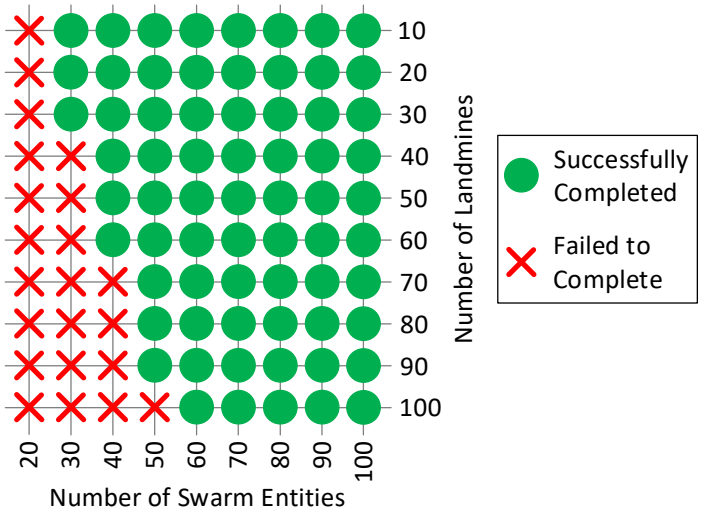


Figure 5.47: Completion of Task to Locate Landmines - Deterministic Walk

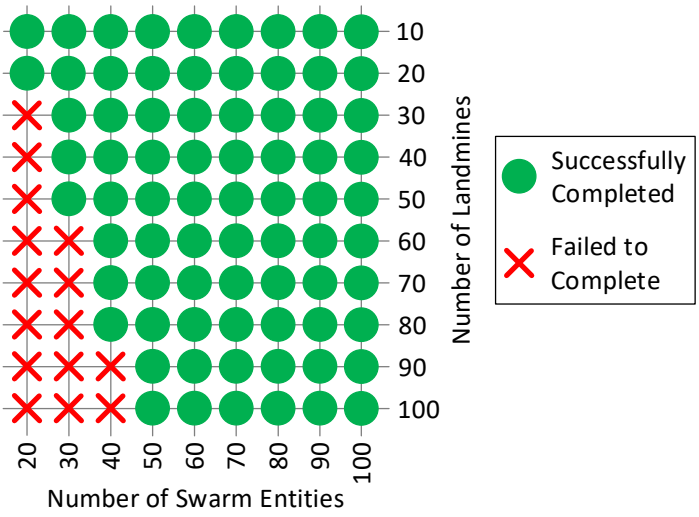


Figure 5.48: Completion of Task to Locate Landmines - Random Walk

Due to the better simulation results that were obtained by using the random walk methodology, further simulations were only conducted that utilised the random walk methodology of moving with the operating environment.

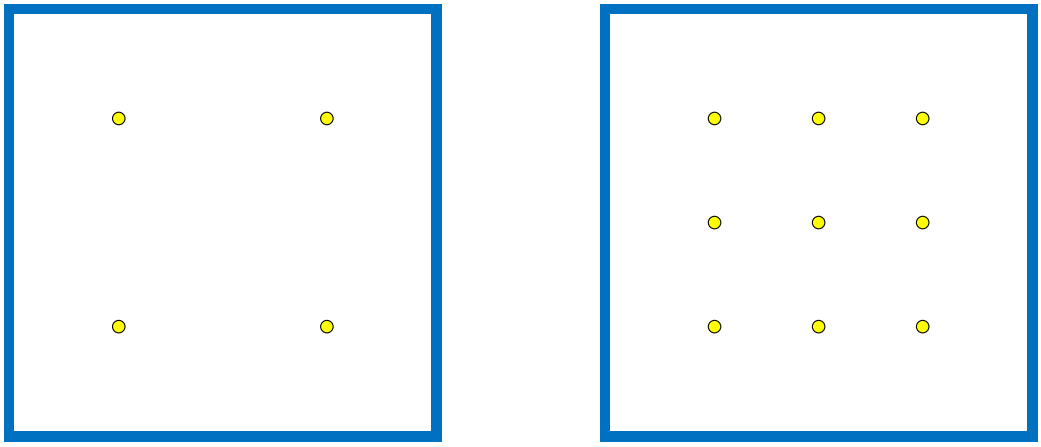
5.3.3.2 Attacking With False Landmines

A swarm entity communicates with other swarm entities by utilising scent within the operating environment, in order to locally recruit other swarm entities to assist in making located landmines safe. An attacker therefore attempts to utilise the knowledge of the communications methodology, in order to realise an attack.

The attacker undertakes the attack by manipulating the operating environment by masquerading as genuine landmines. An assumption is that the attacker understands how a genuine landmine would be detected by a swarm entity and is therefore able to produce a false landmine that would be perceived as a genuine landmine by a swarm entity, realising the threat of masquerade. The goal of the attacker is to modify the operating environment, such that the swarm entity believes that it has located a genuine landmine and then utilises knowledge of the swarm entity's communications methodology against itself. That is, when a swarm entity has located a false landmine, it will attempt to locally recruit further swarm entities, by releasing a scent to communicate the direction of a located landmine. The attacker utilises the communications of a swarm entity to recruit other swarm entities to the false landmine. As the landmine is false, it will never be made safe and therefore the swarm entities will remain in a waiting state and never move from the location.

Initial simulations were undertaken that placed either four (2 x 2) or nine (3 x 3) false landmines within the operating environment, as shown in Figure 5.49a and Figure 5.49b.

The initial simulations were undertaken with limited numbers of false landmines, in order to understand the effect on the swarm and if limited false landmines would result in a successful attack. The effect of this was that the swarm entities would locate a false landmine and act as though it was a genuine landmine. That is, that swarm entity would change state from foraging to waiting and would release its scent, in order to recruit other swarm entities.



(a) Four False Landmines **(b)** Nine False Landmines

Figure 5.49: False Landmine Placements

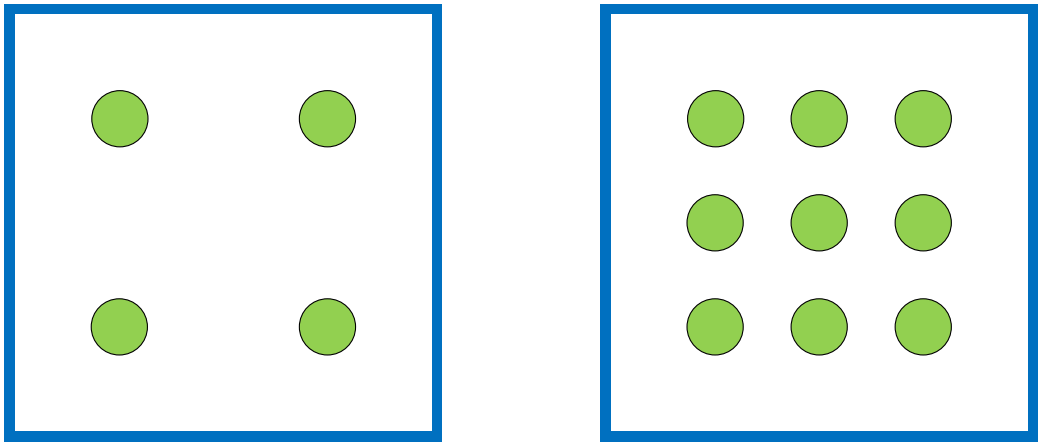
In this attack, the advantage for the attacker is that the attacker does not require an extra resources, other than the false landmines. The potentially limited resources of the production of scent is carried out by the swarm entity itself. As the attack continues, further swarm entities are recruited by the scent. However, as the false landmine will not be removed from the environment by 4, or more, swarm entities being located at the physical location, the attack continues and further swarm entities can continue to be recruited and essentially become "trapped" in the waiting state. This is similar in context to the original research, when the swarm entities could enter a frozen state.

The simulations showed that the effect of placing the false landmines within the operating environment was that the malicious false landmines prevented the original swarm from achieving its goal, of locating and making safe all the landmines within the operating environment. That is, although some of the genuine landmines, which had been randomly placed within the operating environment, would be found, eventually all the swarm entities would usually either locate or scent follow to a false landmine. The swarm entities would then remain at the false landmines indefinitely. This could be considered analogous to a Denial of Service against the swarm, or could even be considered a "Denial of Goal" attack, as the original swarm members are still attempting to perform their task but are prevented from completing this by the malicious activities within the operating environment.

5.3.3.3 Attacking With False Scent

A second type of attack considered the use of a false scent, in order to attract the swarm entities.

Scent generators were positioned within the operating environment, in the same positions as the false landmines in Section 5.3.3.2, and they released a false scent. The false scent attack allowed for more of the operating environment to be initially covered by the attacker. That is, within this attack there is no requirement for a swarm entity to locate a false landmine, before a scent is released. The scent is always present within the operating environment. This can be seen in Figure 5.50. When there were four malicious scent producers, 3.24% of the operating environment was subject to malicious attack. When there were nine malicious scent generators, the area covered increased to 7.29% of the operational environment. All a swarm entity had to do was come into contact with any of the false scent and it was essentially caught by the trap and would remain there indefinitely.



(a) Four False Scent Generators **(b)** Nine False Scent Generators

Figure 5.50: False Scent Locations

The false scent released had the same properties as the genuine scent released by a swarm entity, so that a swarm entity would therefore interpret the false scent as thought it had been released by a genuine swarm entity. The assumption is that the attacker is aware of the properties of the scent released by a swarm entity and the attacker is able to successfully replicate the scent.

The advantage of this attack to an attacker is that the attacker does not have to replicate a landmine, with what could be difficult properties to replicate. The attacker is also able to locate the scent dispensers in what the attacker believes to be the most efficient positions, in order to defend their landmines. The disadvantage of this attack is that the attacker has to release the scent, which would be a finite resource. However, as the attacker essentially controls the operational area, the attacker could have access to large quantities of the false scent.

The simulations were undertaken, assuming a limitless supply of the false scent.

The simulations demonstrated that the attacks were successful. The swarm entities that were in a foraging state would alter to a trail following state and would then follow the scent gradient to where the scent density was at a maximum. As there was no landmine at this location, the swarm entity would remain in a trail following state but would remain around the location of the scent maximum.

In this attack, a swarm entity effectively moves around the point of the maximum scent density, this is shown in Figure 5.51. Say a swarm entity has followed the scent gradient from the left of the maximum, from 4 to 5. A swarm entity would normally expect either a landmine at this location, or an increase in the scent gradient at another immediate location. However, as neither of these two conditions are met, the swarm entity defaults to moving forwards, from 5 to 6. As the scent is dispersing and reducing in density, the largest scent densities are now at 3 and 9. The swarm entity therefore defaults to the left for equal scent density amounts and moves to 3. This process continues and will go on indefinitely.

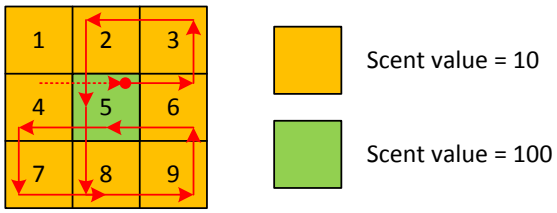


Figure 5.51: Trapped Searcher

As with the false landmines attack simulations, the swarm deployed within a false scent attack simulations did locate a few genuine landmines. However, the swarm quickly entered a position where all the swarm entities were either waiting at genuine landmines for other swarm entities to join them, or, more often the case, the swarm entities had followed the false scent trail to a scent maximum and the swarm entities were moving around that location.

This attack did result in swarm entities switching to a waiting state at genuine landmines, waiting for other swarm entities to join them in order to make the landmine safe, and could be considered similar to the swarm entities entering a frozen state. However, the difference between this and a frozen state is that in a genuine frozen state, all the swarm entities are in a waiting state. In this attack, only the swarm entities at genuine landmines are in a waiting state, the swarm entities attracted by the false scent are in a scent following state, with the remaining swarm entities in a foraging state.

Again, as with the false landmines, this would then run indefinitely and the swarm had effectively been subject to a Denial of Service, or Denial of Goal, as the swarm was no longer able to locate and make safe any remaining landmines.

This attack could have been modified with the addition of false landmines, as detailed in Section 5.3.3.2, such that the false scent attracted the swarm entities to the false landmines. The attacker could then modify the scent producers so that they stop producing scent if a swarm entity is located at its position, so as to preserve the attackers scent resources by utilising the scent produced by the swarm entity. However, the results obtained would be the same as those obtained by utilising a false scent attack and was therefore not considered.

5.3.4 Conclusion of Case 2

The simulations I conducted were able to show that the original research by Kumar, Sahin and Chapman did propose a viable methodology, that utilised foraging techniques and local recruitment schemes. This was demonstrated by simulating their proposed scenario of locating landmines and recruiting other swarm entities, in order to make the landmine safe. The results I obtained successfully replicated the results of the previous research by Kumar, Sahin and Chapman.

Further simulations I undertook were able to show that simple attacks were able to be developed. These attacks utilised the knowledge of how the swarm operated and interacted with the local environment and how this was able to be used against the swarm entities. This allowed the attacker to defend the landmines within the operating environment from the swarm entities attempting to make the landmines safe. The successfully attacks were able to maliciously recruit the swarm entities and thus prevented the swarm from achieving its goal, resulting in the swarm being subject to Denial of Service.

The simulations that I undertook demonstrated that use-case 2, where a swarm utilises foraging and local recruitment techniques in order to locate and make safe landmines, can be subject to a successful attacks which prevent the swarm from achieving its goal.

Again, it is worth noting that this research, as well as the research of Kumar, Sahin and Chapman, was not concerned with the technology and methods required in order to detect and disarm actual landmines, it was concerned with proposing and demonstrating potential attacks that could be undertaken against swarms that utilise foraging techniques and local recruitment schemes in order to achieve a goal.

Indeed, the actual methods and techniques that could be utilised to locate landmines, let alone disarm them, are varied, such as the use of ground penetrating radar or attempting to sense magnetic anomalies within the local environment. The detection methods used could be dependent upon the landmines that are believed to be within the operating environment, such as metal cased landmines or composite landmines and anti-personnel mines or anti-tank mines.

Another challenge could be why false landmines would be used and not just use genuine landmines, as genuine landmines are a relatively inexpensive resource that can be used for area denial. A reason why a force might wish to utilise false landmines is that landmines are often located within an area to either protect an asset or deter people from approaching or entering an area, they are essentially attempting to deny an area.

However, if the party that laid the landmines wishes to have a safe route through the minefield, it would be beneficial for them to have a passage that they know is safe but others would not. Therefore, false landmines would offer a known safe passage to the party that laid the minefield but would appear to be genuine landmines to others, which would impede the progress of others through the minefield. A similar technique is currently in use. If there is a military force operating in a country where its is known that there are minefields, the military force could use *mine tap* or landmine signs as a psychological deterrent in order to protect an area. Mine tape is a physical warning tape that indicates the presence of landmines within the taped area, similar in concept to a police cordon tape. Although the military force has not deployed landmines, they have used mine tape to indicate the presence of landmines, even though there are no landmines, deterring people from entering that area. However, if the military force that placed the mine tape needs to cross the area that is taped off, such as in an emergency escape or countering an enemy, it knows that it can do this safely.

Another reason for using false landmines is that the operator of the swarm, and any troops attempting to cross a minefield, would loose confidence in the abilities of the swarm. The operators and troops would observe more than the required amount of swarm entities permanently located either at or around a position and therefore not making a landmine safe. The troops would then be required to use other methods to traverse the minefield, which could slow them down.

5.4 Discussion

This chapter has provided an overview of different types of swarm implementations that was based on the two swarm use-case studies. These were based on the taxonomies from Section 2.5, with different applications and goals, which was based on literature reviews. These swarm proposals were simulated within benign environments, in order to confirm the validity of the simulations when compared to the original research. Once the simulation results were validated against the original research, the simulations were then modified, in order to undertake attacks against the swarm implementations. The attacks were tailored for the individual swarm implementations, their respective operating characteristics and the operating environment in which the swarm was released.

Several attacks were undertaken against the various swarm implementations, which utilised the characteristics of the swarms to the attackers advantage.

The simulations demonstrated that the swarm implementations were able to be maliciously manipulated. This can be observed from the results, as the attacks undertaken against the swarm entities were successful. This was demonstrated by either preventing the swarm from fulfilling its goal or reducing the efficiency of the swarm in realising its goal.

This work was then used to research the possibilities of defending the swarms from attack, by the use of Intrusion Detection System techniques, and is detailed in Chapter 6.

6 Modelling Intrusion Detection Systems in Swarms

6.1 Introduction

Following on from the swarm simulations in Chapter 5, which demonstrated that a swarm could be adversely affected by a malicious swarm, a review was undertaken to understand if Intrusion Detection Systems (IDS) techniques could be utilised to protect a swarm. The chapter further updates the swarm models from Chapter 3, in order to take IDS into account.

The aim of this chapter is to provide the reader with an overview of IDS and the current techniques that are utilised within existing technologies and systems. The chapter will provide the background of the subject and will then place the IDS techniques within the context of swarms. The chapter then concludes with a discussion regarding the applicability of using IDS techniques within a swarm.

6.1.1 Overview and Background of Intrusion Detection Systems

To assist the reader, NIST defines intrusion detection as "the process of monitoring the events occurring in a computer system or network and analysing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices" [168] and intrusion has been defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [84].

Intrusion detection systems attempt to automatically detect an activity that should not be taking place and then report upon this event. Network domain and computer system IDS operate as either extra devices within a network, or software applications within system components, that are attempting to detect intrusions within these networks and systems. If the IDS suspects anomalous behaviour, the actions undertaken by the network and computer based systems are essentially the same. If the IDS suspects it is being infiltrated, it will log the event and report it to the relevant parties. If the IDS is local to a system, say a PC, then it might also alarm locally, such as by displaying a warning to a user.

The objectives of an IDS will influence the techniques that are used, in order to undertake the IDS task. That is, if the purpose of the IDS is to detect intrusions within a network, the network designer might decide to place the IDS as a separate component on the network or at network boundaries, separate from the operational systems. However, it might be advantageous to place the IDS within individual systems, such as computer end-point hosts, as the end-point IDS are able to detect attacks which an IDS placed on the network would not. For example, the network traffic might be encrypted and therefore prevents the network IDS from monitoring the traffic's content. However, if the traffic is decrypted at an end-point, in order to be used, the traffic can also be accessed by the IDS.

The goal of an IDS is to identify possible incidents, by automating the monitoring of computer systems. An IDS can also be utilised to not only detect an incident but to also detect reconnaissance, which could indicate that an attack is immanent [168].

The main detection methods used by IDS are:

- **Signature Based** Signature based IDS, sometimes referred to as misuse detection, is the simplest detection method and utilises signatures to identify potential incidents. A *signature* is a pre-determined pattern that corresponds to a known threat. The IDS compares data to a library of known threat signatures, in order to identify possible incidents [148, 168].
- **Anomaly Based** In an anomaly based IDS, behaviour models for the normal system characteristics are created, which are known as *profiles*. These normal characteristics are then compared to the actual system characteristics, looking for deviations, within thresholds, from normal behaviour [77, 168].
- **Stateful Protocol Analysis** Stateful protocol analysis IDS use predetermined *profile definitions* of benign protocol activity, sometimes referred to as *stateful signatures*, which characterise the behaviours of each protocol state. These are then compared against observed events to identify anomalies and possible malicious behaviour [2, 168, 206].

It should be noted that an IDS has the potential to either incorrectly identify a benign activity as hostile, known as a *false positive*, or it fails to recognise a malicious activity, known as a *false negative*.

An IDS is often *tuned* in order to reduce false positives, with the effect of increasing false negatives.

A more detailed overview of Intrusion Detection Systems and their history is provided in Annex A.

6.1.2 Systems Similar to Swarms

There has been significant research undertaken of the implementation of IDS within systems that exhibit similar characteristics to swarms, such as mobile ad-hoc networks (MANET) and wireless sensor networks (WSN).

Considering these two examples, both MANETs and wireless sensor networks can have constrained resource characteristics, these typically include: low computational power, limited memory capabilities, limited stored energy, they can operate in a decentralised fashion and they do not always communicate to a concentration point or central controlling authority, which could be used to aid in analysis of data in order to attempt to detect an attack [36, 83, 138, 208]. The properties of MANETs and WSNs have similarities with swarms, so as a first step in investigating swarm IDS, it is useful to consider whether IDS used in MANETs and WSNs can be used or adapted for swarms.

The current research for these similar systems relates to analysing the traffic that is communicated via the various systems and then utilises the standard IDS methodologies, either signature based, anomaly based or stateful protocol analysis. These resource constrained systems utilise the same techniques as systems that are not generally resource constrained, such as fixed infrastructure networks.

The IDS would need to be tailored for the system, its environment and the expected operating characteristics of the system. If a MANET or vehicular ad-hoc network (VANET) had a node disappear from the network, the network would continue to operate and this would not be seen as unusual by an IDS, as the mobile node might have moved out of communications range or it might have been powered down, such as turning a car engine off. This would not be interpreted as an attack by an IDS, as the IDS would not consider this observation as uncharacteristic. From a network perspective, this would be considered as normal operation, conditions which the network has been designed to operate within.

However, swarm implementations typically have other characteristics, that these systems do not possess. The most significant being swarm characteristic 4, in that swarms exhibit an emergent behaviour, based on the swarm's non-deterministic behaviour and external influences. The swarm's emergent behaviour will be determined by the swarm's goal and the interactions and stimulus between the individual swarm entities, that constitute the swarm, and the environment in which the swarm is operating within. Based on these external stimuli, a swarm entity will determine which actions it will undertake and, from these actions, an emergent behaviour will emerge for the swarm as a whole. Unlike MANETs and VANETS, which will not exhibit an emergent behaviour and where the node will move wherever the host vehicle takes it, swarm entities movements will be determined by the various influences and stimuli.

Therefore, the IDS within a MANET or VANET will only be observing the traffic that the system is communicating. These messages might be regarding the node itself, such as battery life or failure modes, but it will not have an influence on its host. An IDS for a swarm will need to be required to detect potential attacks against swarm entities, which will be able to influence the overall swarm's emergent behaviour. To place this into context, suppose that a search and rescue swarm has the goal of detecting a target, such as in use-case 1, and then directing a searcher onto the target.

If a malicious attacker presented the swarm entities with a correctly formatted message with an incorrect location for a located target, the swarm would react as if this were a genuine message and the emergent behaviour would adapt accordingly, such as stop searching for a target and change state into directing searchers towards the target. This would be because an IDS would not detect any malicious or suspicious payload or patterns in the message, there is nothing anomalous about the message and the protocols are correct.

6.2 Considering IDS for Swarms

As discussed in Section 1.2, a swarm entity's behaviours and actions, in order to achieve its goal, are initially based upon pre-deployment information, such as a swarm entity's goal, and once deployed they are based upon received information and stimuli. The information and stimuli could be from received information from other swarm entities and the environment in which it is operating.

It should be noted that there is an assumption that a swarm's entities will act appropriately to external influences, such as received communications and the operating environment.

Therefore, knowing how a swarm entity acts in normal conditions and that it will act appropriately to inputs and stimuli, this presents the question of whether behaviours that appear to be unexpected or uncharacteristic could be detected?

Actions of swarm entities that are observed in isolation might not be considered to be due to malicious activity. However, when the actions of the swarm entities are considered in the context of interactions between other swarm entities and the operating environment, then actions that would previously be considered innocuous might be considered to be unusual behaviour. However, it might be difficult to know, in advance, what actually constitutes unusual or malicious behaviour. This is because the swarm will exhibit different emergent behaviours, based on the various stimuli.

The detection of unusual behaviour could be used as an indicator, to either suspect, or detect, malicious activity. Therefore, based on the possible IDS methodologies, as detailed in Annex A the principles of signature based and anomaly based IDS are considered within the context of a swarm.

There is also a discussion on the consideration of utilising a separate IDS Swarm, which is independent of the original swarm. The goal of the IDS swarm being to detect the presence of malicious activity, removing the detection overhead from the original swarm. The IDS techniques will also be modelled with respect to the previously described generic swarm model, as presented in Chapter 3.

6.2.1 Background Assumptions for a Swarm Entity

This research assumes that the swarm entities are to be considered as relatively simple, low power and resource constrained devices. It is appreciated that as technology develops, then capabilities do increase. However, the potential uses of swarms and the aspirations to develop swarms that will undertake tasks that require very small sized entities, such as undertaking maintenance in jet engines, will lead to the development of smaller and lower power entities. There is currently research considering the medical use of swarms that are researching the utilisation of nanoscale technologies, as described in Section 2.8.

