BIOLOGICALLY INSPIRED, SELF ORGANIZING COMMUNICATION NETWORKS

By

Yousef Elabd Mohammad Hamouda

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Supervised by Dr. Chris Phillips

School of Electronic Engineering and Computer Science

Queen Mary
University of London

February 2011

Abstract

The problem of energy-efficient, reliable, accurate and self-organized target tracking in Wireless Sensor Networks (WSNs) is considered for sensor nodes with limited physical resources and abrupt manoeuvring mobile targets. A biologically inspired, adaptive multi-sensor scheme is proposed for collaborative Single Target Tracking (STT) and Multi-Target Tracking (MTT). Behavioural data obtained while tracking the targets including the targets' previous locations is recorded as metadata to compute the target sampling interval, target importance and local monitoring interval so that tracking continuity and energy-efficiency are improved. The subsequent sensor groups that track the targets are selected proactively according to the information associated with the predicted target location probability such that the overall tracking performance is optimized or nearly-optimized. One sensor node from each of the selected groups is elected as a main node for management operations so that energy efficiency and load balancing are improved. A decision algorithm is proposed to allow the "conflict" nodes that are located in the sensing areas of more than one target at the same time to decide their preferred target according to the target importance and the distance to the target. A tracking recovery mechanism is developed to provide the tracking reliability in the event of target loss.

The problem of task mapping and scheduling in WSNs is also considered. A Biological Independent Task Allocation (BITA) algorithm and a Biological Task Mapping and Scheduling (BTMS) algorithm are developed to execute an application using a group of sensor nodes. BITA, BTMS and the functional specialization of the sensor groups in target tracking are all inspired from biological behaviours of differentiation in zygote formation.

Simulation results show that compared with other well-known schemes, the proposed tracking, task mapping and scheduling schemes can provide a significant improvement in energy-efficiency and computational time, whilst maintaining acceptable accuracy and seamless tracking, even with abrupt manoeuvring targets.

Acknowledgment

First of all, all praises are for my God for giving me his boundless bounties especially giving me a good health and powerful determinations to be able to do this thesis.

All my deep appreciations are due to my supervisor, Dr. Chris Phillips, for his invaluable academic and technical supports and helps. He has not only academically supervised me in the thesis, but he has also logically taught me the right way of thinking, tracking problems and overcoming the challenges. Additionally, I register my indebtedness to him. He always guides me to the right direction, helps me to solve any academic problem and encourages me constantly.

I also would like to extend my thankfulness to all academic and administrative staff in Electronic Engineering Department, especially my academic teachers. Also, I would like to express my thankfulness to Queen Mary, University of London which awarded me a full scholarship to do my PhD.

I also would like to thank my colleagues in the department, in particular, Ali, Frankie, Oliver and Adeel. I extend my thankfulness to all my friends in UK, Egypt and Palestine especially the dean of admission and registration, Dr. Fayik El Nawaak, Omer, Mazen, Bahjat and Dr Abdallah Al Zain about their enthusiasm and encouragement throughout my PhD.

Last but not least, I would like to truly dedicate this work to my parents in Palestine. I record my profound gratitude to them. They have trained me, taught me and encouraged me to pursuer my postgraduate studies. Many lovely thanks to my wife, Ekram, who always technically and emotionally supported me throughout my PhD, despite the vast separating distances. She has shown great patience and understanding and exerted tremendous efforts looking after our kids. The thankfulness is extended to my brothers, sisters and relatives in Egypt and Palestine for their constant emotional support.