As discussed in Section 2.3, there are a range of swarm entities available with various capabilities, such as processing powers ranging from 8bit 8MHz processors through to 32bit 168Mhz processors, as detailed in Annex B. Along with the ability to add extra modules to certain entities, such as the e-puck2 robot, it could be possible to implement security functions on the individual robots, such as cryptography functions and authentication mechanisms.

However, this research is interested in the cases where the robot entities have insufficient processing power to undertake security functions that require higher power and potentially resource intensive processes or operations, such as authentication mechanisms or encryption of the traffic capabilities, and potential attacks against the swarm entities need to be countered by other methods.

As previously noted in Section 6.2, the communications between the various swarm entities are perceived to be valid by the receiving swarm entity, if the received messages are in the correct format and within pre-defined ranges. The receiving swarm entity would then act accordingly upon the received message. To place this into context, a swarm entity expects to receive a message that contains a bearing value between 0 and 359. That is, the swarm entity expects to receive a numeric value and the value to be within a bounded range of permitted values. Anything outside of these bounds would be ignored, as, in normal operating conditions, the system might assume an error on the communications channel. Therefore, a received message that contained a bearing value of between 0 and 359, at the correct position within a message, would be treated as a valid message and the swarm entity would act accordingly upon this information, whether the message was genuine in its source, or not.

6.2.2 Intrusion Detection Methodologies within a Swarm

The proposal within this research is that if the normal actions of a particular swarm could be characterised, then abnormal swarm behaviours could therefore be detected.

As detailed within Section 6.2.1, legitimate messages received by a swarm entity would be acted upon. However, although the swarm entity might be receiving messages that meet the requirements and constraints of legitimate messages, the source of the messages could be a malicious attacker.

The goal of the attacker is to influence the original swarm entities, such that the effect of the messages that they send prevents the swarm from achieving its goal, or degrades the efficiency of the swarm in reaching its goal. In doing so, it is proposed that the messages sent from an attacker, although meeting the requirements of a genuine message, could affect the behaviour of the swarm. Because of the malicious messages, the swarm entities actions could then exhibit abnormal behaviours.

6.2.2.1 Signature Based Intrusion Detection within a Swarm

Within a signature based IDS, the IDS compares signatures of pre-determined patterns of known threats with system activities. Considering the principles of signature based IDS, as described in Annex A.1.4.1, signatures of abnormal swarm behaviours could be loaded into a signature based IDS mechanism.

If the IDS activity was going to be undertaken by either the swarm itself, or by a separate IDS swarm, then, due to a swarm's characteristics of autonomy and decentralised control, any IDS signatures would need to be loaded prior to the swarm's deployment.

The IDS signatures would be dependent up the individual swarm implementations and its normal operating characteristics.

The potential issues of a signature based IDS within a swarm are that the signatures might not only be dependent upon a particular swarm's implementation, the IDS signatures might also need to take into consideration the swarms operating environment. There is also the issue that due to a swarm's characteristic of autonomy and decentralised control, the IDS signatures would not be able to be maintained or updated once the swarm has been deployed. Swarm entities would need to be recovered to enable patching and updating of IDS signatures.

The period of time between updating the IDS signatures would depend upon an attackers sophistication. Essentially, once it is suspected that an attacker has understood the IDS mechanism, has introduced an attack that overcomes the IDS and the attack has been discovered, then the IDS mechanisms will require updating. As with any IDS implementation, there is a finite period of protection offered by the IDS before updates are required.

Management of swarm deployments would then become essential, as all new deployments and re-deployments would need to have up to date IDS signatures.

However, a signature based IDS, based on a swarm's implementation, is a viable option.

6.2.2.2 Anomaly Detection within a Swarm

Anomaly based IDS profiles are created that model the normal system characteristic of a system. The profiles are then compared to the actual system characteristics, in order to look for deviations from expected normal behaviours. The principles of anomaly based IDS are discussed in Annex A.1.4.2.

The anomaly detection requires a profile to be created, that characterises a baseline of normal behaviour for a system. The profile is created over a period of time, referred to as a training period, and the profile can be either fixed, in that the baseline for normal system behaviour does not change, or dynamic, where the baseline is modified over time in order to refine the profile based on further observations.

This presents several issues within a swarm environment. The normal operating characteristics of a swarm within a real world, non-laboratory, environment will be different on each swarm deployment. Swarm entities will be under the influence of differing environmental effects and factors, even if deployed to the same operating area. That is, to position all of the individual swarm entities with exactly the same initial operating conditions, such as specific locations and directions, would be practically impossible. To exacerbate the issue, the environmental conditions and subsequent effects could change over time. Therefore, as the initial operating conditions and environmental effects will never be the same, the baseline for normal behaviours will also be different and therefore very difficult to profile, other than to provide a course guess of the potential conditions. This will be further exacerbated by locating the swarm to different initial operating environments and subsequently differing environmental events.

The differences for an anomaly based IDS for a swarm and that of a network IDS is that the network IDS can be baselined, in order to generate profiles. If the network is physically altered, such as by adding new end points to the network, or its utilisation is modified, such as by altering the time that traffic is expected across the network for backups or users that alter their working patterns, the IDS profiles can be re-baselined. This can be the case for either a static implementation, where the entire baseline would be updated, or dynamically, where the IDS might be able to manage the changes itself. Reports from an anomaly based network IDS often need interpretation by analysts, in order to try and minimise false positives and modify profiles in order to maintain an accurate baseline profile. As the swarm is autonomous, once released, and only has localised communications, it would not be feasible for the reports to be collected and processed.

As a swarm's initial and emergent behaviours will differ, based upon the swarm's operating environment and goal, a swarm's actions cannot be predicted and therefore cannot be baselined into a profile. This is further exacerbated, as swarms are often designed with randomness built in, to assist in reducing the effects from failures and improve the performance of the swarm.

A swarm's characteristics, as detailed in Section 2.4, also make characterising normal behaviour even more difficult. Examples of this are:

Autonomy swarm characteristic 1. Once deployed a swarm will be operating autonomously. Therefore, any anomaly analysis techniques will have to be undertaken on the resources provided by the individual swarm entities.

Decentralised control swarm characteristic 2. As there is no overall control authority that could assist with anomaly detection. All anomaly detection will have to be undertaken by the individual swarm entities, as external resources cannot be utilised in order to provide anomaly detection support to the swarm.

Local sensing swarm characteristic 5. An individual swarm entity can only observe the behaviours of other swarm entities within its immediate environment. A swarm entity is therefore, individually, unable to gain a full understanding of the overall behaviour of the swarm.

Local communications swarm characteristic 6. Individual swarm entities are only be able to communicate to other swarm entities within a local area. Therefore, any observations of the perceived characteristics, for either normal or training behaviours, and the associated profiles would only be communicated within the area local to the swarm entity. As different swarm entities communicate this information, the overall information will become increasingly difficult to manage and maintain.

Scalable in size swarm characteristic 8. As a swarm is able to reduce or increase in size, the communication of up to date and accurate information regarding profile information would become more difficult to manage and maintain. Swarm entities that are introduced to the overall swarm would not have an accurate representation of the baseline profile for normal behavioural characteristics.

In summary, a swarm is a large number of relatively basic robots, which have characteristics that are particular to swarms. Swarm implementations can included randomness and that each swarm deployment in the real world will effectively have slightly different environmental properties and characteristics. Therefore, anomaly detection as a means of IDS within a swarm is not viable.

6.2.2.3 Separate Intrusion Detection System Swarm

In addition to considering the application of existing IDS, we may also consider the use of swarm technology itself to provide the IDS. The use of a separate IDS swarm is considered, that is independent of the original swarm. The objective of the IDS swarm is to observe behaviours of the original swarm, with the goal of detecting uncharacteristic behaviours within the original swarm, which could be caused by malicious behaviours, and is undertaken without affecting the performance of the original swarm.

The rationale for a separate IDS swarm is that the IDS swarm could be tailored specifically to detect malicious activity, regardless of the original swarm's goal.

The IDS swarm would need to be aware of and operate alongside the original swarm, without directly interfering or influencing the original swarm from achieving its goal. As an example, in order to provide the IDS service to the original swarm, the IDS swarm will have to observe the original swarm, whilst also carrying out its on undertakings. To place this into context, if the original swarm moved location, the IDS swarm would be required to follow the original swarm, in order to maintain its observation of the original swarm. The IDS swarm would also have to react according to other external influences, such as stimulus from the local operating environment.

This principle can be considered as being similar to Reynold's rules [153], as described in Section 2.2.3 and shown in Figure 2.1. That is:

- **Separation rule** IDS swarm entities will not collide with the original swarm entities.
- **Alignment rule** IDS swarm entities will align its direction of travel with the mean direction of travel of the original swarm entities.
- **Cohesion rule** In this instance the swarm entities will attempt to move towards the mean position of other swarm entities that are nearby, so the IDS swarm entities will attempt to move towards the original swarm entities.

The number of members of the IDS swarm should be smaller than that of the original swarm but could benefit from increased resources, such as processing capability and communications ranges. That is, the IDS swarm does not necessarily need to adhere to the design constraints of the original swarm, as they are only observing the original swarm, and could observe larger operating areas, when compared to the original swarm. This principle is shown in Figure 6.1, where the original swarm's detection range is shown by the smaller yellow circles and the IDS swarm's detection range is shown by the larger green circles. Therefore, less IDS swarm entities are required to cover the operating environment.

Similarly, the physical implementation and construction of an IDS swarm does not have to be the same as the original swarm. The IDS swarm just has to be able to monitor the actions and behaviours of the original swarm and therefore the IDS swarm can be of a different construction, such as physical size, computational capabilities, power resources, sensor fits and detection capabilities.

In an ideal case, the IDS swarm will not need any additional resources and these capabilities could be built into the original swarm.

6.3 Modelling Swarm IDS Techniques

In Section 6.2 we proposed a novel technique, where swarms provide the IDS. The process of attempting to identify an intruder within a swarm could be undertaken either by the original swarm entities themselves, or by a separate IDS swarm.

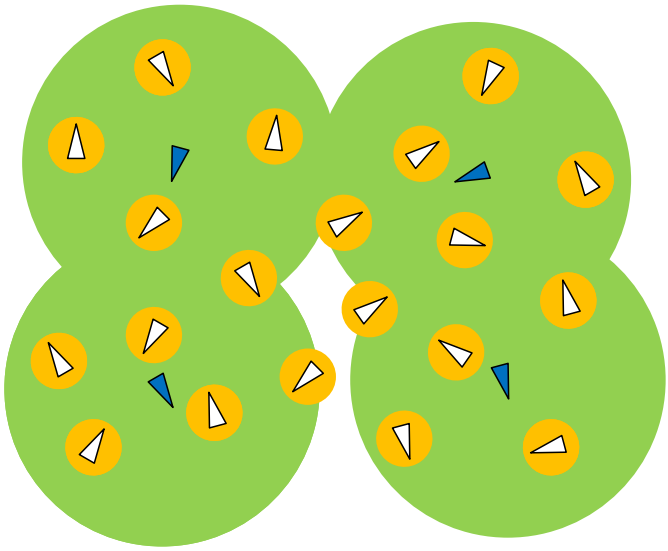


Figure 6.1: IDS and Original Swarms

A generic model was proposed for swarms in Chapter 3. This generic model will now be modified to include the requirements of an IDS. The models will only focus on signature based IDS, as arguments have been previously provided that discount anomaly detection as a realistic method of undertaking IDS within a swarm.

6.3.1 Modelling a Separate IDS Swarm

A separate IDS swarm is defined as a number of swarm entities that have attributes and will exhibit actions based on interactions with the original swarm, with other IDS swarm members, malicious swarm members and the environment in which the IDS swarm is operating within.

The *IDS swarm* is defined as a set of entities S^d, such that:

$$S^d = \{s_0^d, s_1^d, \ldots, s_p^d\} \tag{6.1}$$

$$= \{s_i^d\} \tag{6.2}$$

Where: $i = 0, \ldots, p$

Where p is the number of entities within the IDS swarm, S^d, and each of the IDS swarm entities, s_i^d, have the same specific attributes.

The *attributes* for the overall *IDS swarm*, S^d, are defined as:

$$SA^d = \{sa_0^d, sa_1^d, \ldots sa_v^d\} \tag{6.3}$$
$$= \{sa_i^d\} \tag{6.4}$$

Where: $i = 0, \ldots, v$

Where sa_i^d is an attribute of the IDS swarm and v is the number of attributes, which are specific to the IDS swarm, S^d.

The *attri*butes for an *IDS swarm entity*, s_i^d, are defined by:

$$RA^d = \{ra_0^d, ra_1^d, \ldots, ra_b^d\} \tag{6.5}$$
$$= \{ra_b^d\} \tag{6.6}$$

Where: $i = 0, \ldots, b$

Where ra_i^d is an attribute of an IDS swarm entity and b is the number of attributes specific to an IDS swarm entity, s_i^d.

Various attributes of an IDS swarm entity, s_i^d, could be similar to those of an original swarm entity, s_i^o. The actual attributes will depend on the various swarm implementations but may include, as examples, such items as: location, velocity and remaining power available. Other attributes of the IDS swarm, S^d, will be specific to the task of undertaking intrusion detection within the original swarm, S^o.

The overall goal of an IDS swarm, S^d, is the successful detection of malicious activities, which is perceived as being caused by a malicious attack in the form of a malicious swarm, S^m, attempting to manipulate the original swarm, S^o, within an operating environment, E, or the malicious activity is being caused by malicious modification to the operating environment, E.

For example, say attribute sa_0^d of the IDS swarm S^d is the detection of malicious activities from the malicious swarm S^m, that is being undertaken by malicious swarm entities s^m. This is being undertaken within a local environment e, of the overall operating environment, E.

Although the overall goals of a swarm can be described, it is the actions and interactions of the individual swarm entities and their operating environment, local to the swarm entities, that the consequences of these actions are realised. Which, in turn, leads to the collective emergent behaviour of the overall swarm.

It is therefore required to consider the IDS swarm entities, s^d, that constitute the overall IDS swarm, S^d, in order to understand how the intrusion detection will be undertaken.

We will therefore consider an individual IDS swarm entity, s_i^d, and consider its attributes.

Let us say that the attribute ra_0^d is the goal of an IDS swarm entity and that the current status of the goal can be one of several states, then:

$$ra_0^d = \left\{ g_0^d, g_1^d, \ldots, g_q^d \right\} \tag{6.7}$$

Where g_i^d is one of the possible states of the attribute ra_0^d, of which there are q possible individual states.

Possible states could be:

No Malicious Activity Detected The IDS swarm entity, s_i^d, did not report that it suspected any abnormal activity

Malicious Activity Detected The IDS swarm entity, s_i^d, reported that it did suspect abnormal malicious activity

Potential Malicious Activity Detected The IDS swarm entity, s_i^d, has reported potential abnormal activity but is not confident that it definitely is malicious activity. For example, unusual activity might have been detected but a trigger threshold has yet to be reached

If we therefore consider the possible states for ra_0^d, this can be expressed as:

$$ra_0^i = \begin{cases} g_0^i & if \quad \text{condition } 0 \\ g_1^i & if \quad \text{condition } 1 \\ \vdots & \vdots \quad \vdots \\ g_q^i & else \quad \text{condition } q \end{cases} \tag{6.8}$$

Where condition, c_i^d evaluates to be either true or false.

$$c_i^d = \{TRUE, FALSE\} \tag{6.9}$$

and:

$$c_i^d = f(IDS) \tag{6.10}$$

That is, c_i^d is calculated from a IDS function, $f(IDS)$, that relates to the methods undertaken by the IDS swarm entity, in order to attempt to detect malicious activity.

The function $f(IDS)$ will be dependent upon what the IDS swarm entity is trying to observe and therefore achieve.

The IDS swarm could be attempting to observe the original swarm entities, s^o, for unusual behaviour, it could be attempting to locate malicious swarm entities, s^m, by characteristic relating to their behaviour, or it could be trying to locate unusual activities with the local operating environment, e.

To aid with understanding, these three possible methods of attempting to detect malicious activity by the IDS swarm are considered individually.

6.3.1.1 Monitoring the Original Swarm

Let us consider that the IDS swarm, S^d, only observes the entities from the original swarm, S^o, in order to attempt to detect malicious activity. This section shall be considered from the perspective of a swarm entity, as opposed to that of the swarm as a whole. As discussed earlier, the activities are actually undertaken by the individual swarm entities, in order to achieve the overall goal of the swarm.

Therefore, if an IDS entity, s_i^d, is attempting to observe abnormal behaviour of the original swarm entities, s_i^o, then s_i^d will be observing a subset of S^o, that is within the observation range of s_i^d.

Let us say that an IDS swarm entity, s_i^d can observe m entities of the original swarm S^o, which we will call S^o_{OBS}.

Therefore:

$$S^o_{OBS} = \{s^o_0, s^o_1, \ldots, s^o_m\} \subset S^o \tag{6.11}$$

And therefore the process of undertaking the function to attempt to locate any malicious activity that is affecting the swarm is:

$$ra^d_0 = f_{IDS}(S^o_{OBS}) \tag{6.12}$$

In the most basic form, an IDS swarm entity, s_i^d, will report ra^d_0 as $TRUE$, if any of the of the observations of the swarm entities, s_i^o, by $f_{IDS}(S^o_{OBS})$ report that they suspect malicious activity is altering the behaviour of s_i^o. Otherwise ra^d_0 will report $FALSE$, as $f_{IDS}(S_{OBS})$ has not reported any malicious activity being detected.

This could also be shown as:

$$ra^d_0 = (f_{IDS}(s^o_0) \vee f_{IDS}(s^o_1) \vee \ldots \vee f_{IDS}(s^o_m)) \tag{6.13}$$

Which is simplified to:

$$ra^d_0 = \bigcup_{j=0}^{m} f_{IDS}(s^o_j) \tag{6.14}$$

This highlights that the attribute ra_0^d, as an outcome of $f(IDS)$, does not distinguish between the different swarm entities, s_i^o, within subset S_{OBS}^o.

In order to identify the particular entities within S_{OBS}^o that s_i^d believes to be acting under the influence of a malicious activity, s_i^d could then have another attribute that is used to store this information, say ra_1^d.

That is:

$$ra_1^d = \{s_0^o, s_1^o, \ldots, s_n^o\} \subset S_{OBS}^o \tag{6.15}$$

Where n is the amount of swarm entities that are believed to be exhibiting unusual behaviours, that would indicate malicious activities.

The information stored in ra_1^d could then be used to aid further analysis of suspected issues.

If there were any swarm entities within ra_1^d that were not actually under the influence of any malicious activity, these results would be known as *false positives*, as s_i^d believed their operating characteristics indicated that they were being influenced by a malicious attacker. That is, the assessment of $f_{IDS}(S_{OBS}^o)$ by s_i^d suggested that the swarm entities were acting in a manner that lead $f_{IDS}(S_{OBS}^o)$ to calculate that they were being influenced by a malicious actor, when in fact they were not.

6.3.1.2 Monitoring for Suspected Malicious Entities

Let us consider that the IDS swarm, S^d, has the ability to monitor for other entities, that are not part of the original swarm, S^o.

This is included as a malicious actor has to be able to provide stimulus to the original swarm entities, such as by communications, in order to influence the original swarm in order to achieve an effect. In a practical sense, this means that, an original swarm entity, s_i^o, considers that a malicious swarm entity, s_i^m, to be a member of the original swarm, S^o, when in fact it is a member of a malicious swarm, S^m. The malicious intruder will have realised the threat of *masquerade*, in order to achieve the ability for an original swarm entity, s_i^o, to have perceived s_i^m as a legitimate member of the original swarm, S^o.

If we only consider that the IDS swarm entity, s_i^d, is attempting to observe abnormal behaviours of the malicious swarm, S^m, and is not observing the original swarm, S^o, then an IDS swarm entity, s_i^d, will be observing a subset of S^m, that is within the observation range of s_i^d.

Let us say that an IDS swarm entity, s_i^d can observe p entities of the malicious swarm S^m, which we will call S_{OBS}^m.

Therefore:

$$S^m_{OBS} = \left\{ s^m_0, s^m_1, \ldots, s^m_p \right\} \subset S^m \tag{6.16}$$

And therefore the process of undertaking the function to attempt to locate any malicious activity that is affecting the swarm is:

$$ra^d_0 = f_{IDS}(S^m_{OBS}) \tag{6.17}$$

As discussed earlier, in the most basic form, an IDS swarm entity, s^d_i, will report ra^d_0 as $TRUE$, if any of the of the observations of the swarm entities, s^m_i by $f_{IDS}(S^m_{OBS})$ report that they suspect malicious activity behaviour by s^m_i. Otherwise ra^d_0 will report $FALSE$, as $f_{IDS}(S^m_{OBS})$ has not detected any malicious activity.

Similarly, this could also be shown as:

$$ra^d_0 = \bigcup_{j=0}^{p} f_{IDS}(s^m_j) \tag{6.18}$$

Again, this shows that the attribute ra^d_0, as an outcome of $f_{IDS}(S^m_{OBS})$, does not distinguish between the different swarm entities, s^m_i, within subset S^m_{OBS}.

In order to identify the particular entities within S^m_{OBS} that s^d_i believes to be a malicious entity, s^d_i could again have another attribute that is used to store this information, say ra^d_1.

That is:

$$ra^d_1 = \left\{ s^m_0, s^m_1, \ldots, s^m_q \right\} \subset S^m_{OBS} \tag{6.19}$$

Where q is the amount of swarm entities that are believed to be exhibiting unusual behaviours, that would indicate malicious activities.

The information stored in ra^d_1 could then be used to aid further analysis of suspected issues.

In this instance the difference between the values of p and q would be considered to be the number of *false negatives* by s^d_i, as these s^m_i entities had been a member of S^m_{OBS} but had not been considered to be malicious by their actions. That is, although the members of S^m_{OBS} where within detection range of s^d_i, the assessment by $f_{IDS}(S^m_{OBS})$ had not reviled these as suspected malicious entities, even though they were.

6.3.1.3 Monitoring the Operating Environment for Malicious Activity

Finally, let us consider how a IDS swarm, S^d, could have the ability to detect malicious activity within the operating environment, E.

The assumption is that an IDS swarm entity, s_i^d will have the ability to monitor an area of the environment, which is local to itself, e_i. Essentially, s_i^d is required to be able to observe its local environment, in order to be able to operate successfully within the environment. For example, swarm entities might have the ability to detect obstacles, in order to avoid them, so as to prevent damage to themselves.

However, the difference for an IDS swarm, S^d, is that it also has the requirement to attempt to detect anomalies, or anomalous behaviours, within the operating environment.

Therefore, an IDS swarm entity, s_i^d, will observe the local environment around itself, which will be a subset of the entire operating environment, E.

To aid in clarification, we will assume that s_i^d divides its observable area up to into smaller, more manageable, areas. That is:

$$E_{OBS} = \{e_1, e_2, \ldots, e_r\} \subset E \tag{6.20}$$

Where r is the maximum amount of the local environment sub-sections that s_i^d can observe.

s_i^d can then perform the IDS function on its local operating environment, E_{OBS}. The function $f_{IDS}(E_{OBS})$ would be tailored for the local operating environment, e_i, of the IDS swarm S^d:

$$ra_0^d = f_{IDS}(E_{OBS}) \tag{6.21}$$

Depending what is required of the IDS function from s_i^d, regarding any perceived concerns from the local environment, will depend on the attributes associated with s_i^d. That is, if all that is required to be understood is if the area has perceived any malicious activity or not, then the simpler techniques could again be utilised.

The IDS swarm entity, s_i^d, will report ra_0^d as $TRUE$, if any of the of the observations of the locally observable environment E_{OBS} by $f_{IDS}(E_{OBS})$ report that they suspect malicious activity behaviour either by or within e_i. Otherwise ra_0^d will report $FALSE$, as $f_{IDS}(E_{OBS})$ has not reported any malicious activity being detected.

Similarly, this could also be shown as:

$$ra_0^d = \bigcup_{k=0}^{r} f_{IDS}(e_k) \tag{6.22}$$

Again, this shows that the attribute ra_0^d, as an outcome of $f_{IDS}(E_{OBS})$, does not distinguish between the different local environment areas, e_i, within the subset E_{OBS}.

In order to identify the particular areas within E_{OBS} that s_i^d believes to contain malicious activity, s_i^d could again have another attribute that is used to store this information, say ra_1^d.

That is:

$$ra_1^d = \{e_0, e_1, \ldots, e_b\} \subset E_{OBS} \tag{6.23}$$

Where b is the amount of areas within the local operating environment that are believed to be exhibiting unusual behaviours, that would indicate malicious activities.

The information stored in ra_1^d could then be used to aid further analysis of suspected issues.

If there were any areas of the local environment stored within ra_1^d that were not actually under the influence of any malicious activity, these results would be known as *false positives*, as s_i^d believed their operating characteristics suggested that they were being influenced by a malicious attacker. That is, the assessment of $f_{IDS}(E_{OBS})$, by s_i^d, indicated that the local operating environment displayed characteristics that that lead $f_{IDS}(E_{OBS})$ to calculate that the areas were being influenced by a malicious actor, when in fact they were not.

Similarly, the difference between the values of r and b would be considered to be the number of *false negatives* by s_i^d, as these e_i local operating areas had been a member of E_{OBS} but had not been considered to be malicious under observation. That is, although the members of E_{OBS} where in observation area of s_i^d, the assessment of $f_{IDS}(E_{OBS})$ had not suggested that these areas were suspected of malicious activity, even though they were.

6.3.1.4 Combining the Models

As can be seen from the the proposed models, they have all been considered, such that they essentially follow the same format. It is therefore proposed that the models can be combined, such that there is a single model for an independent IDS swarm.

Assuming that there is an environment, E, that contains an original swarm, S^o, and an independent IDS swarm, S^d. The role of the IDS swarm, S^d, is to attempt to detect and advise of any malicious activity, which could be from another malicious swarm, S^m, or from within the operating environment, E. As already discussed, the actual IDS functionality is required to take place at the individual swarm entity level, s_i^d, as the individual swarm entities have specific swarm attributes, such as they are only capable of local sensing, swarm characteristic 5, and local communications, swarm characteristic 6.

This results in the following proposed model:

$$ra_0^d = f_{S^o IDS}(S^o_{OBS}) + f_{S^m IDS}(S^m_{OBS}) + f_{EIDS}(E_{OBS}) \tag{6.24}$$

Where:

ra_0^d - attribute of s_i^d
Contains the current state of s_i^d, as to whether it believes that there is malicious activity that could influence s_i^o.

$f_{S^o IDS}(S^o_{OBS})$ - an IDS function that is used to monitor the original swarm, S^o
This function is conducted by observing the original swarm entities, s_i^o, within the observation range of s_i^d, in order to determine if any of the individual swarm entities are exhibiting characteristics that would suggest that they are being influenced by malicious activities.

$f_{S^m IDS}(S^m_{OBS})$ - an IDS function that aims to detect the activities of a malicious swarm, S^m
This function is undertaken by attempting to detect and observe malicious swarm entities, s_i^m, that are attempting to influence the original swarm, S^o, within the observation range of s_i^d.

$f_{EIDS}(E_{OBS})$ - an IDS function that aims to detect malicious activities within the operating environment, E
This function has the goal of detecting malicious activities within the local operating environment, e_i, that is observable by the IDS swarm entity, s_i^d that could influence the original swarm, S^o.

6.3.1.5 Modelling an IDS Swarm

The actual implementations of IDS functions will depend upon the swarm's implementation. However, following on from the discussions from Section 6.2.2.1 and Section 6.2.2.2, where an argument was made for only signature based IDS to be considered, the IDS functions can be further refined.

It is proposed that, for a swarm, S^o, an IDS swarm, S^d, will be attempting to observe characteristics that are not within the normal operating characteristics for the original swarm, S^o.

Again, this will not be considered from a swarm perspective but from the perspective of an individual swarm entity, for the various reasons that have already been presented. Therefore, an IDS swarm entity, s_i^d, can observe the characteristic of other swarm entities, both from the original swarm, S^o, and a malicious swarm, S^m.

Considering an individual IDS swarm entity, s_i^d, that can observe s_{obs} other swarm entities.

s_i^d has the ability to observe a predetermined set of characteristics $\{ch_0, ch_1, \ldots, ch_{all}\}$, where ch_{all} is all of the characteristics that s_i^d can observe.

s_i^d undertakes a review of the observed characteristics, in an attempt to see if any of the characteristics operate outside expected behaviours.

That is, s_i^d knows what is expected, from the perspective of normal behaviours, from swarm entities s_i^o and reacts to observations that fall outside this.

It is not attempting to observe specific behaviours or characteristics that are the signature of a specific malicious attack.

Therefore, as an example, if ch_0 is the characteristic for the maximum distance a swarm entity should move in a straight line before it changes its heading, ch_0 will have an attribute relating to this, say $StraightDist_{max}$. If s_i^d observes s_i^o for a period of time and during this period of time the value of $ch_0 > StraightDist_{max}$, then s_i^d will assume that this uncharacteristic behaviour of s_i^o is due to malicious influence.

It is worth noting that, in the example, the malicious behaviour was suspected due to uncharacteristic behaviours and did not take into account any of the potential messaging. That is, although s_i^d might receive and subsequently act upon messages received from other swarm entities, which could include swarm entities from both the original swarm, S^o, and the malicious swarm, S^m, these communications were not considered within the IDS function example.

To place this into context, assume that swarm entities transmit a bearing, which is an integer value between 0 and 359. If a receiving swarm entity receives a value that does not conform with this requirement, such a value is received that is outside of the bounds or it is not a whole number, $\mathbb{W}$, the receiving swarm entity will assume that this is an error. Such an error could have been caused by noise on a data link and the swarm would have been designed to expect data link errors and manage such constraints. It would not believe that it was due to malicious actions.

This can be shown as:

$$f_{ErrorCheck}(RxBearing) = \begin{cases} FALSE & \{RxBearing \in \mathbb{W} \mid 0 \leq RxBearing < 359\} \\ TRUE & \{RxBearing \in \mathbb{W} \mid 359 < RxBearing < 0\} \\ TRUE & \{RxBearing \notin \mathbb{W}\} \end{cases} \quad (6.25)$$

If the function $f_{ErrorCheck}(RxBearing)$ returns $FALSE$, the received bearing is assumed to be within the constraints of that variable and the receiving swarm entity will respond accordingly. There is no perceived error with the value received.

If the function $f_{ErrorCheck}(RxBearing)$ returns $TRUE$, the receiving swarm entity assumes that an error has occurred and ignores the received bearing value.

If we consider the example relating to the maximum distance that a swam entity should travel in a straight line, the format of the checking follows a similar construct:

$$f_{IDS}(SLDist) = \begin{cases} FALSE & \{SLDist \in \mathbb{R} \mid 0 \leq SLDist \leq StraightDist_{max}\} \\ TRUE & \{SLDist \in \mathbb{R} \mid SLDist > StraightDist_{max}\} \end{cases} \quad (6.26)$$

In this example, the function $f_{IDS}(SLDist)$ calculates the current distance that an observed swarm entity has travelled in a straight line, $SLDist$. The IDS function then returns $FALSE$ if the observed swarm entity has travelled in a straight line without altering course before reaching a maximum permitted threshold.

The function $f_{IDS}(SLDist)$ will return $TRUE$ if the swarm entity under observation has travelled for more than a maximum distance in a straight line. It therefore suspects that there is malicious activity being carried out.

Therefore, to describe this from a generic perspective:

$$f_{IDS}(AttObs) = \begin{cases} FASLE & \{Attribute_{MinValue} \leq AttObs \leq Attribute_{MaxValue}\} \\ TRUE & \{Attribute_{MinValue} > AttObs > Attribute_{MaxValue}\} \end{cases} \quad (6.27)$$

Essentially, an attribute of a swarm entity is observed, $AttObs$, and the observations of the attribute are compared against the known characteristics of the attribute.

The IDS function will report $FALSE$, no malicious activity suspected, if the attributes being observed by the function are within the expected and constrained limits. If the IDS function reports $TRUE$, malicious activity suspected, as the attributes being observed are outside of pre-determined limits.

It is worth noting that the simplicity of this formula is of benefit to the requirements of a swarm entity, when considering the characteristics and constraints of a swarm that are being considered in this research. As described in Section 2.3, this research considers implementations of swarms where the swarm entities that have limited availability for processing and storage, as well as taking into account the swarm characteristics described in Section 2.4.

However, it should be noted that the implementation of such an IDS function will be dependent upon the actual implementation and characteristics of both the original swarm, S^o, and the IDS swarm, S^d. For example, to enable the $f_{IDS}(SLDist)$ IDS function to operate, the IDS swarm entity, s_i^d, will be required to have sufficient storage, in order to store and process previous observations, in order to calculate the straight line distance travelled and then conduct the IDS function.

6.3.1.6 Modelling the IDS within the Swarm

So far, the discussions have been considering a separate IDS swarm, S^d, as providing the IDS functionality for the original swarm, S^o. However, it is possible for the IDS functionality to be undertaken by the original swarm.

The IDS function is essentially the same as previously discussed within a separate IDS swarm, where an attribute is observed and then compared to what are its predefined operating limits and a decision is computed, based on the observed value and permitted limits.

There are several advantages to undertaking the IDS function within the original swarm entities, such as the swarm entity knows actual values, as opposed to observations, and the original swarm entity can then act accordingly.

There is also the logistical advantage that a second swarm does not have to be released. That is, the original swarm operator does not have to invest in and manage a second swarm.

The disadvantage is that the original swarm now has to undertake extra functionality, in what is already a resource constrained system. There is also the assumption that the original swarm has the capabilities to observe extra attributes, to aid in the IDS functionality. For example, due to resource constraints within the swarm entity, the IDS functionality would need to be undertaken without the addition of extra sensors.

As an example of how the original swarm could undertake IDS functions, consider the proposal for a swarm, S^o, to be utilised to locate and deactivate landmines, within an operational area, E [112, 113, 133], as in use-case 2.

The original swarm elements, s_i^o, will deactivate a landmine when there are four swarm entities present at the position of a landmine.

In order to do this, when a swarm entity locates a landmine, it releases a scent, in order to attract other swarm entities. This is a similar methodology to animals releasing pheromones, in order to communicate with other animals. Kumar, Sahin and Chapman's research refers to this as the collective recruitment of the swarm entities.

That is, a quantity of landmines, M, are located within the operating environment, E and the swarm, S^o, has the goal of locating and disarming the landmines.

There are several attacks that could be undertaken against the original swarm, as described in Section 4.5.

From the perspective of a landmine, and not the perspective of a swarm entity, a landmine will be disarmed in accordance with the following function:

$$f_{DISARM}(Landmine) = \begin{cases} FALSE & \{s_i^o Qty \in \mathbb{W} \subset S^o | s_i^o Qty < s_i^o Req\} \\ TRUE & \{s_i^o Qty \in \mathbb{W} \subset S^o | s_i^o Qty \geq s_i^o Req\} \end{cases} \tag{6.28}$$

Where:

$s_i^o Qty$ is the amount of swarm entities

$s_i^o Req$ is the amount of swarm entities required to disarm a landmine, which is four in the example

That is, $f_{DISARM}(Landmine)$ will report $FALSE$ if the whole number quantity of swarm entities is less than four. The $f_{DISARM}(Landmine)$ function will report $TRUE$ when there are four or more swarm entities situated at a landmine location and the landmine will be disarmed.

As the swarm, S^o, is aware of the requirements in order to disarm a landmine, this knowledge can be utilised in order to attempt to detect malicious activity. That is, if a swarm element, s_i^o, detects the presence of three, or more, other swarm entities at a particular location, then it could suspect malicious activity. This is because the presence of itself and three other swarm entities should have disarmed the landmine at this location and the swarm entities should have then moved on, in an attempt to locate and disarm other landmines.

This can be shown as:

$$f_{IDS}(s_i^o Qty) = \begin{cases} FALSE & \{s_i^o Qty \in \mathbb{W} \mid 0 \leq s_i^o Qty < s_i^o Req\} \\ TRUE & \{s_i^o Qty \in \mathbb{W} \mid s_i^o Qty \geq s_i^o Req\} \end{cases} \tag{6.29}$$

The function $f_{IDS}(s_i^o Qty)$ observes the quantity of of swarm entities at a particular location and will report $TRUE$ when it detects more than the required amount of swarm entities that are needed at the same location and therefore suspects malicious activity. This assumes that a landmine is instantly deactivated with the presence of four entities at the landmine's location. In practise, the time taken to disarm a landmine might need to be considered, to prevent false positives. That is, where there are four swarm entities located at a landmine and they are in the process of disarming the landmine.

Therefore, $f_{IDS}(s_iQty)$ could be further refined to:

$$f_{IDS}(s_x^oQty) = \begin{cases} FALSE & \{s_i^oQty \in \mathbb{W} \mid 0 \leq s_i^oQty < s_i^oReq\} \\ TRUE & \{s_i^oQty \in \mathbb{W} \mid (s_i^oQty \geq s_i^oReq) \wedge (t_{disarm} > t_{max})\} \end{cases} \tag{6.30}$$

Where:

t_{disarm} is the elapsed time when the required amount of swarm entities are located at the landmine location.

t_{max} is the maximum amount of time to disarm a landmine.

Although this is showing detail of a specific swarm implementation, it does demonstrate how particular results can be dependent upon the values of multiple attributes. In this example, it is the quantity of entities at a particular location and time at that location.

It is also worth noting that, in this example, the IDS implementation might only consider the amount of entities at a particular location. That is, as opposed to f_{IDS} reporting $TRUE$ when considering $s_i^oQty \geq s_i^oReq$, the designer might consider using $s_i^oQty > s_i^oReq$ instead. By utilising greater than, as opposed to equal to or grater than, the designer is able to implement a function that is simpler and requires less storage and processing, as it does not have to consider time, and therefore assists with the inherent resource constraints of a swarm entity.

This demonstrates how an IDS function will vary, based upon the actual implementation of a swarm.

However, the principle still remains the same, in that the swarm entity is attempting to detect attributes that are operating outside of their normal operating characteristics or constraints. It also demonstrates how certain attributes might have dependencies upon other attributes.

6.4 Summary

This chapter has provided an overview of Intrusion Detection Systems and has provided details as to the mechanisms for the various types of IDS techniques. Comparisons are made to systems with similar properties to swarms and how IDS had been considered for these similar systems, detailing how swarms had different requirements due to the swarm's unique characteristics.

The chapter highlighted how anomaly detection was not a suitable IDS technique within a swarm and how signature based IDS techniques could be utilised, if tailored appropriately to the implementation of a swarm. Details were provided regarding the implementations of IDS within the swarm itself and within a separate IDS swarm.

The chapter concluded by modifying the generic swarm model, in order to take into account the IDS functions. This model was then implemented and the attacks undertaken in Chapter 5 were then re-run, to understand if IDS implementations can improve the efficiency of the swarm in reaching its goal. This research takes into account the two swarm use-case studies and is presented in Chapter 7.

7 Simulations of Robotic Swarm Intrusion Detection Systems

7.1 Introduction

This chapter provides details of how the proposed models for robotic swarm intrusion detection systems were simulated. This was conducted in order to consider the possibility of intrusion detection systems being realised within robotic swarms.

This follows on from the proposed IDS model, presented in Chapter 6, and the previous simulations in Chapter 5, which demonstrated that swarms could be successfully manipulated by attacks.

IDS implementations were developed and simulations were undertaken on the swarm implementations that had previously been subjected to a malicious attack, as described in Chapter 5. This ensured that the simulations were for a variety of swarm implementations were being considered, based upon the swarm taxonomies in Section 2.5. This included both interactions between the swarm entities and interactions between swarm entities and the swarm's operating environment.

The intrusion detection was considered from the perspectives of both external to the swarm, where a separate IDS swarm is utilised, and internally to the swarm, with the swarm entities undertaking the IDS themselves.

7.2 Independent Swarm IDS

The scenario was for an independent IDS swarm to be simulated within an operating environment that contained an original swarm. The IDS swarm was attempting to identify uncharacteristic behaviour of the original swarm, which would indicate malicious activity.

The scenario was designed such that the IDS swarm interacted with and performed the same as the original swarm. However, upon an IDS swarm entity detecting suspicious activity, that IDS swarm entity would begin to work independently to assess and, if required, act upon the situation. Upon completion of its actions, the IDS swarm entity returns to interacting with the original swarm and other IDS swarm entities.

In this section, the attack that is being undertaken against the swarm is a misinformation attack, that has been undertaken by the realisation of the threat of masquerade, such that a malicious swarm has the ability to influence the original swarm. This is achieved by malicious swarm entities being able to masquerade within the original swarm, such that the original swarm entities believe the malicious swarm entities to be members of the original swarm. As such, the original swarm entities will react to actions and communications of the malicious swarm entities as if they were genuine.

When considering use-case 1, several attack simulations were undertaken against a swarm which had the goal of locating a target within an operating environment and utilised collaborative navigation techniques in order to achieve the goal [59].

As described in Section 5.2.3.5, one of the attacks that were simulated involved a malicious swarm entity locating the target and then attempting to lead the original swarm entities away from the target. The method of the attack was that when a malicious swarm entity located a target, the malicious swarm entity would turn through 180° and then move away in a straight line and broadcasting information as to what appeared to be better navigational information regarding the location of the target. The effect was that the searchers within the original swarm, that were hunting for the target, would follow the malicious entities, away from the target.

This attack was then modified, such that when the malicious entity came to an obstacle, it would stop and continue to broadcast false navigational information. Therefore, two IDS implementations were proposed, one for detecting straight line movement and one for detecting stationary entities.

7.2.1 Implementing the Straight Line IDS

Within the simulations, the independent IDS swarm entities had the goal of identifying uncharacteristic behaviours of the observed swarm entities. As the threat of masquerade has been realised, this could be entities from either the original swarm or the malicious swarm.

It was realised that the misinformation attack would cause entities from both the original and malicious swarms to move in a straight line until they reached an obstacle. If an entity reached an obstacle it would react in a way to avoid colliding with the obstacle, such as "reflecting" off a wall or remaining stationary.

The first proposal for the IDS was to attempt to detect the straight line movement of a swarm entity. Although swarm entities are expected to travel in straight lines, there is a maximum distance that they are able to travel in a straight line, before they would randomly change direction.

The objective of the independent IDS swarm was to therefore identify swarm entities that were travelling in a straight line, which had travelled in a straight line gone beyond a maximum limit expected for a swarm entity.

It should be noted that the independent IDS swarm did not have the objective of locating the target, which was the goal of the original swarm's search. This was because the objective of the simulation was to determine if an IDS was possible within a swarm. There was also the consideration that in a practical implementation of a swarm, a searcher entity might also require a different or unique fit, in order to find the target, such as specialist sensors.

As described in Equation 6.27, the generic IDS function model for observing swarm attributes was defined as:

$$f_{IDS}(AttObs) = \begin{cases} FALSE & \{Attribute_{MinValue} \leq AttObs \leq Attribute_{MaxValue}\} \\ TRUE & \{Attribute_{MinValue} > AttObs > Attribute_{MaxValue}\} \end{cases} \quad (7.1)$$

The IDS implementation was therefore modelled as:

$$f_{IDS}(SLDist) = \begin{cases} FALSE & \{SLDist \in \mathbb{R} \mid 0 \leq SLDist \leq StraightDist_{max}\} \\ TRUE & \{SLDist \in \mathbb{R} \mid SLDist > StraightDist_{max}\} \end{cases} \quad (7.2)$$

Where:

$f_{IDS}(SLDist)$ - Straight Line Distance IDS function being carried out by an IDS swarm entity.

$SLDist$ - The Straight Line Distance of other entities, as calculated by an IDS swarm entity.

$StraightDist_{max}$ - The maximum Straight Line distance that an original swarm entity is expected to travel.

The simulations were conducted from the perspective of the IDS swarm entities, as opposed to being from a global perspective. That is, straight line movements were not detected for within the simulated operating environment, such as by observing the locations and movements of all the swarm entities and identifying those that travelled in a straight line beyond a maximum distance. Instead, the simulation was conducted from the perspective of a swarm entity. That is, the individual IDS swarm entities were simulated to take into account the swarm entities characteristics and associated limitations, such as swarm characteristic 5 in that the IDS swarm entities were only able to observe and sense their local environment.

Initial simulations were conducted to prove the IDS concept, with the aim of determining of an IDS swarm could locate entities that are travelling in a straight line, beyond their normal operating characteristics. These simulations were conducted within either an empty operating environment, with no maze, or an operating environment with a simple maze.

Malicious entities were placed in the operating environments, in one of three configurations, as shown in Figures 7.1, 7.2 and 7.3, along with varying numbers of IDS entities in the centre of the operating environment. The malicious swarm location configurations were exactly the same for an empty operating environment. A target was randomly placed within the operating environment and various numbers of independent entities were randomly placed within the operating environment.

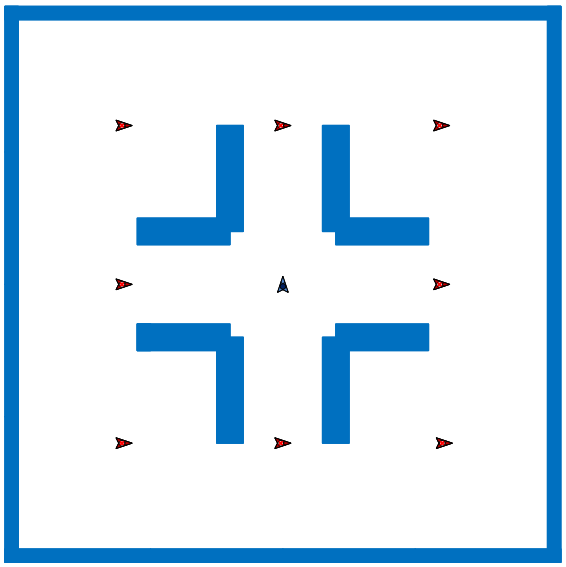


Figure 7.1: IDS - Small Malicious Entity Configuration

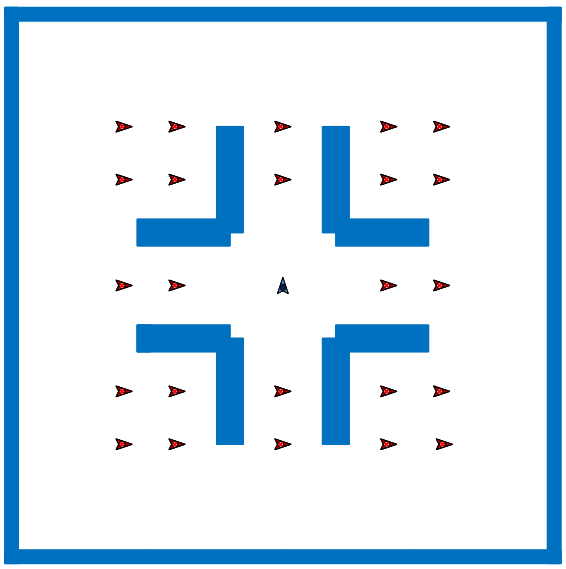


Figure 7.2: IDS - Medium Malicious Entity Configuration

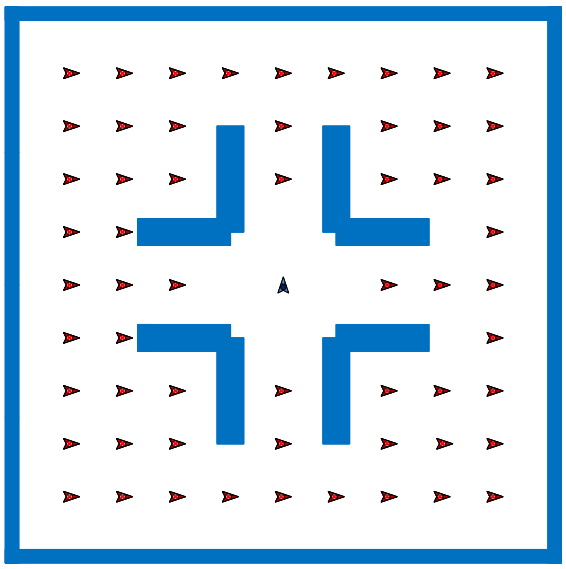


Figure 7.3: IDS - Large Malicious Entity Configuration

The simulation enabled the malicious entities to utilise the navigational information, provided by the independent entities, to locate the target. Once the target had been located, the malicious entity would then turn away and travel in a straight line, reflecting off walls where required. The goal of the IDS swarm was to attempt to detect the straight line movement.

The method used within the simulations was designed to be relatively simple, due to the processing constraints being considered for the swarm entities within this research.

The IDS swarm entities behaved in a similar fashion as if they were a searcher from within the original swarm. That is, they would receive information from the other independent swarm entities within their range of communication and they would attempt to make their way towards the target, based on the information that they received. Therefore, if a malicious entity has located a target it will be broadcasting false, but better, navigational information. The IDS entities will therefore attempt to make their way towards the malicious entity. However, the malicious entity, unlike a target, will move in a straight line and therefore, so will the IDS entity. The IDS entity will essentially follow the malicious entity that is broadcasting the false navigational information.

The IDS swarm entity keeps a local record of its own distance moved on a particular bearing, within a defined tolerance for the bearing. That is, if an IDS swarm entity moved on a bearing that did not change more than a pre-defined tolerance, then this would be considered by the IDS entity as continuing to move as if it were a straight line and maintain a record of the distance travelled. This is shown in Figure 7.4, where ϑ is the detection angle used to detect and track other entities in order to determine if the IDS entity is travelling in a straight line.

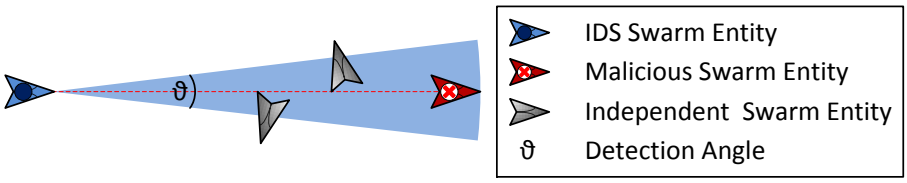


Figure 7.4: Mobile IDS Implementation

If the IDS swarm entity changes its bearing before a maximum permitted straight line distance is reached, either due to receiving better navigational information as to the location of the target or it has reached its distance to travel in a straight line before changing direction, then the IDS swarm entity resets its distance travelled in a straight line to zero and begins the process again.

If the IDS swarm entity calculates that it has travelled a distance in a straight line that is greater than the maximum distance permitted in a straight line, then the IDS swarm entity assumes malicious activity and responds accordingly.

Two different responses were proposed to the suspected detection of malicious activity:

1. "Kill" all swarm entities within a pre-defined area in front of the IDS swarm entities bearing.

2. "Kill" the furthest swarm entity within the pre-defined area in front of the IDS swarm entities bearing.

Simulations Initial simulations considered the removal of all entities that were within the area in front of the IDS entity. when malicious activity was suspected. The rationale for removing all the swarm entities from within the area in front of the IDS swarm entity was that this would be easier to implement in practise and a swarm would not be affected by the reduction in its overall quantity of swarm entities.

However, in order to reduce the number of false-positives being recorded, the simulation was configured to minimise this. In order to do this, the tolerance for detecting straight line movement was set to 3°.

As expected, false-positive were recorded, as original swarm entities were also removed. The results were fairly consistent across the simulations. That is, the results for false-positives for a small quantity of malicious entities followed a similar profile for both with and without a maze. Likewise, medium quantities of malicious entities were similar for with and without a maze and so were the large amount of malicious swarm entities. Examples are provided in Figure 7.5 and Figure 7.6.

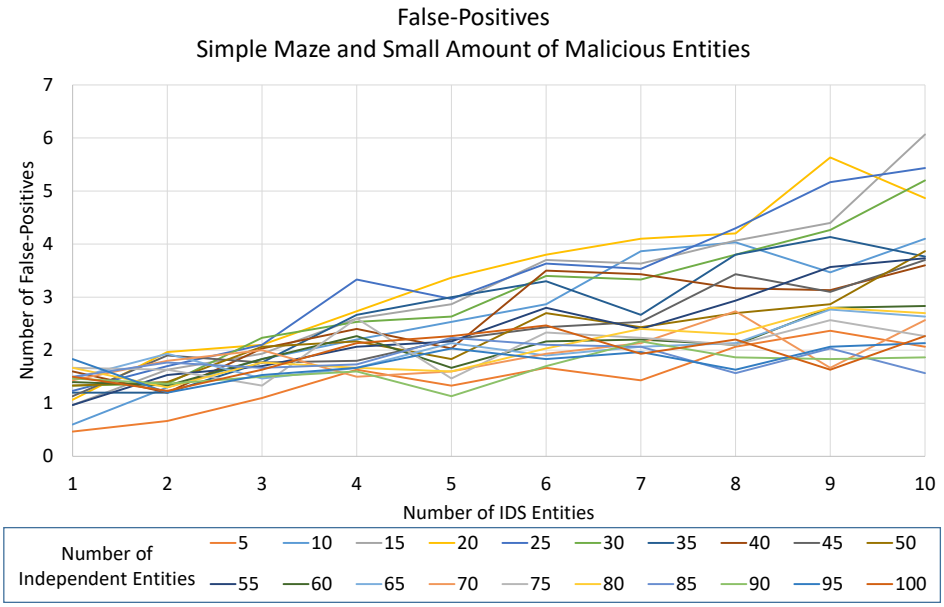


Figure 7.5: Kill All: Number of False-Positives for a Simple Maze with a Small Amount of Malicious Entities

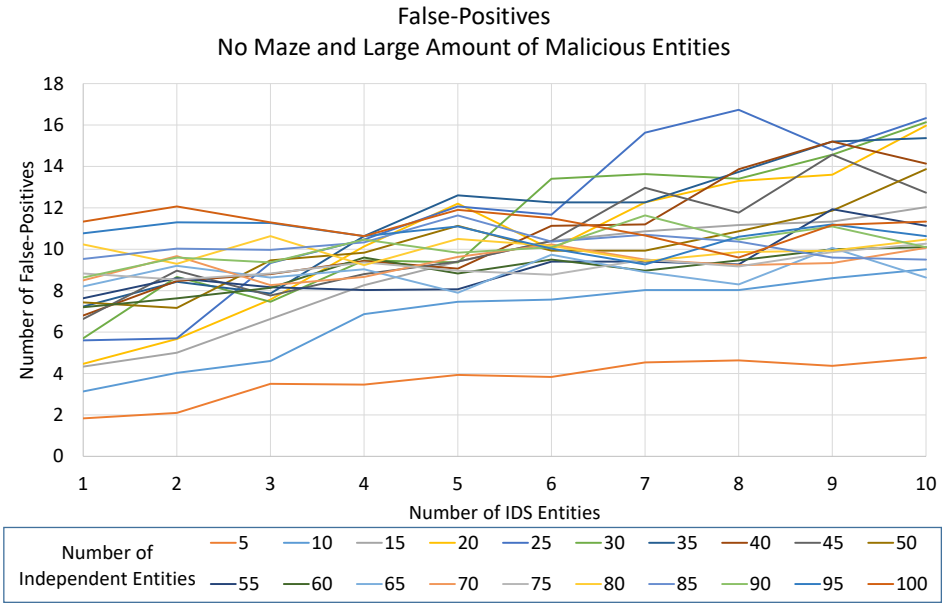


Figure 7.6: Kill All: Number of False-Positives for No Maze with a Large Amount of Malicious Entities

This was due to the fact that if any of the original independent swarm entities were in front of the IDS swarm entity, in its area of influence, when the IDS entity determined that an entity had travelled too far in a straight line, then these swarm entities would also be removed. This can be seen in Figure 7.4.

Because of this, the simulations were re-run with the action for suspected malicious activity to be that of only removing the swarm entity that is furthest away from the IDS swarm entity, within its area of influence.

Initially, the simulations were expected to produce less false-positives. However, the results were practically the same. The simulations were configured with the same conditions and the results can be seem in Figure 7.8 and Figure 7.9, for comparison.

The false-positives observed were due to an original swarm entity entering the IDS swarm entity's area of influence at the same moment as the IDS swarm entity reached its threshold for maximum distance travelled in a straight line and was therefore targetted to be removed. This can be seen in Figure 7.7, where the IDS entity is tracking the malicious entity but as the IDS entity calculates that the threshold for travelling in a straight line has been reached, the independent entity enters its area of influence.

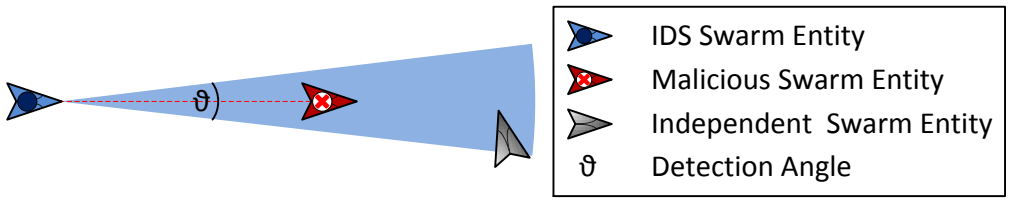


Figure 7.7: Mobile IDS False Positive

All of the simulations successfully ran to completion, in that all the malicious entities were detected and subsequently then removed.

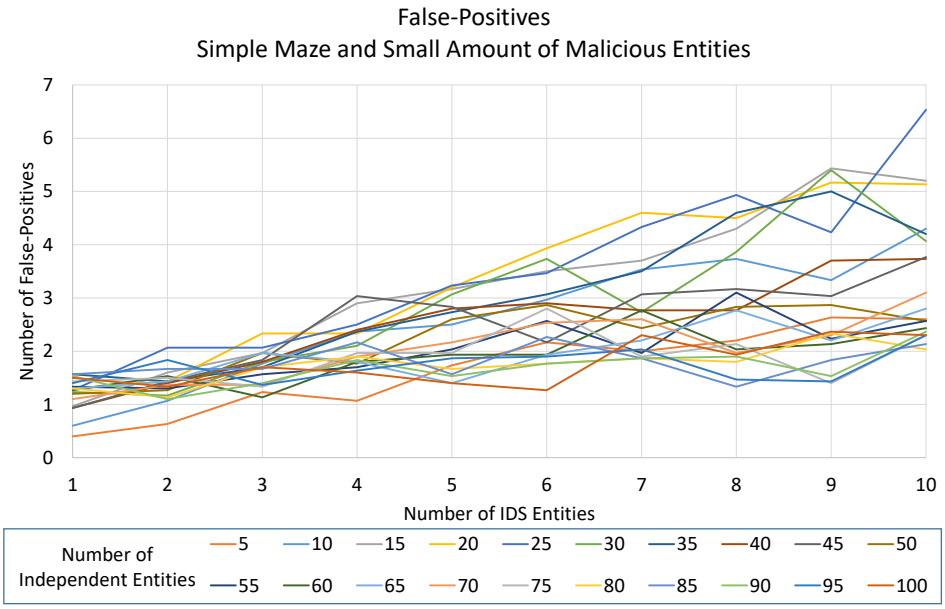


Figure 7.8: Kill Last: Number of False-Positives for a Simple Maze with a Small Amount of Malicious Entities

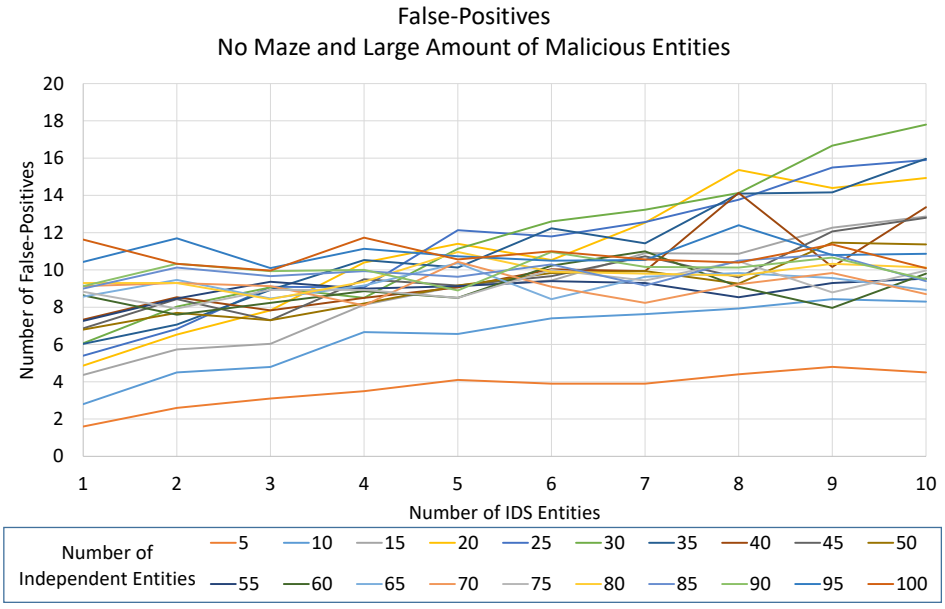


Figure 7.9: Kill Last: Number of False-Positives for No Maze with a Large Amount of Malicious Entities

7.2.2 Implementing Stationary Detection IDS

In order to investigate whether an IDS swarm could directly observe the behaviour of other swarm entities, as opposed to its interpreting its own actions, it was decided to implement a stationary detection methodology.

Therefore, the goal of the IDS swarm entity was to detect stationary swarm entities. This was achieved by the IDS swarm entity making observations within its local area and recording the positions of the other entities within this local area.

Once an IDS entity had moved, the IDS entity repeated its observations for its local area. If any of the other swarm entities were located in the positions the IDS entity had calculated for an entity if it remained stationary, then it would assume malicious activity.

Entities that were assumed to be malicious were "killed" and removed from the operating environment. The IDS implementation was therefore modelled as:

$$f_{IDS}(Stationary) = \begin{cases} FALSE & \{LocObs \neq CalculatedPos\} \\ TRUE & \{LocObs = CalculatedPos\} \end{cases} \tag{7.3}$$

Where:

$f_{IDS}(Stationary)$ - Stationary IDS function being carried out by an IDS swarm entity.

LocObs - The positions of swarm entities, as observed by an IDS swarm entity within its local area.

CalculatedPos - The positions that the IDS entity had previously calculated for stationary entities.

The positions were relative to the IDS entity and were calculated from the other swarm entities' bearings and distances, from the IDS entity. These positions were then recorded and stored within the IDS swarm entity.

As the IDS swarm entity is aware of its velocity, it can use its own bearing and speed to calculate what the relative distances and bearings of the other swarm entities would be from itself if they did not move. That is, the IDS swarm entity, using simple geometric calculations, can calculate what the position of the other swarm entities would be if they remained stationary, after the IDS swarm entity had moved. This can be seen in Figure 7.10.

The IDS swarm entity moves from location A, at time t, to location B, at time $t + 1$.

The IDS swarm entity is able to measure the bearing of the entity under observation from itself and it knows its own bearing, the IDS entity can therefore calculate angle A.

The IDS entity is also able to measure the separation distance between itself and the other entity, enabling distance b to be known.

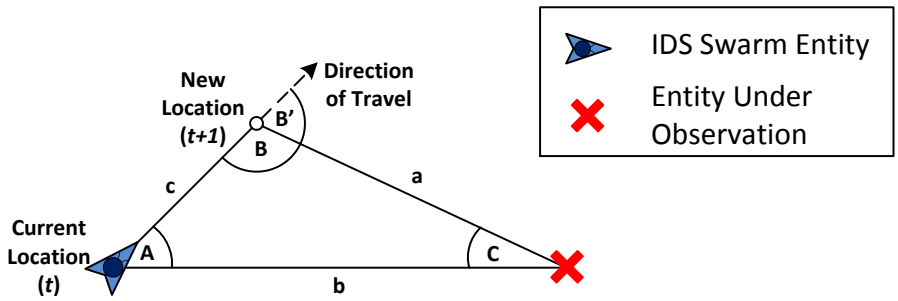


Figure 7.10: Stationary IDS Implementation

As the IDS entity is aware of its own velocity, it is able to calculate what the distance c will be, at time $t + 1$.

Using simple trigonometry, the IDS entity is able to calculate the distance a by using the cosine rule:

$$a = \sqrt{b^2 + c^2 - 2bc\ \cos A} \tag{7.4}$$

The IDS entity is also able to calculate angle B:

$$B = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right) \tag{7.5}$$

Therefore:

$$B' = 180 - B \tag{7.6}$$

The values a and B' are then stored by the IDS entity.

Once the IDS entity has moved, at $t + 1$, the IDS entity knows its own heading and observes its local environment for any other entities. Those that it detects, it compares their locations to the stored locations calculated at time t. Therefore, if there was an entity at a bearing of B' from its own heading and this other entity was at a distance a away from the IDS entity, then the IDS entity would consider the other entity to be stationary and the other entity would therefore be assumed malicious.

The initial simulations were again conducted with malicious swarm entities only, in order to prove the concept. These simulations were conducted with the same operational environment configurations as for the mobile IDS implementation. The differences being that the malicious entities remained stationary and there was no target placed within the operational environment, as there was no searcher to hunt for it. Independent entities were still placed randomly placed to move around within the operational environment, in order to observe any false positives that might occur.

The number of IDS entities were varied, to observe the effects of introducing more to the simulation. The IDS entities operated independently of each other and moved randomly around the operating environment, in a same manner as the independent entities. That is, they would randomly choose a bearing to travel along and a random distance, with an upper bound, to travel along the bearing. Once the IDS entity reached the distance to travel, it repeated the process of randomly choosing a new bearing and randomly choosing a new distance to travel along the bearing.

All of the simulations successfully ran to completion and all of the results followed similar patterns. As would be expected, the greater the number of IDS entities, the quicker the simulation was to complete. This is shown in Figure 7.11.

The simulations were conducted with the same configuration tolerances throughout, regarding the detection angle accuracy and the detection range accuracy. The configuration consistently led to the same pattern of false positives when compared to the number of independent swarm entities. This can be seen in Figure 7.12 and, typically, the amount of false positives detected would be about half the number of independent swarm entities for the simulation. This was regardless of the amount of numbers of IDS entities, the number of malicious entities or the operating environment.

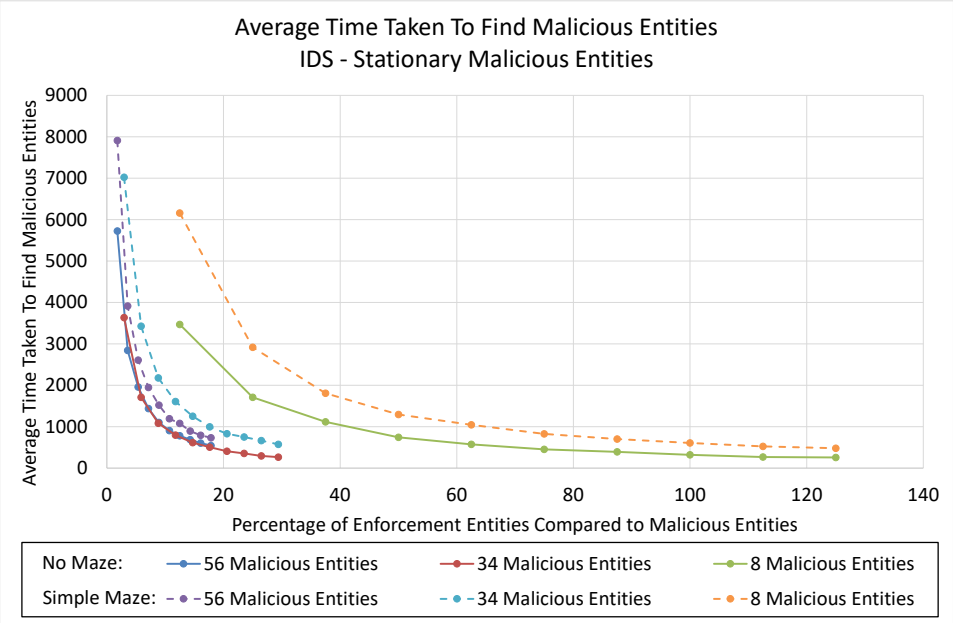


Figure 7.11: Average Times to Locate Malicious Entities

This was not considered to be of concern, as the aim of the simulation was to prove that the concept of IDS could be conducted by observing other entities within the operating environment.

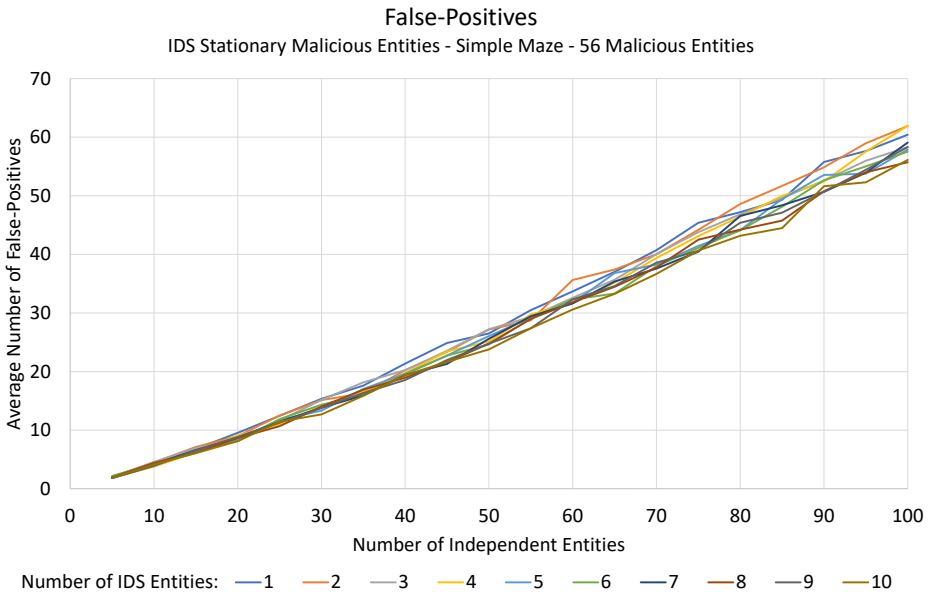


Figure 7.12: False-Positives and Number of Independent Entities

The actual configurations for the IDS simulations, such as the tolerances, could have been tailored further, in order to reduce the false positives. However, the objective was not to design an IDS system for a particular swarm implementation, but to understand if an swarm based IDS system could be successfully undertaken.

7.2.3 Combining the IDS Methods

Once the IDS methods had been shown to work, the IDS entities were then introduced into a swarm that was being attacked by malicious entities. The simulations were configured such that the IDS entities would not remove either the searcher or the target from the operating environment.

The attack method chosen to simulate the IDS swarm against was the Hunter Runner Wall Stop attack, as described in Section 5.2.3.5. The reason for this attack being chosen was that the malicious entities could exhibit either of the two characteristics that the IDS swarm could detect. That is, when a malicious entity detected a target it would move away in a straight line, which the $f_{IDS}(SLDist)$ function would attempt to detect. During the simulations, the searcher entity, malicious entities and the IDS entities would attempt to locate the target. If the malicious entity was first to successfully reach the target first, the searcher entity and IDS entities would then attempt to locate the malicious entity that was broadcasting the best navigational information. This is shown in Figure 7.13.

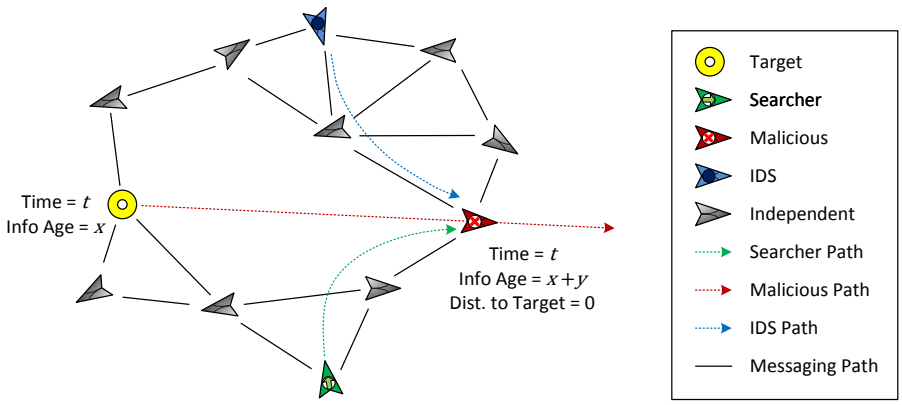


Figure 7.13: Searcher Entity and IDS Entity Navigate Towards Malicious Entity

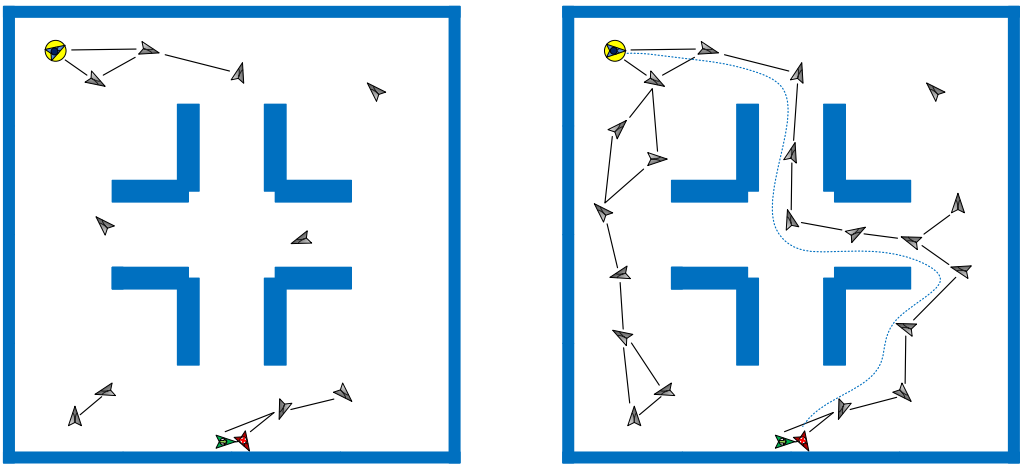
What was also observed within the simulations was that if the IDS entities located the target first, they would essentially wait at the target. If a malicious entity then located the target, the malicious entity would then start to broadcast better navigational information than the target itself.

The IDS entities would therefore leave the target and move towards the malicious entity. The $f_{IDS}(SLDist)$ function was shown to be effective at locating this event within unobstructed operating environments, as a malicious entity could detect a target at one extreme of the operating environment and attempt to move towards the other side of the operating environment. This allowed the malicious entity to move, without locating a wall, for a distance beyond what would be expected within the normal operating characteristics of a swarm entity.

Once a malicious entity, that had discovered a target, had detected an obstacle, such as a wall, it would stop and remain stationary, which the $f_{IDS}(Stationary)$ function would attempt to detect. The $f_{IDS}(Stationary)$ function was observed to work well in operating environments with obstructions.

An observation, from the simulations, was that in operating environments that contained lower numbers of independent entities and lower numbers of IDS entities, it was possible for malicious entities to locate the target and move away from it, without the IDS entities detecting this. The IDS entities would eventually locate the target, and remain there, and the searcher entity would locate the malicious entity and remain there, as shown in Figure 7.14a. This situation would then run indefinitely and the malicious swarm undertook a successful Denial of Service attack on the searcher.

However, as the numbers of independent entities increased, communications paths were observed between the IDS entities and the malicious entities, which would cause the IDS entities to leave the target and move towards the malicious entities, as shown in Figure 7.14b. The blue dashed line shows the most efficient path for the IDS entities to the best navigational information.



(a) IDS Entities "Stuck" **(b)** IDS Entities "Freed"

Figure 7.14: IDS Entities Located at the Target

Another observation, in a similar fashion, that exacerbated the issue of low numbers of independent entities, and therefore no path between the IDS entities located at the target and the searcher located at a malicious entity elsewhere in the operating environment, was when the independent entities had been reduced in number by the IDS swarm incorrectly identifying independent entities as malicious entities and removing them from the operating environment. These false positive results did, on occasions with lower initial numbers of independent entities, remove all of the independent entities from the operating environment.

7.2.4 Independent IDS Simulation Results

Simulations were undertaken that repeated the previous simulations of the Hunter Runner Wall Stop attack on the swarms but with the inclusion of a combined IDS swarm. The numbers of IDS entities ranged from 1 through to 10.

The results from these simulations demonstrated that the combined IDS swarm functionality had performed successfully, detecting both straight line moving malicious entities and detecting stationary malicious entities.

The effect on the performance of the searcher in reaching its goal showed a significant improvement in the searcher's performance. To illustrate this, in the previous simulations, detailed in Section 5.2.3.5, Figure 5.33 shows how, within an operating environment without a maze, a starting separation of 15, with 90 independent entities and 10 malicious entities, the searcher was not able able to achieve its goal. With 9 malicious entities, the searcher was able to fulfil its goal approximately 1% of the time. The most efficient that a searcher could be with 90 independent entities was approximately 22%, with only one malicious entity. When the IDS swarm was utilised, the searcher achieved its goal 100% of the time.

Within the entire IDS simulation for the unobstructed operating environment, the worst result obtained for a searcher achieving the goal of locating a target was 56%, which was for a starting separation of 15, 10 independent entities, 10 malicious entities and one IDS entity. Within this unobstructed operating environment, there were only 21 results where the searcher reached its goal less than 75% of the time, there were 640 results where the searcher achieved between 75% and 95% success in achieving its goal and 658 results were the searcher achieved between 95% and 99% success in achieving its goal. Out of a total of 3600 simulations, the searcher successfully achieved its goal 100% of the time for the remaining 2,281 simulations.

The majority of the lower numbers from the simulations were as expected, when there were lower numbers of IDS entities and independent entities. However, the results did show that the IDS swarm enabled the searchers in achieving their goal more effectively.

7.2.5 Conclusion of Independent IDS Swarm

From the results and observations of the simulations, it is apparent that an independent IDS swarm, that is external to the original swarm, is capable of detecting uncharacteristic behaviour, which can be attributed in malicious behaviours.

In detecting the malicious behaviours, the IDS swarm improved the effectiveness and efficiency of the original swarm in achieving its goal, or even just enabling the original swarm to actually achieve its goal. In this instance, the simulations demonstrated that a previously successful attack on a swarm that utilises cooperative navigation, use-case 1, could be successfully identified and defended against.

7.3 Internal Swarm IDS

In this section, the IDS function is considered when undertaken by the original swarm itself. In these simulations the IDS function was attempting to detect anomalies within the normal characteristics of the swarms operating environment.

The goal of the original swarm is to detect and clear an operating area of landmines [112, 113, 133], utilising a foraging and collective recruitment technique, as in use-case 2. The attacker is attempting to either prevent the original swarm from completing the task or to decrease the efficiency of the original swarm, so as to increase the time taken by the original swarm to complete its task. The goal of the IDS function, within the original swarm entities, is to detect anomalies during its operation that would suggest malicious behaviours. The IDS functionality is attempting to identify attacks that are being undertaken against a swarm that is utilising foraging techniques and local recruitment methods.

These attacks have been previously described in Section 5.3.3.2, by utilising false landmines, and Section 5.3.3.3, by generating a false scent. As previously discussed, the attacks can prevent the original swarm from completing its task of locating and clearing all the landmines.

The two attacks are considered, along with the IDS functions that detect and then mitigate the attacks. As the IDS function is undertaken within the original swarm entity, the original swarm entity is able to alter its current operational state. The original state, as described in Section 5.3.2 are: Foraging, Following and Waiting. The IDS function introduced another state, which was entitled "Move Away" and can be seen in Figure 7.15. The Move Away state puts the original swarm entity into a state where it moves away from its current location, until there is no scent detected beneath it. This ensues that the original swarm entity has removed itself from any influences within the operating environment and, whilst in this state, the original swarm entity does not measure the scent density or attempt to locate any landmines. This allows the original swarm entity to move away from any malicious influence, due to scent gradients. Once the original swarm entity has moved away from any influence of scent, it will change its state back to Foraging.

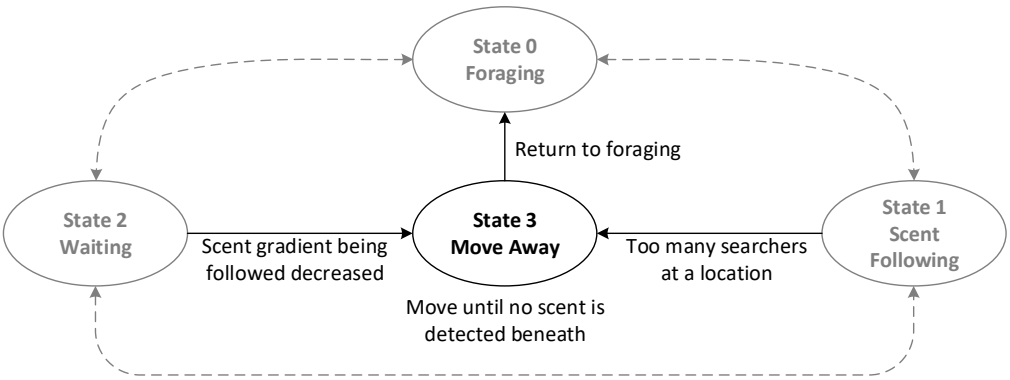


Figure 7.15: Extra IDS State

Simulations for IDS implementations were then conducted on the two potential attacks.

7.3.1 False Landmine Detection

In this attack, false landmines are located within the operating environment, along with the actual landmines.

The IDS function proposed for this attack is based on the knowledge of the operating environment and the original swarm entities operating principles:

- the original swarm entity will enter a Waiting state if it detects a landmine directly beneath it.
- the environment will disarm a landmine if 4, or more, original swarm entities are located directly upon the landmine.

Therefore, as false landmines were used, no mater how many original swarm entities were located at the false landmine, they would remain at that location. That is, there is no actual landmine to disarm, so the original swarm entities do not move away.

In this instance, the advantage for the attacker is that the attacker does not require any resources, other than the false landmines. The scent production is carried out by the original swarm entities, as they believe that they have found a genuine landmine and have entered a Waiting state and are thus releasing scent. This action, in turn, attracts more original swarm entities.

Utilising this knowledge, the IDS was implemented such that the original swarm entities were able to detect and count the presence of other original swarm entities at a location. The ability to detect a malicious attack is then quite simple. That is, if there any more than 4 original swarm entities are located at a particular location, then assume a malicious attack.

$$f_{IDS}(s_i^o Qty) = \begin{cases} FALSE & \{s_i^o Qty \in \mathbb{W} \mid 0 \leq s_i^o Qty < 4\} \\ TRUE & \{s_i^o Qty \in \mathbb{W} \mid s_i^o Qty \geq 4\} \end{cases} \tag{7.7}$$

Where:

$f_{IDS}(S_i^o Qty)$ - IDS function being carried out by swarm entity relating to quantity of landmines at its location.

$S_i^o Qty$ - The total quantity of swarm entities at a specific location.

The implementation on the original swarm entities is therefore to count the number of other original swarm entities at their location. If there are more than 3 other original swarm entities at the location, then assume malicious activity and change the operating state to a Moving Away. As detailed in Section 5.3.2 the number of swarm entities required to disarm the landmine in the simulation is an arbitrary number as 4 was chosen so as to reproduce the research undertaken by Kumar, Sahin and Chapman. All original swarm entities should realise the malicious activity at the same time and they should all change their state to Moving Away. They will then move away until there is no scent beneath them, when they will return to a Foraging state.

There was also consideration given to making the swarm entity move away a predefined distance away from the false landmine, in order to remove the immediate possibility of any influence from any other original swarm entities that might have located the false landmine whilst the initial original swarm entities are moving away, or in case the swarm entity changed direction and returned to the landmine.

This IDS implementation does not interact with the mechanism of the attack, such as by attempting to destroy the false landmines. Instead, the IDS function detects the malicious attack and then alters the original swarm entity's behaviour, by altering its state, in order to remove the influence of the malicious attack.

The simulation results showed that, where previous attacks had prevented the original swarm from completing its goal of successfully locating and disarming all of the landmines within the operating environment, the introduction of the IDS had enabled the goal to be successfully achieved. That is, during the original attack, the false landmines had essentially trapped the original swarm entities, the IDS function had prevented this from happening.

When there were 4 malicious mines placed in the operating environment, the introduction of the IDS techniques enabled the swarm to achieve its goal of locating and disarming the landmines. The times taken to successfully complete the goal of locating and disarming all of the landmines within the operating environment was comparable to the baseline simulations, for a benign operating environment.

When there were nine malicious mines placed in the operating environment, the original swarm was able to successfully achieve its goal and locate all the landmines. The amount of time taken to achieve this did tend to take longer, the percentage comparisons of comparing the time taken to the original baseline simulation results can be seen in Figure 7.16. Although the simulation showed that the task to locate and disarm the landmines did take longer when there are 9 malicious landmines, the task was, however, successfully completed.

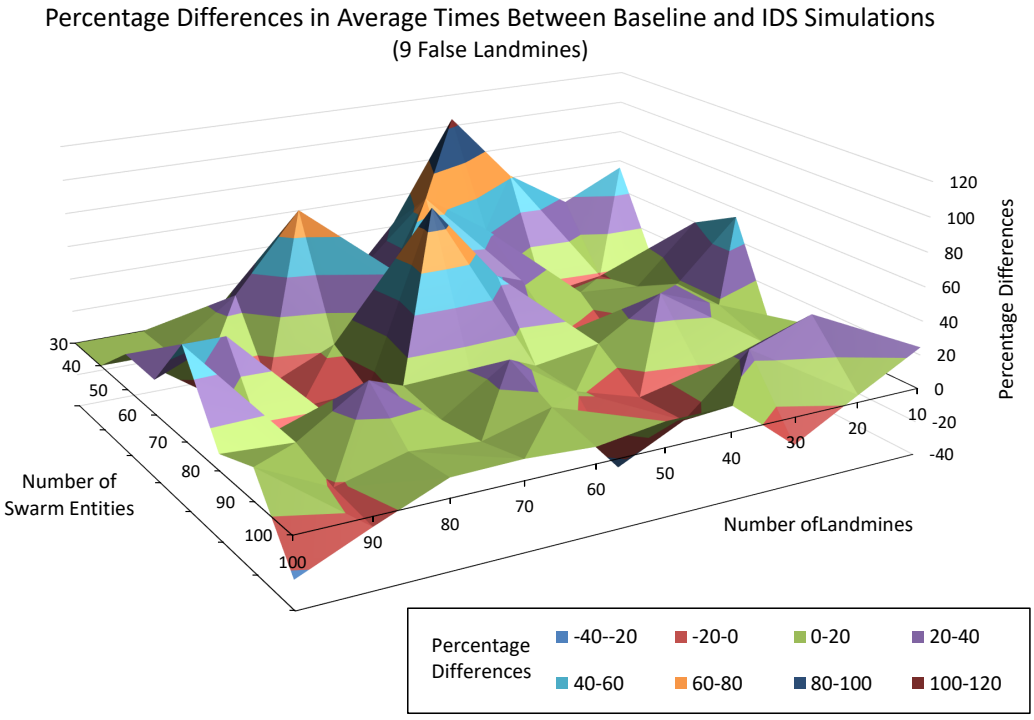


Figure 7.16: Percentage Difference in Average Times for an IDS Enabled Swarm Compared to a Baseline Swarm

It was also noted, when observing the simulations, that there was a large amount of false-positives, suspected false landmine when there was none present, upon the initial release of the original swarm. This was demonstrated by the large amount of swarm entities entering a Move Away state. This was not unexpected and was due to the fact that the entire original swarm was always released at the same point. The original swarm entities would then move away to their randomly chosen locations. During the initial period, immediately after the swarm release, there was often the case that there would be more than 4 original swarm entities at a particular location. This was due to the density of original swarm entities within the operating environment and these false-positives were expected. Depending upon the implementation of swarms, the original swarm entities could be released in a Move Away state for a period of time or distance, in order to allow the original swarm entities to move away and disperse. However, this would then require the original swarm entities to return to their start point, to ensure all landmines are located that might have been missed, during the initial deployment.

The distribution of the swarm's states during the swarm's initial release period can be seen in Figure 7.17 and shows the initial spike in swarm entities entering a Move Away state, due to the false-positives. Once the swarm entities began to disperse, the number of swarm entities within the Move Away state can be seen to reduce and the amount of swarm entities in a Foraging state increase.

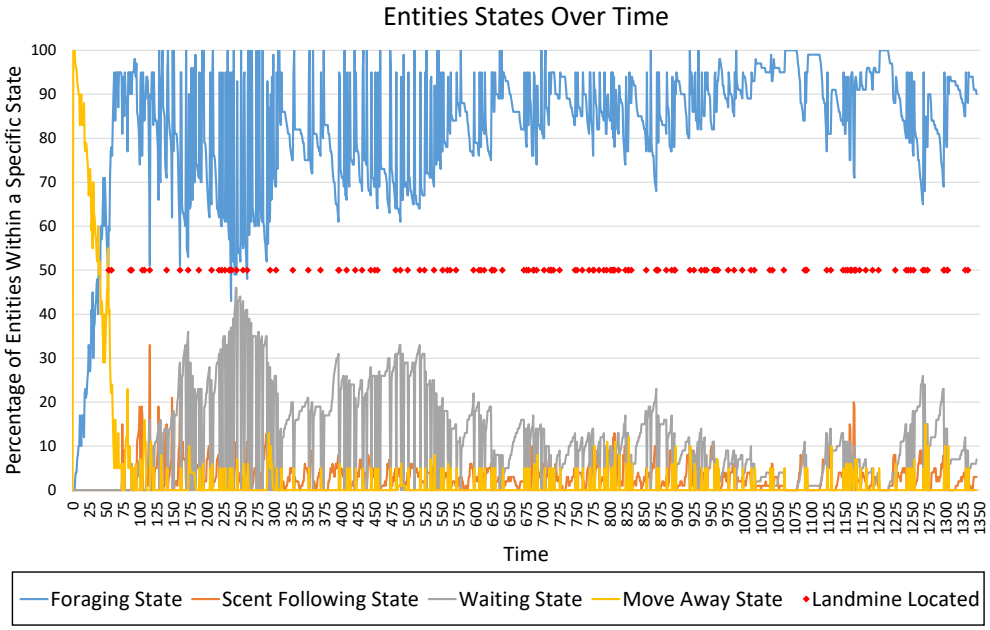


Figure 7.17: Entities' States Within a Swarm Over Time

Again, the disadvantage is that the original swarm entities would need extra resources in order to achieve the IDS capability. In this instance this would be the sensing capability in order to detect the other original swarm entities. There would also be processing and storage requirements, in order to maintain knowledge of the amount of other original swarm entities within the same local operating environment.

7.3.2 False Scent Generation Detection

In these attacks, false scent generators were positioned within the operating environment and landmines were then randomly distributed throughout the operating environment.

The IDS function proposed for this attack was based on the knowledge that the initial implementation of the original swarm entity, in that:

- the original swarm entity will only stop on a landmine.
- an original swarm entity will always move to the location with the greatest scent density.

Therefore, if an original swarm entity located a false scent generator, the original swarm entity would move continuously around the centre of the false scent generator, as described in Section 5.3.3.3.

In order to detect this attack, the original swarm entity was modified to allow it to store the current scent density value. Therefore, if the only movement options available to the original swarm entity would result in a lower scent density, when compared to its current scent density, then the original swarm entity would believe that this was due to malicious activity. The original swarm entity would then change its operating state to "Moving Away". This state will allow the original swarm entity to move away from the scent, produced by the false scent generator, and then change state to a Foraging state and continue with the task of locating and disarming the landmines within the operational environment.

$$f_{IDS}(Scent) = \begin{cases} FALSE & \{ScentDetected > ScentStored\} \\ TRUE & \{ScentDetected < ScentStored\} \end{cases} \tag{7.8}$$

Where:

$f_{IDS}(Scent)$ - IDS function being carried swarm entity based on detected scent density.

ScentDetected - The maximum scent density value that s_i^o is able to move towards.

ScentStored - The current scent density value for the current location of s_i^o.

Simulations were successfully undertaken. Where previous simulations had failed to achieve the goal of locating and disarming all the landmines, as the original swarm entities had become "trapped" within the influence of the false scent generators, these simulations ran to the successful completion of the goal.

However, it should be noted that the time taken in order to achieve a successful goal was increased, when compared to no malicious activity within a benign operating environment. The time was significant and the completion times could be considered to be too large to be considered practical.

The reason for the increase in time taken to undertake the task to a successful conclusion, when compared to an operating environment without malicious attackers, was due to the fact that the original swarm entities were, at times, in the Moving Away state and therefore not Foraging.

This implementation of an swarm IDS makes no attempt to influence are interact with the actual mechanism of attack. Instead, the original swarm entity detects the attack and then modifies its behaviour, so as to remove itself from the influence of the attack.

The disadvantage of this IDS is that extra resources are required within the original swarm entity. In the original implementation, the original swarm entity only had to move towards the greatest scent density, information which might have been able to have been directly provided by its sensors. Now the original swarm entity also has to take a measurement from a sensor and record this value.

A characteristic of the original swarm's operation that was observed during the simulations, which was also discussed in the original papers describing the swarm proposal [112, 113, 133], was when an original swam entity would enter a local scent density maximum. This was due to an original swarm entity attempting to follow a scent gradient to a landmine. However, other entities of the original swarm have located mines in close proximity and their scents are overlapping and therefore generating a local scent density maximum. The original entity therefore becomes stuck at this location, as all other scent densities around it are lower than its current location. Therefore, whichever way it moves, it will always return to the centre location, as detailed in Section 5.3.3.3. However, as the swarm entity perceives this as a malicious activity, the swarm entity changes its state to Moving Away and the swarm entity no longer becomes frozen.

7.3.3 Additional Measures to Improve Performance

There was the issue of when the quantity of deployed original swarm entities is such that there are potentially not enough original swarm entities to successfully complete the goal of successfully locating and disarming all of the landmines within the operating environment, described as a "frozen" event. The frozen concept was described in Section 5.3.3.1

To prevent this from happening, a Timeout function was implemented that would ensure that the original swarm entities would only maintain a Waiting state for a maximum time limit. Once the maximum time limit was reached, the original swarm entity would enter a Moving Away state, to allow the original swarm entity to move far enough away from the landmine to lower the chances of the original swarm entity from returning back to landmine it had just left and therefore improve its chances of locating a different landmine.

$$f_{Timeout}(s_i^o Time_{waiting}) = \begin{cases} FALSE & \{0 \leq s_i^o Time_{waiting} \leq t_{max}\} \\ TRUE & \{s_i^o Time_{waiting} > t_{max}\} \end{cases} \tag{7.9}$$

Where:

$f_{Timeout}(s_i^o Time_{waiting})$ - Function to determine the amount of time a swarm entity has been in a Waiting state.

$s_i^o Time_{waiting}$ - Amount of time s_i^o has been in a Waiting state.

t_{max} - A variable that determined the maximum amount of time s_i^o should remain in a Waiting state.

As the Timeout function was based on the Waiting state of the original swarm entity, the Timeout function moved the original swarm entities away from both false landmines and genuine landmines. Simulations were observed where a number of swarm entities were in a waiting state at a genuine landmine, with other swarm entities in a Foraging state within the operating environment. However, as the other swarm entities did not come close enough to detect the scent and enter a Scent Following state, the swarm entities in the Wait state at the landmine would timeout and enter the Moving Away state.

7.3.4 Conclusion of Internal Swarm IDS

From the results and observations of the simulations, it is apparent that an IDS, which is internal to the swarm entities, is capable of detecting uncharacteristic behaviour that can be attributed in malicious behaviours. In this instance these were characteristics of a swarm entity's local environment, when attempting to make landmines safe by foraging and local recruitment methods, use-case 2. The original swarm is then capable of undertaking simple actions, by modifying its operating state, in order to counter the malicious attacks.

The efficiency of the swarm is reduced, when compared to a benign environment, but this would be expected. The trade-off is that the original swarm is now capable of completing its goal, which it was not able to do when subjected to malicious attack without an IDS function.

It is also apparent that the false scent attack, which covered a reasonable proportion of the operating environment, 3.24% or 7.29%, and was essentially a constant attack, as the false scent was always present, was a much more effective attack against this implementation of a swarm and its associated IDS implementations.

7.4 Discussion

The simulations of both external IDS swarms and IDS techniques that are internal to swarm entities showed, that through the simulations of the use-case scenarios, that the IDS functions could be implemented successfully.

The IDS techniques that were implemented were signature based, with the characteristics for normal operation being known through the swarm's design. From knowing the design of the swarm, signatures could be determined that would indicate if a swarm entity was operating outside of its expected operating characteristics, which would therefore indicate the suspicion of malicious activity. If an IDS function did determine that malicious activity could be being undertaken, this would result in an action being carried out, so as to allow the original swarm to continue in its operation. That is, the signatures for malicious activities were essentially when the swarm entities operated outside of pre-determined specified limits.

In order to realise the characteristics of a swarm, the IDS functions were simple in both their construct and implementation. To ensure a realistic approach was simulated, all IDS functions were simulated from the perspective of the swarm entity and not the swarm as a whole.

Although the simulations of the attacks and the simulations for the IDS implementations were tailored for the specific swarm implementations and their operating environments, the IDS implementations were based on the generic IDS model, as presented in Chapter 6. The two use-cases were simulated, based on the generic IDS model, to demonstrate that the principles of a signature based IDS implementation could be applied and were both achievable and able to defend a swarm, which is summarised in Chapter 8.

Part III

Conclusions

8 Discussion

8.1 Introduction

The initial purpose of this research was to gain an understanding of whether swarms could be maliciously manipulated, utilising the unique characteristics of swarms.

As it was shown that swarms could be maliciously manipulated, the main focus of the research was then conducted, to understand if swarms could detect the malicious activity.

This research was interested in swarm implementations that were constrained with their capabilities, as discussed in Section 2.3, such that intensive processes or operations could not be undertaken by the individual swarm entities. This research also constrained the swarm implementations to the capabilities and limitations, as described in Section 1.2.

The swarms that were simulated were taken from previous research literature and were initially simulated within benign conditions, in order to replicate the original researchers' results. The swarms chosen were such that they represented different types of swarm, when viewed from the taxonomies detailed in Chapter 4, and different attacks were conducted, as described in Chapter 5, allowing for the realisation of different threats.

The swarms chosen included cooperative navigation techniques, in order to locate a target, and foraging and local recruitment schemes, in order to locate and make safe landmines. This enabled the research to review swarms which had different tasks and implementations, in order to achieve their goals.

These swarm proposals were discussed through this thesis as use-case 1, cooperative navigation, and use-case 2, foraging and local recruitment schemes.

8.2 Manipulating a Swarm

Based on the implementations of the swarms, various attacks were undertaken in order to maliciously manipulate the swarms. The aim of the attacker was to realise a threat, in order to allow an attack to be undertaken. The ultimate aim of an attack was to utilise the unique characteristics of a swarm, as discussed in Section 1.2, to the attacker's advantage.

Several attacks were then successfully conducted against the swarms. The simulations showed that swarms could be maliciously manipulated, in order to either prevent the swarm from achieving its goal, or to reduce the efficiency of the swarm in undertaking its task to achieve its goal.

8.2.1 Case 1 - Attacking Cooperative Navigation

The research I undertook on attacking cooperative navigation was in order to consider attacks against use-case 1, a swarm tasked with locating a target utilising cooperative navigation.

The simulations of the attacks were conducted, so as to be comparable with the original research, where the operating environment could be either empty, have a simple maze or have a complex maze.

Initially, the attacks conducted against the swarm were essentially traditional jamming attacks, that could also be undertaken against other systems. Several versions of the jamming attack were undertaken, these being where the jammers were initially stationary and then they were mobile, within the operating environment, and whether the jamming entities were visible or not to the original swarm entities.

These attacks were conducted in order to observe effects upon the swarm and to further verify the implementation of the swarm simulations.

What was interesting about these attacks was that the attacks had very little effect on the performance of the swarm. Indeed, if the malicious enteritis were visible to the the original swarm, then the original swarm's efficiency at locating a target could actually improve, as now there were more overall entities within the operating environment. Also, by the very nature of a swarm implementation, if one information path is prevented, by the actions of jamming, then other information paths will often still be available that are not effected by the jamming.

This demonstrated how swarms could be robust to external effects.

Following the jamming attacks, attacks were then undertaken that utilised the unique characteristics of a swarm. The main characteristic of the swarm that was targetted, was the swarm's decentralised control, swarm characteristic 2, and the subsequent emergent behaviour, swarm characteristic 4. Knowing how the swarm entities interacted with each other, within a local environment, malicious swarm entities were introduced that also interacted with the original swarm entities. This was achieved by the malicious swarm entities realising the threat of masquerade, as the original swarm entities interacted with the malicious swarm entities as if the malicious swarm entities were part of the original swarm. This then allowed the malicious swarm entities to carry out an attack that utilised mis-information, by communicating incorrect information to the original swarm entities.

Various attacks were simulated, from a simple random walker that presented better navigational information, to more subtle attacks that tailored the moment that better navigational information was presented to the original swarm.

All of the attacks undertaken, that utilised the swarm's characteristics, decreased the efficiency of the original swarm from achieving its goal. This could be either by increasing the amount of time a swarm takes to achieve its goal, or the decreasing the likelihood of a swarm achieving its goal.

8.2.2 Case 2 - Attacking Foraging and Local Recruitment

The attack was undertaken against a foraging and local recruitment technique that was used to locate and then make safe landmines, use-case 2. When a swarm entity locates a landmine, then the entity will attempt to recruit other swarm entities, in order to gain a sufficient quantity of swarm entities to make the landmine safe. Recruitment was carried out by releasing a scent, which formed a scent gradient that peaked at the location of the landmine. Other swarm entities, that located the scent, followed the scent gradient until the swarm entity located the landmine, where it would also release a scent, reinforcing the original scent gradient.

There were therefore two obvious attacks that could be carried out, that involved manipulating the swarm's operating environment. These were where an attacker could either masquerade as landmine or the attacker could release a false scent. Both attacks aimed to deceive and then locally recruit entities of the original swarm into a trap.

The advantage of masquerading as a landmine is that the attacker only has to deploy false landmines. The attacker essentially uses the scent resources of the original swarm against itself. The disadvantage is that the attacker has to produce a false landmine that the original swarm will believe is a genuine landmine when it is foraging.

The advantage of generating false information by releasing a scent is that an attacker can attract original swarm entities from a larger area than initially just the location of a landmine. Essentially, the attacker can cover an area as large as the scent can disperse within, such that the scent can still be detected by an original swarm entity. However, the disadvantage for the attacker is that the attacker has to produce a scent that the the original swarm will recognise and the attacker might be resource constrained on the amount of scent available for a prolonged attack. This could lead to a time limited attack against the original swarm.

Both attacks proved to be successful, to the extent that the swarm was generally denied the ability to complete its task. The attacks would either make the task of disarming all the landmines within the operational area less efficient, as the task took longer to achieve the goal, or, more often, the goal was not achieved at all.

8.2.3 Summary of Attacking a Swarm

The simulations that were undertaken against different implementations of swarms, from both use-case 1 and use-case 2, showed that it is possible to successfully undertake an attack against a swarm and its associated swarm entities. The simulations also demonstrated that the unique characteristics of the swarm could be utilised in order to assist in undertaking an attack, therefore making the attacks unique to swarms.

8.3 IDS within a Swarm

Following on from the attacks against the various swarms, the main objective of the research was undertaken. This was to identify if it was possible for a swarm entity to detect if the swarm was being subject to malicious activity, which was attempting to manipulate the swarm.

A review of current Intrusion Detection System techniques was undertake and the principles of *signature* based IDS was taken forward.

Importantly, *anomaly* based IDS techniques were discounted as an option, due to the unique characteristics of a swarm and the constantly varying environments in which swarms operate.

Knowing how the swarm should operate, it was possible to generate signatures for abnormal behaviours. This was carried out on all the original swarm configurations and operating environments, as well as for all of the previous attacks that utilised the characteristics of a swarm to aid with the attack. The IDS operation was also designed to be as simple as possible, due to the resource constraints within a swarm entity implementation.

The implementation of the IDS was simulated externally to the original swarm, utilising an IDS swarm, and internal to the original swarm, using the original swarm entities themselves. All of the simulations were undertaken from the perspective of a swarm entity. That is, the simulations were not undertaken by observing all of the swarm entities, and the entire operational environment, but from the perspective of the individual swarm entities. All detections, calculations and actions were undertaken by the individual swarm entities.

8.3.1 IDS within Cooperative Navigation

It was decided not to attempt to perform intrusion detection against a jamming attack. This was because the attack had little effect on the swarm and there are already techniques that can detect jamming attacks. The focus of this research was to detect attacks that are attempting to utilise the unique characteristics of a swarm.

The research focused on attempting to identify behaviours that, although possible by a swarm entity, were uncharacteristic for normal operation. The characteristics that were chosen to be investigated were to attempt to detect stationary swarm entities and to detect mobile swarm entities, in which a swarm entity's movements were undertaken in a way that did not meet the swarm's normal operating characteristics.

Initial baseline simulations were conducted, to prove the concept of the IDS functions within a swarm. These simulations included several operating environments and various numbers of malicious entities.

The results showed that, although there were both false-positives and false-negatives, the malicious swarm entities could be successfully detected. As the results of the IDS initial baseline simulations were successful, simulations were then undertaken of the swarms that had previously been successfully attacked but now with the addition of an IDS function within the swarm. The IDS function being a combination of the two baseline IDS configurations. The results quite clearly demonstrated the success and effectiveness of the IDS in this combination of swarm, operating environment and IDS implementation.

Various actions were undertaken when a malicious activity was suspected but the actions effectively removed swarm entities from the operating environment.

8.3.2 IDS within Foraging and Local Recruitment

The IDS implementations within the foraging and local recruitment simulations were undertaken separately for the two different attacks, in order to attempt to detect and defeat the different attacks.

The IDS implementations effectively added an extra operating state to the swarm entities and the swarm entities attempted to detect anomalies within the operating environment.

If a swarm entity perceived that the operating environment appeared to be uncharacteristic, then the swarm entity change to the extra *Move Away* state, which allowed the swarm entity the ability to remove itself from the attack. The effectiveness of the IDS was dependent upon the attack that was being undertaken.

The simulation results demonstrated a marked difference in IDS performance for the two different attacks, depending on how the operating environment was being manipulated.

Although the IDS function for the false scent attack did work successfully, the coverage area and continuous nature of the attack meant that an entity would free itself from one attack location and then often move directly into another attack location. This meant that the IDS had a minimal impact against the malicious attack.

The IDS function proved to work well against the false landmine attack, turning the situation from all the swarm entities becoming trapped at the false landmines to a result where the robotic swarm achieves its goal.

Depending on the amount of attackers present in the operational environment, the swarm can accomplish its goals in a similar time period to that of an operating environment with no malicious entries within it.

8.3.3 Summary of IDS within a Swarm

The simulations undertaken on the user-cases showed that the individual swarm entities were capable of detecting what was believed to be malicious activity. That is, the simulations undertook the IDS functionality from the perspective of individual swarm entities and not from the perspective of the swarm as a whole.

The use-cases demonstrated how signature based IDS schemes could be successfully utilised within swarms.

At times, there could be either false-positives or false-negatives. However, the introduction of the capability to detect what was believed to be malicious activity, and then respond to this, enabled all the simulations to successfully achieve the swarm's goal. However, even though the swarm entities did achieve their goal, certain attacks had such an effect on the swarm entities, that the attack made this time so great, that the IDS was effectively impractical.

When a swarm did complete its goal, the overall efficiency of a swarm achieving that goal was lower than when the swarm operated within a benign environment. This was because the attacks were still being undertaken and realised against the original swarms. However, the IDS techniques employed did reduce the impact of the attacks, to such an extent as to actually enable the swarm to achieve its goal where previously it had not been able to do so.

8.4 Future Work

Proposed future work within this research area would be to consider what actions should be taken by a swarm, once malicious activity is suspected or detected. Actions that could be undertaken if malicious activity is suspected could be to ignore the attacker; attack, or "kill", the attacker; monitor the attack and the malicious entities undertaking the attack; or the swarm could alter its behaviours. These proposals are detailed as follows:

Ignore the attacker
So long as the attacker does not have an affect upon the robotic swarm entities that are local to the attacker.
This could be difficult, if not impossible, to achieve, due to the physical presence of an attacker. That is, the physical presence alone might be enough to alter the emergent behaviour of a robotic swarm, such as the robotic swarm attempting to physically avoid an attacker essentially causes an interaction between the robotic swarm and the attacker. This could then affect the emergent behaviour of the robotic swarm in a way that is unavoidable.

Attack the attacker
Prosecute the attacker in such as way as it can no longer affect the robotic swarm.
The principle behind this is similar in how nature can deal with an attack on a swarm. For example, if a bee hive is attacked, a bee, local to the attack, will produce pheromones to alert and attract other bees to the attack on the bee hive, in order to defend the bee hive. Other bees react to the pheromone and also join in with the attack on the attacker, in the defence of the bee hive. The military also use similar techniques, if an adversary is using radar to track friendly forces, then the friendly forces can prosecute the enemy radar using anti-radiation missiles, designed specifically for the role of attacking radar emitters.

Observe the attacker
In order to gain intelligence about how an adversary conducts an attack.
It is often beneficial to understand an enemy's strategy and tactics in how they attempt to attack a system, in order to design suitable defences and actions against future attacks. The observations can be obtained from various sources. Network operations centres can attempt to follow an attack or incident in real time and system logs can be reviewed post incident. The observations can take place on a real system, a test system or a fake system, where the fake system is intended to attract and lure an attacker into conducting an attack that can then be observed. These systems are often refereed to as "honey-pots".

Alter the behaviour of the robotic swarm entity being attacked
A robotic swarm entity that perceived it was under attack could alter its behaviour, in order to attempt to defeat an attack. If the robotic swarms entity's actions were uncharacteristic it could change its behaviour, such as it could stop following another robotic swarm entity that it believed to be malicious, or it could stop forwarding received messages. It might stop forwarding messages because the robotic swarm entity did not trust the content of the message, or it believed that the message had originated from a malicious source. A level of caution would need to be exercised though, as this could also prevent genuine messages from being forwarded. Also, if this defensive behaviour was known by an attacker, then it could possibly be used to an attacker's advantage, in order to counter the swarm entity's IDS.

Confidence Thresholds
Another area of future research would be the consideration of *confidence thresholds* within the swarm entities, in order to determine if an attack likely or is being undertaken against the swarm entities. Typical examples could be for a swarm entity to observe all of its local entities and ignore information from swarm entities where their information appears to be significantly different from the local environment as a whole. Effectively ignore the edge cases if they are too different from the majority.

Confidence values could also relate to swarm entity movement, as well as the information transmitted with the swarm.

An example of this could be that if a swarm entity has been heading in a particular direction and so had all of the other swarm entities that it could observe. Then, one of the other swarm entities changed direction it could take this into account in its confidence that it should alter its own heading accordingly.

This is best described by an example. Swarm entity A is the swarm entity being considered in the example shown in Figure 8.1.

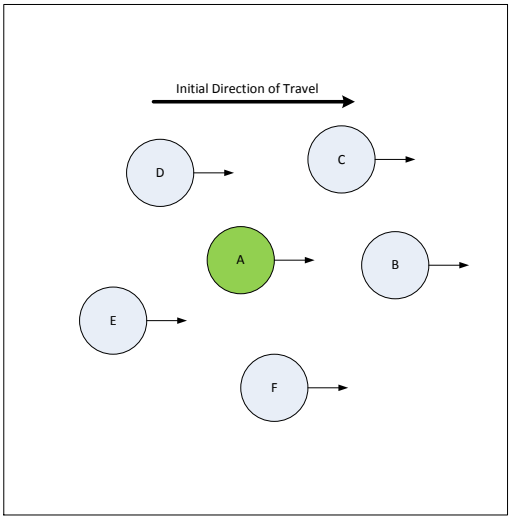
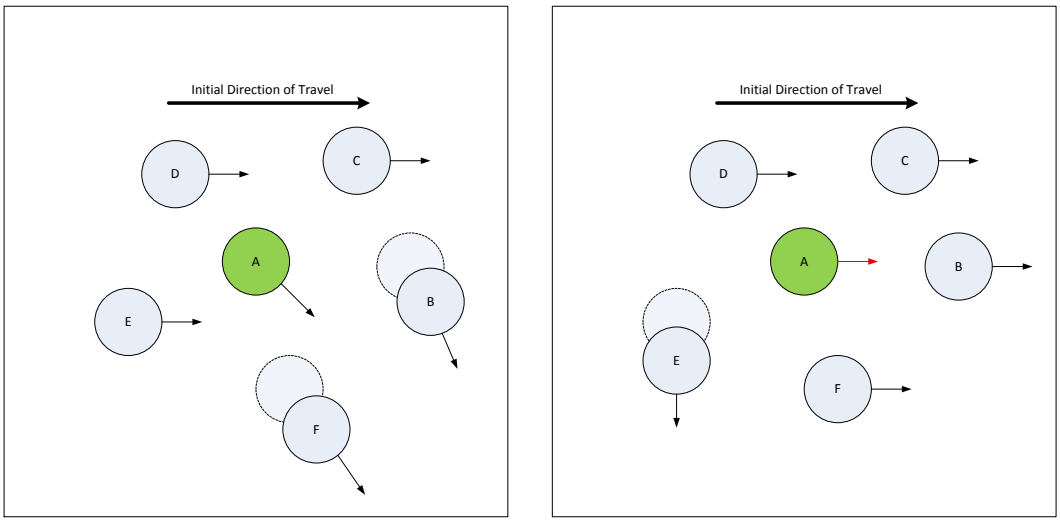


Figure 8.1: Initial Swarm Conditions

The confidence an entity has with the actions of local entities is shown in Figure 8.2. If the leading swarm entities, B and F, changed direction, as shown in Figure 8.2a, then the confidence that this is a required change in direction is high and therefore swarm entity A also changes direction accordingly.

However, if swarm entity E changed direction, as shown in Figure 8.2b, then there is less confidence and swarm entity A does not change direction.

If swarm entity C were to change direction, as shown in Figure 8.3, this could be a valid change in direction. However, as swarm entities D and B have not changed direction, then swarm entity A is less confident in the change in direction. Swarm entity A therefore decides to alter its course but uses a weighting factor to alter its direction accordingly.



(a) Entity With High Confidence **(b)** Entity With Low Confidence

Figure 8.2: Confidences in Swarm Actions

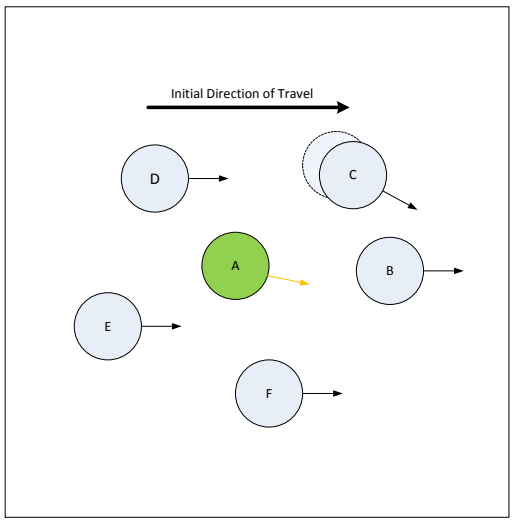


Figure 8.3: Entity with Weighted Confidence

It should be noted that the swarm entities would generally attempt to prevent a collision, in order to enable self-preservation. Therefore, if swarm entity C changed direction such that it would collide with swarm entity A, then swarm entity A would alter course accordingly and, if necessary, without any weighting factor. Essentially the swarm entity's response will be appropriate to the situation and could be considered similar to the artificial potential fields that can be utilised to repulse an object, as described in the underwater mine countermeasure proposed use for a swarm in Section 2.7.6.

Annexes

A Intrusion Detection Systems

A.1 Introduction

The aim of this annex is to supplement the information provided in Chapter 6 and provide the reader with a more detailed overview of Intrusion Detection Systems (IDS) and the current techniques that are utilised within existing technologies and systems. The annex will also provide an overview of the background and history of the subject.

A.1.1 Overview and Background of Intrusion Detection Systems

Intrusion detection systems are essentially attempting to automatically detect an activity that should not be taking place and then report upon this event.

To place this into context, within the physical world a house or car alarm is a form of intrusion detection system. When activated, the systems are attempting to detect the presence of an intruder and, upon detection of a suspected intruder, the system will generate an alert, in order to notify others of this. This could be as simple as sounding an alarm on the house or vehicle, in order to attract attention. The response could also be to send a message to the owner of the house or to law enforcement, which could be via a third party control centre, to inform them of the event. On larger systems, such as IDS systems on remote or industrial premises, the system will also log the events. This allows for post incident analysis.

Within computer and network domains, IDS systems operate within a similar context. That is, extra devices within a network, or software applications within system components, are attempting to detect intrusions within the networks and systems. If the IDS suspects anomalous behaviour, it will log the event and provide an alert to a user.

The objectives of an IDS will influence the techniques that are used, in order to undertake the IDS task. That is, if the purpose of the IDS is to detect intrusions within a network, the network designer might decide to place the IDS as a separate component on the network, separate from the operational systems. This allows the IDS system to monitor the network traffic, without affecting the performance of the individual operational systems. If the IDS is configured to only monitor the traffic on the network, it is less likely to be being detected by an attacker that is attempting to undertaking the intrusion. When placing the IDS within individual systems, such as computer end points, the IDS is able to detect attacks, that an IDS placed on the network would not.

There are various techniques that are utilised with IDS systems, which are described in more detail.

When an intrusion event is suspected, the actions undertaken by the network and computer based systems are similar, in context, to the actions undertaken by a physical IDS. That is, if the IDS suspects it is being infiltrated, it will log the event and report it. If the IDS is local to a system, say a PC, then it might also alarm locally, such as by displaying a warning to a user.

Conceptually, the actions undertaken by either a physical, network or computer IDS are essentially the same. Log the event and inform interested parties.

To further assist the reader, NIST defines intrusion detection as "the process of monitoring the events occurring in a computer system or network and analysing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices" [168] and intrusion has been defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [84].

A.1.2 Background and history of Intrusion Detection Systems

The remainder of this annex will only discuss IDS in the context of computer systems, both networks and system components, such as end points. Therefore, the goal of an IDS is to identify possible incidents, by automating the monitoring of computer systems. This is in order to detect possible actions that could attempt to compromise the confidentiality, integrity or availability of the systems. An IDS can also be utilised to not only detect an incident but to also detect reconnaissance, which could indicate that an attack is immanent [168].

It should be noted that an IDS has the potential to either incorrectly identify a benign activity as hostile, known as a *false positive*, or it fails to recognise a malicious activity, known as a *false negative*.

An IDS is often tuned in order to reduce false positives, with the effect of increasing false negatives. Intrusion prevention is the attempt to prevent incidents that have been detected by intrusion detection from being realised.

A.1.3 History of Computer Intrusion Detection Systems

Originally, James Anderson proposed how to improve computer security auditing and surveillance capacity in 1980 [7]. He suggested that security audit trails on computer systems were important. However, he suggested that they would benefit from being supplemented with a security monitoring surveillance system. This was because he felt that the security audit files were not necessarily well structured, or that they did not contain enough detailed information.

A model for an IDS was first proposed and presented by Dr Dorothy Denning [50, 51] in 1985, and was regarded has a rule based pattern matching system. The model was entitled "Intrusion Detection Expert System" and consisted of six main components:

- **Subjects** Initiators of activity, normally the users
- **Objects** Resources managed by the system, such as files and commands
- **Audit Records** Records generated by the target system in response to performed actions
- **Profiles** Characteristics of the behaviour of subjects
- **Anomaly Records** Records generated due to the detection of abnormal behaviour
- **Activity Rules** Actions taken when a condition is realised

Within the proposal, an audit record is generated from an action being undertaken. The generated audit record is then compared against known profiles. The auditing can be generated at several points during an undertaking of a command, or the auditing can just be undertaken at the end, or completion, of a command.

To place this into context, consider copying a file. The audit log could record all actions involved with undertaking the copy command. This could consist of: execute the copy command, read the original file, write the new file and record action complete. Or, it could just record the fact that the task was successfully undertaken.

The advantage of the former approach, is that it provides an operator reviewing the logs an entire timeline of actions, which could be useful if a system failed part way through a series of actions. The operator would therefore be able to undertake detailed investigations in an attempt to ascertain what had caused the failure. The advantage of the later approach, where only record is that the task has been successfully undertaken is recorded, is that it is quicker and utilises less resources, such as processing and storage.

In 1991, Kathleen A. Jackson, David H. DuBois and Cathy A. Stallings proposed the Network Anomaly Detection and Intrusion Reporter (NADIR) project within their paper "A Phased Approach to Network Intrusion Detection" [93]. Within this work, they demonstrated the feasibility of undertaking automating security auditing, within a distributed environment. Their work demonstrated the benefits of utilising "expert systems", which exploited the information from multiple hosts by aggregating the derived information. This aided in the detection of coordinated attacks, across an integrated computer network.

Their proposed system incorporated user profiles. These were were initially generated and then regularly modified, in order to ensure that the user profiles accurately match a user's behaviour. Old user profiles were maintained, in order to perform future comparisons and to enable permanent records to be maintained.

A.1.4 Categorising Intrusion Detection Systems

It is generally agreed that IDSs can be categorised by the detection strategy used, either signature based, anomaly based or based on stateful protocol analysis, and by where they are located, either network based or host based [13, 100, 148]. These classifications can be further refined, based on objectives and implementations [10]. The intention of all IDS is to gather and log information, detect incidents and alert to a potential incident happening.

It is also worth noting that an IDS will require a management overhead, which will vary, based on the actual IDS implementation and its complexity.

The main detection methods used by IDS are:

- Signature based
- Anomaly based
- Stateful Protocol Analysis

A.1.4.1 Signature Based

Signature based IDS, sometimes referred to as misuse detection, is the simplest detection method and utilises signatures to identify potential incidents. A *signature* is a pre-determined pattern that corresponds to a known threat. The IDS compares data to a library of known threat signatures, in order to identify possible incidents.

Signature based detection is very effective at detecting known threats and, as the signatures are usually well defined, generally have a low amount of false positive detections.

However, signature based detection is ineffective against unknown threats, as the signatures for these do not exist and can therefore not be compared with the IDS. There is also a management overhead in the signature generation and issuing of new signatures, in order to maintain the signature library and maintain the effectiveness of the detection. The amount of signatures that are required, which is constantly increasing, also has an overhead on the amount of memory required, in which to store the signatures. That is, the greater the number of signatures, then the more memory resources are required to store the signature library [148]. There is often a trade-off, between the size of the signature library and the effectiveness of the system to being able to detect a possible attack.

A.1.4.2 Anomaly Based

In an anomaly based IDS, behaviour models for the normal system characteristics are created, which are known as *profiles*. These normal characteristics are then compared to the actual system characteristics, looking for deviations, within thresholds, from normal behaviour.

The intrusion detection is based on the assertion that intrusions are observable deviations from normal behaviour [77].

An anomaly based IDS often uses behavioural based machine learning techniques to characterise normal system behaviours [174, 194]. The initial characteristic profile is generated over a period of time, which is sometimes called a *training period*. The profiles can be either *static* or *dynamic*, based upon the implementation of the IDS [168]. Once implemented, static profiles do not change, whereas dynamic profiles will adapt to the observed operating environment. Typical techniques for anomaly detection include: cluster analysis, stepping stone analysis, statistical analysis, heuristic based analysis, artificial neural networks, hidden Markov model, entropy based analysis and genetic algorithms or immune based analysis [10, 148].

Anomaly based systems have an advantage over signature based systems as, once the system has been trained and a profile of normal behaviour has been determined, then the system is essentially left to run on its own, without any significant need to interact with it. That is, the through life management overhead of an anomaly based IDS is minimal, when compared to a signature based IDS, as there are no signatures that require updating.

Typical interactions that might still be required with an anomaly based IDS would be the adjustment, or refinement, of normal behaviour. An example could be when a static profile might require updating, to ensure it is tailored for any changes within the systems operating environment.

However, in practise anomaly based IDS do tend to produce higher false positive errors, when compared to signature based IDS. Because of this, it is possible for the amount of false positives to render the anomaly based IDS unusable [174]. This is especially the case in more diverse or dynamic environments [168].

A potential issue with anomaly based IDS which utilises dynamic profiles, is that an attacker can use this to their advantage. The attacker purposely attempts to alter what the normal operating environment characteristics are, as perceived by the anomaly based IDS. Malicious changes can be made either during the training phase, or they can be made more subtly over a period of time. Changes that are made over time would need to be subtle enough, such that the anomaly based IDS would not suspect an attack. The subtle changes would then subsequently modify the signature, so as to take the perceived new operating conditions into account.

Another potential issue with a dynamic IDS, is the subsequent analysis of alerts that have been raised. The alerts can sometimes be difficult for an analyst to interpret after an event, as the analyst will need to investigate why an alert was triggered. In order to be able to do this, the analyst needs to understand what the anomaly based IDS believes are normal operating behaviours. The analyst therefore has to determine what the anomaly based IDS believes to be normal behaviour, in order to understand the context of the alert, which is based on its implementation and the methods it uses to modify its dynamic rule sets. As the modification of rules sets can be complex in nature, this could be a difficult undertaking for the analyst.

A.1.4.3 Stateful Protocol Analysis

Stateful protocol analysis IDS use predetermined *profile definitions* of benign protocol activity, sometimes referred to as *stateful signatures*, which characterise the behaviours of each protocol state. These are then compared against observed events to identify anomalies and possible malicious behaviour [2, 168, 206].

Stateful protocol analysis tends to be a resource intensive operation and cannot detect attacks that do not violate a protocol's characteristic behaviour. To place this in to context, if an attacker was attempting to realise the threat of Denial of Service, the attack that is utilised might use a protocol in a way that is perceived as benign by the IDS.

The advantage of utilising stateful protocol analysis is that it is able to monitor and analyse an entire session, as opposed to single requests and responses within a session [65].

A.2 Intrusion Detection System Technologies

The more mature forms of IDS technologies are network based and host based instantiations. There have been more recent developments within the research and implementations of IDS within Wireless LAN installations, sometimes referred to as Wi-Fi, which are also discussed for completeness.

A.2.1 Network Based IDS

A Network IDS (NIDS) would typically be located within a single host, which forms part of a multicast network. The NIDS reads data off the network, in an attempt to detect intrusions before an attack reaches its target and subsequently infects the target [10, 13, 148]. NIDS are often located either just before or just after a firewall, although they can also be utilised in both locations.

When placed outside the firewall, the network based IDS can provide the defending analyst with useful information as to how attackers are attempting to attack a network.

When placed on the inside of a firewall, it allows the effectiveness of the firewall to be monitored. There are also suggestions where the NIDS is undertaken by separate process, such as one to capture and pre-process information and the other to conduct pattern analysis on the information [84].

NIDS have several advantages:

- Only one is needed within the network, as opposed to one on each host. This can free up resources on the individual hosts, such as processing time and memory requirements.
- They tend to be agnostic of the hosts' operating systems and patch statuses.
- Attacks should be detected before they reach the hosts and therefore limiting potential damage caused by an attack.
- Attackers tend to attack host machines and not the NIDS infrastructure. It can therefore be difficult for an attacker to remove any evidence relating to the method of attack.

However, NIDS do have drawbacks:

- Real time detection on one asset running NIDS can be a significant undertaking. This can lead to latency issues between receiving data for inspection, detecting an incident and then reacting to it.
- There is a trade-off between effectiveness and efficiency. If the NIDS inspected all data and did so to a great degree of depth, then the time taken to complete the undertaking could be such that it makes the network so slow that it would become unusable. However, if the NIDS did not inspect the data to the high degree of depth, although the network might run at an acceptable rate, attacks might not be detected.
- Encrypted files cannot be inspected.
- NIDS tend to work on the assumption that attacks are undertaken from outside of the network and would therefore not be attempting to detect attacks that are undertaken form the inside.

A.2.2 Host Based IDS

Host IDS (HIDS) operate directly on the single host that they are attempting to protect and is undertaken by monitoring the hosts characteristics, such as the memory usage and system calls, in order to monitor for suspicious activity occurring.

HIDS have several advantages:

- There is no need for separate hardware.
- They can generally access encrypted files, as the host will have decrypted them.
- The protection can be tailored for the host and its operating environment.

HIDS also have disadvantages:

- The attack has to have reached the host, in order for the attack to be detected. Therefore, the HIDS has to detect and then react to the incident, before the attack is realised.
- They are potentially vulnerable to attack themselves, or the HIDS can be disabled if the host is compromised.
- There are potential resource restrictions, such as processing or storage limitations within the host. That is, the host is still required to undertake its normal business functions, as well as accommodate the HIDS. This is again another example of the trade-odd between effectiveness and efficiency.
- The HIDS only protects the hosts that it is deployed on within a network architecture, other hosts remain vulnerable to potential attack.

A.2.3 Wireless IDS

A significant number of deployed IDS are utilised within the known environment of a Local Area Network, allowing the managed positioning of NIDS or where to place HIDS agents.

Where a Wireless Local Area Network (WLAN) is utilised, Wireless IDS (WIDS) have been developed and a significant amount of research has been conducted on this subject [3, 80, 168]. WIDS bring further capability to the analyst, over that of conventional IDS. If the operational environment for the wireless network has been surveyed correctly, wireless sources can be tracked within the operating environment, through triangulation between the wireless sources' locations and the wireless access points' locations, providing the analyst with a richer understanding to aid in an investigation.

This can also include wireless sources that are not currently connected to the operational network but are transiting within the operational environment of the wireless network. This could be beneficial if a company had certain areas or locations where it only permitted the presence or use of its own corporate devices. This would allow the analyst to detect and track unknown devices within these controlled areas an assist in determining the location of unknown devices.

Similarly, WIDS can also be designed to observe different frequency bands and protocols, other than that of the WLAN. A typical example is Bluetooth energy sources. Certain instantiations of WIDS will again be able to track these sources through the operational environment

Within a WIDS, the monitoring can be carried out from within the WLAN that is being utilised for normal traffic or as a separate WIDS network, both of which have advantages and disadvantages.

A WIDS that is deployed as part of the normal operational traffic network is generally cheaper to deploy, as it utilises the majority of the existing infrastructure, such as wireless access points, switches and routers. It also allows the WIDS access to details such as actual users on the wireless network and the associated network traffic. The wireless network can also be an extension of a wired network and the IDS can utilise knowledge of the wired infrastructure in order to assist in its detection. As an example, if a user is located at a particular point within the fixed infrastructure, this should correspond to what the wireless infrastructure is detecting. If the details did not correspond, this could be an attacker attempting to realise the threat of masquerade on either the wireless or the wired network.

The disadvantage of utilising the operational network as the WIDS is that there is the potential for an attacker to successfully join the operational wireless network and from there attempt to attack the WIDS. There is also a disadvantage that WIDS will only be capable of attempting to detect rouge devices, which have not joined the operational network, when the operational network is not being utilised to transfer traffic. That is, the priority of the wireless network is generally the connection and subsequent transfer of network traffic.

The WIDS capability is often seen as a secondary requirement for the wireless network. Essentially, when the wireless network is not being used to authenticate users or transfer data traffic, it falls back in to the WIDS mode of operation. Therefore, when a wireless network is heavily populated with users, or the users have high bandwidth requirements, then the WIDS will be less efficient.

To prevent this compromise in performance, or to reduce the risk of an attacker jumping from the operational traffic network to the WIDS, a separate WIDS infrastructure can be implemented. The advantage of a separate WIDS and associated network infrastructure is that, as the WIDS is totally separate from the operational traffic network, it is not constrained by the normal network activities and associated traffic, the wireless access points can be set to receive only, preventing network reconnaissance by an attacker, and the wireless access points can be specified specifically for undertaking the WIDS function, such as by installing access points that have a larger frequency range than that required for the WLAN, or have improved direction finding capabilities, such as by utilising directional antennas. Receive only wireless access points area sometimes utilised outside of the normal wireless operating environment, to assist in triangulation, and subsequent location, of a device. They can also be utilised to monitor for unexpected emissions from a location, such as at the boundary point of a particular location.

The disadvantages of a separate WIDS is the cost of installation and the overhead of administering and maintaining a separate network. There is also the disadvantage of the system only knowing where the wireless devices are and would therefore not be able to correlate between other available information, such a log on location for a user on a wired network.

There are instances of where both implementations are utilised within the same operational environment, with appropriate network protections in place between the two WIDS, to prevent network attack and to maintain the integrity of the information collected by the WIDS.

A.2.4 Typical IDS Components

Although the components of an actual IDS implementation will vary, the components that are typically found on an IDS are [168]:

Sensor or Agent
A sensor or agent is deployed within a system in order to monitor and then analyse activity. The term sensor is typically used when an IDS is utilised to monitor networks, including NIDS and WIDS. The term agent is typically used within HIDS deployments.

Management Server
A management server is often used as a centralised device that collates information from sensors or agents. It allows for the management of the individual sensors or agents and allows information received from the sensors or agents to have further processing undertaken on them. This can be either more in-depth analysis of an individual incident or for system wide analysis, such as trend analysis and correlation. Some smaller installations might not employ a management server, as a small installation might not warrant the expenditure or the overheads. However, on larger deployments, multiple management servers are deployed and they can be deployed within a hierarchical structure.

Database Server
A database server is a repository for all event information recording. This can be from all the elements of the IDS, such as the sensors, the agents, and, if applicable, the management servers.

Console
A console is a program that provides an interface for the IDS's users and administrators. In certain implementations, consoles are used for IDS administration only, such as configuring sensors or agents and applying software updates, while other consoles are used for the tasks of monitoring and analysis.

A.3 Summary

This Annex has provided the reader with an overview of the history of Intrusion Detection Systems and has also provided details as to the mechanisms for the various types of IDS technologies.

B Characteristics of Current Robotic Swarm Implementations

The following annex provides details, specifications and capabilities of typical robotic swarm entities that are currently available.

1. **Kilobot**
   The Kilobot was developed by a team in Harvard University and is now produced by K-Team. The details provided in Table B.1 are taken from their current website [99].

2. **Colias**
   The Colias robot has been developed by The University of Lincoln. The details in Table B.2 are taken from their current website [191] and their publication detailing the robots build [9].

3. **Mona Robo**t
   The Mona robot is an Arduino based robot developed by The University of Manchester Robotic Lab. The details provided in Table B.3 are taken from their current website [121].

4. **Erratic Mobile Platform**
   The Erratic mobile was the robot used for the Guardians project. The characteristics provided in Table B.4 are taken from GUARDIANS reports [135, 147].

5. **S-Bot**
   The S-Bot was developed by École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, for the Swarm-Bot research project. The details provided in Table B.5 are taken from the S-Bot website [181].

6. **e-puck**
   THe e-puck is an educational desktop mobile robot developed by École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. The characteristic provided in Table B.6 has been taken from their documentation [129].

7. **e-puck2**
   In 2018, a new version of the e-puck was produced that increased the performance of certain characteristics and added the ability to store information by Micro SD. The characteristic information in Table B.7 is taken from their website [71].

Property	Details
Processor	ATmega 328P (8bit @ 8MHz)
Memory	32 KB Flash used for both user program and bootloader. 1KB EEPROM for storing calibration values and other non-volatile data. 2KB SRAM.
Battery	Battery Rechargeable Li-Ion 3.7V, for a 3 months autonomy in sleep mode. About 2.5 hours in standard use with motors. Each Kilobot has a built-in charger, which charges the onboard battery when +6 volts is applied to any of the legs, and GND is applied to the charging tab.
Charging	Kilobot charger (optional). Time for charge is about 3 hours.
Communications	Kilobots can communicate with neighbours up to 7 cm away by reflecting infrared (IR) light off the ground surface.
Sensing	When receiving a message, distance to the transmitting Kilobot can be determined using received signal strength. The brightness of the ambient light shining on a Kilobot can be detected.
Movement	Each Kilobot has 2 vibration motors, which are independently controllable, allowing for differential drive of the robot. Each motor can be set to 255 different power levels.
Light	Each Kilobot has a red/green/blue (RGB) LED pointed upward, and each colour has 3 levels of brightness control.
Dimensions	The diameter is 33 mm and the height is 34mm (including the legs).
Software	The KiloGUI interface is available for controlling the controller board, sending program files to the robots and controlling them.
Programming	For programming, the open source development software WinAVR combined with Eclipse gives a C programming environment. An API with basic functions such as motor speed, led control, distance measurement is available and some examples are provided.
Debug	A serial output header is available on each robot for debugging via computer terminal.
Simulator	V-REP, realistic 3D Simulator and robot programming (included for education), with Kilobot model.

Table B.1: Kilobot Characteristics [99]

Property	Details
Processor	2 ATMEL processors, ATMEGA-168 and ATMEGA-644.
Battery	3.7V, 600mAh (extendible to 1200mAh).
Charging	Docking chargers.
Communications	IR, range 5mm to 2m.
Sensing	Long Range IR proximity, Short range IR bump proximity, light sensor on bottom of robot.
Movement	22mm wheels.
Speed	350mm/sec.
Dimensions	40mm Diameter.
Programming	C, BASIC and PASCAL.

Table B.2: Colias Robot Characteristics [9, 191]

Property	Details
Processor	ATmega328 (3.3V, 8MHz).
Communications	I^2C, SPI, RS232, RF and WiFi.
Sensing	Modules: Vision, sonar, manipulator.
Movement	Wheels.
Dimensions	Diameter 65mm.

Table B.3: Mona Robot Characteristics [121]

Property	Details
Processor	Texas Instruments OMAP3, 600MHz Arm Cortex-A8.
Memory	512Mb NAND Flash, 256MB DDR SDRAM.
Communications	WiFi, Bluetooth, I^2C, SPI, USB and UART.
Sensing	Laser range finder.
Movement	Wheels.
Dimensions	130mm diameter.
Programming	OpenRobotix.

Table B.4: Erratic Mobile Platform Characteristics [135, 147]

Property	Details
Processor	XScale Linux 400MHz Processor, 12 PIC Processors.
Memory	64M RAM, 32M Flash.
Battery	10Wh Lithium ION Battery.
Communications	8 Sector RGB LED light ring, omnidirectional camera.
Sensing	15 IR proximity sensors around body, 4 IR proximity sensors on bottom of robot, 8 light sensors around body, 3 axis accelerometer, humidity, temperature, force, torque, one speaker and four microphones.
Movement	Tracks and wheels (Treels) drive mechanism.
Dimensions	116mm diameter, 100mm high.
Actuators	One degree of freedom rigid arm with gripper, three degrees of freedom flexible arm with gripper

Table B.5: S-Bot Characteristics [181]

Property	Details
Processor	16-bit dsPIC30F6014A at 60MHz (15 MIPS), DSP core for signal processing
Memory	8 KB of RAM and 144 KB of flash.
Battery	1800mAh swappable and rechargeable Li-Ion. Approximately 3 hours of autonomy.
Charging	External to robot. Recharge around 2 to 3 hours
Communications	Bluetooth for communications and 8 red LEDs to be observed by camera for visual interaction, I^2C and RS232. Zigbee can be added as an extra extension module.
Sensing	8 infra-red sensors measuring ambient light and proximity of objects up to 60mm, 3D gyro, 3D accelerometer, 3 microphones, 640x480 colour CMOS camera, extension boards include: IR scanner and 3 linear cameras.
Movement	41mm wheels.
Speed	129mm/sec.
Light	Indication to user: 8 red LEDs, 1 strong front red LED and set of green body LEDs.
Dimensions	70mm Diameter, 55mm High. Height can increase and depends on connected extension modules that are used.
Software	C compiler and IDE, Webots simulator, external debugger

Table B.6: e-puck Characteristics [129]

Property	Details
Processor	32-bit STM32F407 at 168 MHz (210 DMIPS), DSP and FPU, DMA
Memory	192 KB of RAM and 1024 KB of flash
Battery	1800mAh swappable and rechargeable Li-Ion. Approximately 3 hours of autonomy.
Charging	USB connected to robot. Recharge around 2.5 hours.
Communications	Bluetooth, 4 red LEDs and 4 RGB LEDs to be observed by camera for visual interaction, USB, WiFi, BLE, I^2C and RS232. Zigbee can be added as an extra extension module.
Sensing	8 infra-red sensors measuring ambient light and proximity of objects up to 60mm, front real distance sensor, time of fight, up to 2 meters, 3D gyro, 3D accelerometer, 3D magnetometer, 4 microphones, 640x480 colour CMOS camera, extension boards include: IR scanner and 3 linear cameras.
Movement	41mm wheels.
Speed	154mm/sec.
Light	Indication to user: 4 red LEDs, 4 RGB LEDs, 1 strong front red LED and set of green body LEDs.
Dimensions	70mm Diameter, 55mm High. Height can increase and depends on connected extension modules that are used.
Software	C compiler and IDE, Webots simulator, onboard debugger (GDB)
Storage	Micro SD slot.

Table B.7: e-puck2 Characteristics [71]

Bibliography

[1] M. Abdel Wahid, C. Murray, K. Johnstone, and C. McInnes. Internal agent states: experiments using the swarm leader concept. In *Towards Autonomous Robotic Systems (TAROS 08)*, 2008.

[2] Y. Al-Nashif, A. A. Kumar, S. Hariri, Y. Luo, F. Szidarovsky, and G. Qu. Multi-level intrusion detection system (ml-ids). In *International Conference on Autonomic Computing, 2008. ICAC '08.*, pages 131–140, 2008.

[3] Q. I. Ali and S. Lazim. Design and implementation of an embedded intrusion detection system for wireless applications. *IET Information Security*, 6(3):171–182, 2012.

[4] J. Amenyo, D. Phelps, O. Oladipo, F. Sewovoe-Ekuoe, S. Jadoonanan, S. Jadoonanan, T. Tabassum, S. Gnabode, T. D. Sherpa, M. Falzone, A. Hossain, and A. Kublal. Medizdroids project: Ultra-low cost, low-altitude, affordable and sustainable uav multicopter drones for mosquito vector control in malaria disease management. In *IEEE Global Humanitarian Technology Conference (GHTC 2014)*, pages 590–596, Oct 2014.

[5] S. H. M. Amin and A. Adriansyah. Particle swarm fuzzy controller for behavior-based mobile robot. In *9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06.*, pages 1–6, 2006.

[6] M. Anand, Z. Ives, and I. Lee. Quantifying eavesdropping vulnerability in sensor networks. In *Proceedings of the 2nd international workshop on Data management for sensor networks*, pages 3–9. ACM, 2005.

[7] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., 1980.

[8] R. C. Arkin, T. Balch, and E. Nitz. Communication of behavorial state in multi-agent retrieval tasks. In *IEEE International Conference on Robotics and Automation, 1993. Proceedings., 1993*, pages 588–594. IEEE, 1993.

[9] F. Arvin, J. Murray, C. Zhang, and S. Yue. Colias: An autonomous micro robot for swarm robotic applications. *International Journal of Advanced Robotic Systems*, 11(7):113, 2014.

[10] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Chalmers University of Technology, 2000.

[11] F. M. Aziz, J. S. Shamma, and G. L. Stüber. Resilience of lte networks against smart jamming attacks. In *2014 IEEE Global Communications Conference*, pages 734–739, Dec 2014.

[12] G. Baele, N. Bredeche, E. Haasdijk, S. Maere, N. Michiels, Y. Van de Peer, T. Schmickl, C. Schwarzer, and R. Thenius. Open-ended on-board evolutionary robotics for robot swarms. In *IEEE Congress on Evolutionary Computation, 2009. CEC '09.*, pages 1123–1130, 2009.

[13] Y. Bai and H. Kobayashi. Intrusion detection systems: technology and development. In *17th International Conference on Advanced Information Networking and Applications, 2003. AINA 2003.*, pages 710–715, 2003.

[14] T. Balch and R. C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14:926–939, 1997.

[15] M. Barley, H. Mouratidis, A. Unruh, D. Gordon-Spears, P. Scerri, and F. MASSACCI. *Safety and Security in Multiagent Systems: Research Results from 2004-2006*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009.

[16] L. Bayindir and E. Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147, 2007.

[17] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In J. Clark, R. Paige, F. Polack, and P. Brooke, editors, *Security in Pervasive Computing*, volume 3934 of *Lecture Notes in Computer Science*, pages 104–118. Springer Berlin Heidelberg, 2006.

[18] D. Beekman, J. Mait, and T. Doligalski. Micro autonomous systems and technology at the army research laboratory. In *IEEE National Aerospace and Electronics Conference, 2008. NAECON 2008.*, pages 159–162, 2008.

[19] G. Beni. From swarm intelligence to swarm robotics. In E. Şahin and W. M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer Berlin Heidelberg, 2005.

[20] S. Berman, A. Halasz, V. Kumar, and S. Pratt. Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In *in Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[21] J. D. Bjerknes, A. F. Winfield, C. Melhuish, and C. Lane. An analysis of emergent taxis in a wireless connected swarm of mobile robots. In *IEEE Swarm Intelligence Symposium*, pages 45–52. IEEE Press, 2007.

[22] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence*. Oxford, 1999.

[23] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behavior, 2000.

[24] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[25] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, 1986.

[26] D. Brown. Nasa refines asteroid apophis' path toward earth, 2009. https://cneos.jpl.nasa.gov/news/news164.html (Last Accessed: 03/01/2019).

[27] J. Brown and X. Du. Detection of selective forwarding attacks in heterogeneous sensor networks. In *IEEE International Conference on Communications, 2008. ICC '08.*, pages 1583–1587, 2008.

[28] BS ISO/IEC. BS ISO/IEC 13335-1 information technology - security techniques - management of information and communications technology security, 12 2004.

[29] J. Cacace, A. Finzi, and V. Lippiello. Implicit robot selection for human multi-robot interaction in search and rescue missions. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 803–808, Aug 2016.

[30] J. Cacace, A. Finzi, V. Lippiello, M. Furci, N. Mimmo, and L. Marconi. A control architecture for multiple drones operated via multimodal interaction in search amp;amp; rescue mission. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 233–239, Oct 2016.

[31] Y. Cao, A. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[32] CESG. HMG IA Standard No. 1 - Technical Risk Assessment, 2009.

[33] Q. Chai, G. Gong, and D. Engels. How to develop clairaudience - active eavesdropping in passive rfid systems. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012*, pages 1–6, 2012.

[34] E. Chapman and F. Sahin. Application of swarm intelligence to the mine detection problem. In *SMC (6)*, pages 5429–5434, 2004.

[35] Z. Chen, S. Guo, K. Zheng, and Y. Yang. Modeling of man-in-the-middle attack in the wireless networks. In *International Conference onWireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.*, pages 2255–2258, 2007.

[36] E. J. Cho, C. S. Hong, and D. Choi. Distributed ids for efficient resource management in wireless sensor network. In *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pages 1–5, 2011.

[37] J. Choi, J. Lee, and S. Oh. Swarm intelligence for achieving the global maximum using spatio-temporal gaussian processes. In *American Control Conference, 2008*, pages 135–140, 2008.

[38] A. L. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, Aug 2009.

[39] C. M. Cianci, X. Raemy, J. Pugh, A. Martinoli, and E. P. Federaledelausanne. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In *in Simulation of Adaptive Behavior (SAB-2006), Swarm Robotics Workshop*, pages 103–115. Springer Lecture, 2006.

[40] A. Colorni, M. Dorigo, V. Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[41] Committee On National Security Systems. Committee on national security systems national information assurance (IA) glossary, 2010.

[42] D. W. Corne, A. Reynolds, and E. Bonabeau. Swarm intelligence. In G. Rozenberg, T. Bäck, and J. Kok, editors, *Handbook of Natural Computing*, pages 1599 – 1622. Springer Berlin Heidelberg, 2012.

[43] N. Correll and A. Martinoli. Towards optimal control of self-organized robotic inspection systems. *IFAC Proceedings Volumes*, 39(15):304 – 309, 2006. 8th IFAC Symposium on Robot Control.

[44] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole. Buffer overflows: attacks and defenses for the vulnerability of the decade. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, volume 2, pages 119–129 vol.2, 2000.

[45] Y.-S. Dai, M. Hinchey, M. Madhusoodan, J. Rash, and X. Zou. A prototype model for self-healing and self-reproduction in swarm robotics system. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 3–10, 2006.

[46] David Chandler. Asteroid is 'practice case' for potential hazards, 2007. http://news.mit.edu/2007/asteroid-1013 (Lasdt Accessed: 03/10/2019).

[47] A. Davison. *Killer Game Programming in Java*. O'Reilly Media, Inc., 2005.

[48] dcist. Distributed Collaborative Intelligent Systems and Technology (DCIST). https://www.dcist.org (Last Accessed: 19/12/2018).

[49] M. De Gennaro and A. Jadbabaie. Formation control for a cooperative multiagent system using decentralized navigation functions. In *American Control Conference, 2006*, page 6, 2006.

[50] D. Denning and P. G. Naumann. Requirements and model for ides - a real-time intrusion-detection expert system. Technical report, SRI Internatioal, 1985.

[51] D. E. Denning. An intrusion-detection model. In *1986 IEEE Symposium on Security and Privacy*, pages 118–118, 1986.

[52] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[53] J. L. Dohner. A guidance and control algorithm for scent tracking micro-robotic vehicle swarms. Technical report, Sandia National Laboratories, 1997.

[54] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, Mar 1983.

[55] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[56] M. Dorigo, E. Tuci, R. Groß, V. Trianni, T. Labella, S. Nouyan, C. Ampatzis, J.-L. Deneubourg, G. Baldassarre, S. Nolfi, F. Mondada, D. Floreano, and L. Gambardella. The swarm-bots project. In E. Şahin and W. M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 31–44. Springer Berlin Heidelberg, 2005.

[57] J. R. Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[58] F. Ducatelle, G. A. Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, F. Mondada, R. O'Grady, C. Pinciroli, P. Rétornaz, V. Trianni, and L. M. Gambardella. Cooperative navigation in robotic swarms. *Swarm Intelligence*, 8(1):1–33, 2013.

[59] F. Ducatelle, G. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella. Communication assisted navigation in robotic swarms: Self-organization and cooperation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*, pages 4981–4988, 2011.

[60] G. Dudek, M. Jenkin, and E. Milios. A taxonomy of multirobot systems. *Robot Teams: From Diversity to Polymorphism*, pages 3 – 22, 2002.

[61] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

[62] G. Dudek, M. R. M. Jenkin, and D. Wilkes. A taxonomy for swarm robots. In *Proceeding on the IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 441–447. IEEE, 1993.

[63] Federal Information Processing Standards. FIPS pub 200 - minimum security requirements for federal information and information systems, 3 2006.

[64] W. Ford and M. S. Baum. *Secure electronic commerce: building the infrastructure for digital signatures and encryption*. Prentice Hall PTR, 2000.

[65] K. K. Frederick. Network intrusion detection signatures, part 5, 2002. https://www.symantec.com/connect/articles/network-intrusion-detection-signatures-part-five (Last Accessed: 19/12/2018).

[66] J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[67] T. Fukuda, D. Funato, K. Sekiyama, and F. Arai. Evaluation on flexibility of swarm intelligent system. In *ICRA*, pages 3210–3215. IEEE Computer Society, 1998.

[68] T. Fukuda, G. Iritani, T. Ueyama, and F. Arai. Optimization of group behavior on cellular robotic system in dynamic environment. In *ICRA*, pages 1027–1032. IEEE Computer Society, 1994.

[69] T. Fukuda and S. Nakagawa. Approach to the dynamically reconfigurable robotic system. *Journal of Intelligent and Robotic Systems*, 1(1):55–72, 1988.

[70] S. Garnier, J. Gautrais, and G. Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1):3–31, 2007.

[71] GCtronic. e-puck2, 2018. http://www.gctronic.com/doc/index.php/e-puck2 (Last Accessed: 19/12/2018).

[72] K. Geetha and N. Sreenath. Syn flooding attack - identification and analysis. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pages 1–7, Feb 2014.

[73] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In M. Naor, editor, *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer Berlin Heidelberg, 2004.

[74] B. P. Gerkey and M. J. Mataric. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[75] M. Gerla, Y. Yi, K. Xu, and X. Hong. Team communications among airborne swarms. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 3, pages 1303 – 1312, 2003.

[76] Goddard Space Flight Center. Autonomous NanoTechnology Swarm ANTS. http://attic.gsfc.nasa.gov/ants/ (Last Accessed: 19/12/2018).

[77] D. Gollmann. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):544–554, 2010.

[78] L. Gong. Variations on the themes of message freshness and replay-or the difficulty in devising formal methods to analyze cryptographic protocols. In *Computer Security Foundations Workshop VI, 1993. Proceedings*, pages 131–136, 1993.

[79] P.-P. Grassié. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[80] F. Haddadi and M. A. Sarram. Wireless intrusion detection system using a lightweight agent. In *Second International Conference on Computer and Network Technology (ICCNT)*, pages 84–87, 2010.

[81] H. Hamann and H. Wörn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2):209–239, Dec 2008.

[82] J. A. Hansen and N. M. Hansen. A taxonomy of vulnerabilities in implantable medical devices. In *Proceedings of the second annual workshop on Security and privacy in medical and home-care systems*, pages 13–20. ACM, 2010.

[83] A. Hassanzadeh and R. Stoleru. Towards optimal monitoring in cooperative ids for resource constrained wireless networks. In *Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, 2011.

[84] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM, 1990.

[85] P. E. Heegaard and O. J. Wittner. Self-tuned refresh rate in a swarm intelligence path management system. In H. de Meer and J. P. G. Sterbenz, editors, *Self-Organizing Systems*, IWSOS2006 EuroNGI 2006, pages 148 – 162, Berlin, Heidelberg, 2006. Springer-Verlag.

[86] O. Holland, J. Woods, R. De Nardi, and A. Clark. Beyond swarm intelligence: the ultraswarm. *Proceedings 2005 IEEE Swarm Intelligence Symposium*, pages 217–224, 2005.

[87] Y. Hu, A. Perrig, and D. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[88] International Federation of Robotics - IFR. History of industrial robots - from the first installation until today - milestones of technology and commercialization, 2012.

[89] ISO. ISO 13491-1 banking - secure cryptographic devices (retail) - part 1: Concepts, requirements and evaluation methods, 2007.

[90] ISO. ISO 13491-2: 2005: Banking - secure cryptographic devices (retail), part 2: Security compliance checklists for devices used in financial transactions, 2013.

[91] ISO/IEC. BS ISO/IEC 27005:2011 – information technology – security techniques – information security risk management, 2011.

[92] E. Izquierdo-Torres. Collective intelligence in multi-agent robotics: Stigmergy, self-organization and evolution. Technical report, University of Sussex, 2004.

[93] K. Jackson, D. DuBois, and C. Stallings. DOE computer security group conference. In *A phased approach to network intrusion detection*, Jan 1991.

[94] M. Jain and H. Kandwal. Notice of violation of ieee publication principles a survey on complex wormhole attack in wireless ad hoc networks. In *International Conference on Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09.*, pages 555–558, 2009.

[95] K. Jin, P. Liang, and G. Beni. Stability of synchronized distributed control of discrete swarm structures. In *ICRA*, pages 1033–1038. IEEE Computer Society, 1994.

[96] C. Johnson, G. Venayagamoorthy, and P. Palangpour. Hardware implementations of swarming intelligence - a survey. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pages 1–9, 2008.

[97] J. E. Jones. On the determination of molecular fields. ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738):pp. 463–477, 1924.

[98] R. P. Jover, J. Lackey, and A. Raghavan. Enhancing the security of lte networks against jamming attacks. *EURASIP Journal on Information Security*, 2014(1):7, Apr 2014.

[99] K-Team. Kilobot. https://www.k-team.com (Last Accessed: 19/12/2018).

[100] P. Kabiri and A. A. Ghorbani. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1:84–102, 2005.

[101] N. M. Kakalis and Y. Ventikos. Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials*, 154(1):880 – 887, 2008.

[102] S. Kazadi. *Swarm Engineering*. PhD thesis, 2000.

[103] S. Kazadi. On the development of a swarm engineering methodology. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1423–1428, 2005.

[104] S. Kelly and T. C. Clancy. Control and provisioning of wireless access points (capwap) threat analysis for ieee 802.11 deployments. Request for Comments, 2009.

[105] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[106] W. Kerr and D. Spears. Robotic simulation of gases for a surveillance task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)*, pages 2905–2910, 2005.

[107] M. Y. Khalid. Intrusion detection system in wireless sensor networks. In *12th Research Seminar Series Workshop*, 2013.

[108] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference onRobotics and Automation*, volume 2, pages 500–505, 1985.

[109] C. King, X. Palathingal, M. Nicolescu, and M. Nicolescu. A control architecture for long-term autonomy of robotic assistants. In *Advances in Visual Computing*, pages 375–384. Springer, 2007.

[110] K. Kruzelecki. Flying Ad-Hoc Networks, 2015. http://smavnet.epfl.ch/ (Last Accessed: 03/01/2019).

[111] V. Kumar, S. Chakraborty, F. Barbhuiya, and S. Nandi. Detection of stealth man-in-the-middle attack in wireless lan. In *2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC)*, pages 290–295, 2012.

[112] V. Kumar and F. Sahin. Cognitive maps in swarm robots for the mine detection application. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3364–3369 vol.4, 2003.

[113] V. Kumar and F. Sahin. Foraging in ant colonies applied to the mine detection problem. In *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications, 2003. SMCia/03*, pages 61–66, 2003.

[114] V. Kumar M and F. Sahin. A swarm intelligence based approach to the mine detection problem. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 6 pp. vol.3–, 2002.

[115] D. Kushner. The real story of stuxnet - how Kaspersky lab tracked down the malware that stymied iran's nuclear-fuel enrichment program, 2013.

[116] K. Lemke, C. Paar, and A. Sadeghi. Physical security bounds against tampering. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security*, volume 3989 of *Lecture Notes in Computer Science*, pages 253–267. Springer Berlin Heidelberg, 2006.

[117] Y. Liu and K. M. Passino. Swarm Intelligence: Literature Overview. Technical report, Ohio State University, 2000.

[118] X. Luo, X. Ji, and M. Park. Location privacy against traffic analysis attacks in wireless sensor networks. In *2010 International Conference on Information Science and Applications (ICISA)*, pages 1–6, 2010.

[119] X. Luo, S. Li, and X. Guan. Flocking algorithm with multi-target tracking for multi-agent systems. *Pattern Recognition Letters*, 31(9):800–805, 2010.

[120] C. A. Maddock, M. Vasile, C. McInnes, G. Radice, and L. Summerer. Designs of multi-spacecraft swarms for the deflection of apophis by solar sublimation. In *1st IAA Planetary Defence Conference: Protecting Earth from Asteroids*, 2009.

[121] Manchester University Robotic Lab. Mona Robot Project. http://www.monarobot.uk/index.htm (Last Accessed: 19/12/2018).

[122] L. Marconi, S. Leutenegger, S. Lynen, M. Burri, R. Naldi, and C. Melchiorri. Ground and aerial robots as an aid to alpine search and rescue: Initial sherpa outcomes. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–2, Oct 2013.

[123] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello,

A. Finzi, B. Siciliano, A. Sala, and N. Tomatis. The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–4, Nov 2012.

[124] A. Martinoli, K. Easton, and W. Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *Int. Journal of Robotics Research*, 23:415–436, 2004.

[125] R. S. Michael L. Rilee. ANTS: Autonomous Nanotechnological Swarm, 2005. http://www.mind.ilstu.edu/curriculum/ants_nasa/ants_pam.php (Last Accessed: 03/01/2019).

[126] MIT. Seaswarm, 2018. http://senseable.mit.edu/seaswarm (Last Accessed: 19/12/2018).

[127] Y. Mo and B. Sinopoli. Secure control against replay attacks. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 911–918, 2009.

[128] C. Moeslinger, T. Schmickl, and K. Crailsheim. A minimalist flocking algorithm for swarm robots. In G. Kampis, I. Karsai, and E. Szathmáry, editors, *Advances in Artificial Life. Darwin Meets von Neumann*, pages 375–382, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[129] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1(1):59–65, 2009.

[130] F. Mondada, L. Gambardella, D. Floreano, S. Nolfi, J. Deneuborg, and M. Dorigo. The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robotics Automation Magazine*, 12(2):21 – 28, 2005.

[131] F. Mondada, G. C. Pettinaro, A. Guignard, I. W. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, and et al. Swarm-bot: a new distributed robotic concept. *Autonomous Robots*, 17:2004, 2003.

[132] D. Morin. *Introduction to Classical Mechanics With Problems and Solutions*. Cambridge University Press, 2008.

[133] V. Munirajan, F. Sahin, and E. Cole. Ant colony optimization based swarms: implementation for the mine detection application. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 716–721 vol.1, 2004.

[134] M. Murphy. Project apophis, 2017. https://eapsweb.mit.edu/news/2017/project-apophis (Last Accessed: 03/01/2019).

[135] A. Naghsh, J. Gancet, A. Tanoto, and C. Roast. Analysis and design of human-robot swarm interaction in firefighting. In *The 17th IEEE International Symposium*

on Robot and Human Interactive Communication, 2008. RO-MAN 2008, pages 255–260, 2008.

[136] W. Naik Bhukya, G. Suresh Kumar, and A. Negi. A study of effectiveness in masquerade detection. In *TENCON 2006. 2006 IEEE Region 10 Conference*, pages 1–4, 2006.

[137] Nanotechnology Characterization Laboratory. Nanotechnology Characterization Laboratory, 2018. https://ncl.cancer.gov/ (Last Accessed: 30/09/2018).

[138] H. J. A. Nasir and K. R. Ku-Mahamud. Wireless sensor network: A bibliographical survey. *Indian Journal of Science and Technology*, 9(38), 2016.

[139] National Initiative For Cybersecurity Careers And Studies - NICCS - US CERT. A glossary of common cybersecurity terminology - attack. http://niccs.us-cert.gov/glossary (Last Accessed: 19/12/2018).

[140] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004.

[141] Northwestern University. NetLogo, 2018. https://ccl.northwestern.edu/netlogo/index.shtml (Last Accessed: 03/01/2019).

[142] S. Nouyan, R. Groß, M. Bonani, F. Mondada, and M. Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, 2009.

[143] L. E. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[144] H. V. D. Parunak. "go to the ant": Engineering principles from natural multi-agent systems. *Annals of Operations Research*, 75:69–101, 1997.

[145] D. W. Payton, M. J. Daily, B. Hoff, M. D. Howard, and C. L. Lee. Pheromone robotics. In *Intelligent Systems and Smart Manufacturing*, pages 67–75. International Society for Optics and Photonics, 2001.

[146] V. Pejovic, S. Bojanic, C. Carreras, and O. Nieto-Taladriz. Detecting masquerading attack in software and in hardware. In *IEEE Mediterranean Electrotechnical Conference, 2006. MELECON 2006.*, pages 836–838, 2006.

[147] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal. A robot swarm assisting a human fire-fighter. *Advanced Robotics*, 25(1-2):93–117, 2011.

[148] H. E. Poston. A brief taxonomy of intrusion detection strategies. In *2012 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 255–263, 2012.

[149] A. Purnamadjaja and R. A. Russell. Pheromone communication: implementation of necrophoric bee behaviour in a robot swarm. In *2004 IEEE Conference on Robotics, Automation and Mechatronics*, volume 2, pages 638–643 vol.2, 2004.

[150] D. N. Renzo. *Flocking of UAVs Software model and limited vision simulations*. PhD thesis, 2004.

[151] R. Research Laboratory. Micro Autonomous Systems and Technology (MAST). http://www.arl.army.mil/www/default.cfm?page=332 (Last Accessed: 19/12/2018).

[152] C. Reynolds. Boids, 2007. http://www.red3d.com/cwr/boids/ (Last Accessed 19/12/2018).

[153] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[154] J. Roberts, J. Zufferey, and D. Floreano. Energy management for indoor hovering robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 1242–1247, 2008.

[155] T. Roosta, S. Shieh, and S. Sastry. taxonomy of security attacks in sensor networks and countermeasures. In *In The First IEEE International Conference on System Integration and Reliability Improvements. Hanoi*, pages 13–15, 2006.

[156] A. W. Roscoe, M. Goldsmith, S. J. Creese, and I. Zakiuddin. The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model. In *Proc. of First International Workshop on Formal Aspects in Security and Trust*, 2003.

[157] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298, May 2012.

[158] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[159] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics*, pages 10–20, 2005.

[160] E. Şahin and A. Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2):69–72, 2008.

[161] F. Sahin. Groundscouts: architecture for a modular micro robotic platform for swarm intelligence and cooperative robotics. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 929–934 vol.1, 2004.

[162] SANS. Glossary of security terms. http://www.sans.org/security-resources/glossary-of-terms/?pass=m (Last Accessed: 19/12/2018).

[163] I. Sargeant. RHUL-Swarm, 2019. https://github.com/RHUL-Swarm (Last Accessed: 12/02/2019).

[164] I. Sargeant and A. Tomlinson. Modelling malicious entities in a robotic swarm. In *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*, pages 7B1–1–7B1–12, 2013.

[165] I. Sargeant and A. Tomlinson. Maliciously manipulating a robotic swarm. In *International Conference Embedded Systems, Cyber-physical Systems, & Applications (ESCS'16)*, pages 122–128, 2016.

[166] I. Sargeant and A. Tomlinson. Review of potential attacks on robotic swarms. In Y. Bi, S. Kapoor, and R. Bhatia, editors, *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, pages 628–646, Cham, 2018. Springer International Publishing.

[167] K. Scarfone, D. Dicoi, M. Sexton, and C. Tibbs. Guide to securing legacy IEEE 802.11 wireless networks. *NIST Special Publication*, 800:48, 2008.

[168] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). Special Publication 800-94, National Institute of Standards and Technology, 2007.

[169] P. Scerri, D. Pynadath, L. Johnson, P. Rosenbloom, M. Si, N. Schurr, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 433–440, New York, NY, USA, 2003. ACM.

[170] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2011.

[171] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong. Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, 17(1):93–105, 2004.

[172] M. D. Shervin Nouyan, Alexandre Campo. Path formation in a robot swarm - self-organized strategies to find your way home. *Swarm Intelligence*, 2(1):1–23, 2008.

[173] C. Shields. What do we mean by network denial of service. In *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security*, volume 4, 2002.

[174] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, 2010.

[175] O. Soysal, E. Bahceci, and E. Şahin. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal of Electrical Engineering*, 15(2), 2007.

[176] O. Soysal and E. Şahin. A macroscopic model for self-organized aggregation in swarm robotic systems. In *Proceedings of the 2nd International Conference on Swarm Robotics*, SAB'06, pages 27–42, Berlin, Heidelberg, 2007. Springer-Verlag.

[177] O. Soysal and E. Sahin. Probabilistic aggregation strategies in swarm robotic systems. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005*, pages 325–332, 2005.

[178] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Auton. Robots*, 17(2-3):137–162, 2004.

[179] R. L. Stewart. A distributed feedback mechanism to regulate wall construction by a robotic swarm. *Adaptive Behavior*, 14:21 – 51, 2006.

[180] S. Subchan, B. White, A. Tsourdos, M. Shanmugavel, and R. Zbikowski. Pythagorean hodograph (ph) path planning for tracking airborne contaminant using sensor swarm. In *IEEE Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008*, pages 501–506, 2008.

[181] Swarm-Bots. Swarm bots, 2014. http://www.swarm-bots.org/index.html (Last Accessed: 19/12/2018).

[182] F. Swiderski and W. Snyder. *Threat Modelling*. Microsoft Press, 2004.

[183] Y. Tan and Z. yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18 – 39, 2013.

[184] Y. C. Tan and B. Bishop. Evaluation of robot swarm control methods for underwater mine countermeasures. In *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory, 2004*, pages 294–298, 2004.

[185] H. Tang, R. Sun, and W. Kong. Wireless intrusion detection for defending against tcp syn flooding attack and man-in-the-middle attack. In *2009 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1464–1470, 2009.

[186] S. Tang and B. Mark. Analysis of virus spread in wireless sensor networks: An epidemic model. In *7th International Workshop on Design of Reliable Communication Networks, 2009. DRCN 2009*, pages 86–91, 2009.

[187] G. Theraulaz, E. Bonabeau, and J.-N. Denuebourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1393):327–332, 1998.

[188] G. Theraulaz and E. Bonbeau. A brief history of stigmergy. *Artificial Life*, 5(2):97–116, 1999.

[189] G. Theraulaz, M. Pratte, and J. Gervet. Behavioural profiles in polistes dominulus (christ) wasp societies: a quantitative study. *Behaviour*, pages 223–250, 1990.

[190] United States Computer Emergency Readiness Team. Security Tip (ST04-015) Understanding Denial-of-Service Attacks. https://www.us-cert.gov/ncas/tips/ST04-015 (Last Accessed: 19/12/2018).

[191] University of Lincoln. Colias. http://www.colias.uk/index.htm (Last Accessed: 19/12/2018).

[192] US Department of Defense. Department of defense announces successful micro-drone demonstration, 2017.

[193] US Department of Defense. Perdix fact sheet, 2017.

[194] E. Viegas, A. Santin, A. Franca, R. Jasinski, V. Pedroni, and L. Oliveira. Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. *IEEE Transactions on Computers*, PP(99):1–1, 2016.

[195] S. von Mammen and C. Jacob. Evolutionary swarm design of architectural idea models. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 143–150, New York, NY, USA, 2008. ACM.

[196] S. von Mammen, S. Novakowski, G. Hushlak, and C. Jacob. Evolutionary swarm design: How can swarm-based systems help to generate and evaluate designs? *DESIGN Principles & Practices: An International Journal*, 3, 2009.

[197] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1530–1539, June 2002.

[198] X. Wang and J. Wong. An end-to-end detection of wormhole attack in wireless ad-hoc networks. In *31st Annual International Computer Software and Applications Conference, 2007. COMPSAC 2007*, volume 1, pages 39–48, 2007.

[199] F. WeiXing, W. KeJun, Y. XiuFen, and G. ShuXiang. Novel algorithms for coordination of underwater swarm robotics. In *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, pages 654–659, 2006.

[200] M. Whitman and H. Mattord. *Principles of information security*. Cengage Learning, 2011.

[201] E. O. Wilson. The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1):pp. 89–98, 1984.

[202] A. F. T. Winfield, C. J. Harper, and J. Nembrini. Towards dependable swarms and a new discipline of swarm engineering. In *Proceedings of the 2004 international conference on Swarm Robotics*, SAB'04, pages 126–142. Springer-Verlag, Berlin, Heidelberg, 2005.

[203] A. F. T. Winfield and J. Nembrini. Safety in Numbers: Fault Tolerance in Robot Swarms. *International Journal on Modelling Identification and Control*, 1(1):30–37, 2006.

[204] H. Woern, M. Szymanski, and J. Seyfried. The I-SWARM project. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006*, pages 492–496, 2006.

[205] A. Yamashita, T. Arai, J. Ota, and H. Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, 19(2):223–237, 2003.

[206] Y. Yang, K. McLaughlin, S. Sezer, Y. B. Yuan, and W. Huang. Stateful intrusion detection for iec 60870-5-104 scada security. In *2014 IEEE PES General Meeting | Conference Exposition*, pages 1–5, 2014.

[207] B. Yu and B. Xiao. Detecting selective forwarding attacks in wireless sensor networks. In *20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006.*, pages 8 pp.–, 2006.

[208] M. M. Zanjireh, A. Shahrabi, and H. Larijani. Anch: A new clustering algorithm for wireless sensor networks. In *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 450–455, 2013.