Table of Contents

Abstract 2
Acknowledgment 3
Table of Contents 4
List of Figures 8
List of Tables 11
Glossary of Terms 12
Chapter 1 Introduction 15
1.1 Chapter Introduction 15
1.2 The Research Problem Definition 15
1.2.1 Brief Introduction about WSNs 15
1.2.2 Target Tracking in WSNs 16
1.2.3 Task Mapping and Scheduling in WSNs 18
1.3 The Research Motivation 18
1.4 The Research Objectives 19
1.5 Novelty & Contributions 20
1.6 Notation Conventions 21
1.7 Thesis Structure 21
1.8 Authorship 22
Chapter 2 Background and Literature Review 24
2.1 Chapter Introduction 24
2.2 Wireless Sensor Networks (WSNs) Overview & Applications 24
2.3 Energy Consumptions Factors in WSNs 26
2.4 WSN MAC Protocols 27
2.5 WSNs Routing Protocols 30
2.6 WSNs Target Tracking 32
2.6.1 Target Detection 33
2.6.2 Target Classification 34
2.6.3 Node Selection 35
2.6.4 Target Localization 35
2.6.5 Target Tracking 37
2.6.5.1 Bayesian Networks 38
2.6.5.2 Extended Kalman Filter (EKF) 41
2.6.6 Behaviour Analysis 41
2.6.7 Person Identification 42
2.6.8 WSNs Target Tracking Architecture 42
2.7 Biologically Inspired Research & Self-Organized Networks 43
2.8 Task Mapping and Scheduling in WSNs 44
2.9 State-of-the Art Review 45
2.9.1 Biological Inspired Techniques and Algorithms 45
2.9.2 Target Tracking in WSNs 46
2.9.2.1 Target Tracking Framework 46
2.9.2.2 Sensor Nodes Selection Algorithms 52
2.9.2.3 Sensor Nodes Election Algorithms 56
2.9.2.4 Target Tracking Techniques 56
2.9.3 Task Mapping and Scheduling in WSNs 58
2.10 Chapter Summary 61
Chapter 3 Single Target Tracking in WSNs 62
3.1 Chapter Introduction 62

3.2 Target Dynamic Model 62
3.3 Sensor Detection and Measurement Model 63
3.4 Energy Consumption Model 64
3.5 Extended Kalman Filter for Single Target Tracking 65
3.6 Multi-Sensor Adaptive Single Target Tracking Framework 67
3.7 Target Metadata 69
3.8 Adaptive Sampling Interval Selection Algorithm 71
3.9 Sensor Nodes Selection Management 73
3.9.1 Target Model 73
3.9.2 Sensor Nodes Selection Algorithm 74
3.9.3 Adaptive Group Size Algorithm 75
3.10 Sensor Node Election 76
3.11 Tracking Recovery Mechanism 78
3.12 Sensor Nodes Deployment 81
3.13 Complete Single Target Tacking Algorithms 82
3.14 Biologically Inspired and Self-Organizing Aspects 84
3.15 Chapter Summary 85
Chapter 4 Multi Target Tracking in WSNs 86
4.1 Chapter Introduction 86
4.2 Target Dynamic Model 86
4.3 Sensor Detection and Measurement Model 87
4.4 Extended Kalman Filter for Multi-Target Tracking 88
4.5 Multi-Sensor Distributed Multi-Target Tracking (MS-DMTT) 90
4.5.1 Problem Formalization 90
4.5.2 MS-DMTT Framework and Assumptions 91
4.5.3 Sampling Interval Selection, Sensors Selection, Sensors Election and Recovery Mechanism 92
4.5.4 Distributed Multi-Target Selection (DMS) Algorithm 92
4.6 Multi-Sensor Adaptive Multi-Target Tracking (MS-AMTT) 93
4.6.1 Problem Formalization 94
4.6.2 MS-AMTT Framework and Assumptions 95
4.6.3 The Proposed Algorithms for MS-AMTT scheme 96
4.6.4 Adaptive Target Importance 97
4.6.5 Local Search Algorithm 99
4.6.5.1 Initial Solution Selection 99
4.6.5.2 Neighbourhood Structure 100
4.6.5.3 Complete Local Search Heuristic Algorithm 101
4.6.5.4 Computational Complexity 102
4.6.6 Main and Leader Node Election 103
4.7 Biologically Inspired and Self-Organized Aspects 103
4.8 Chapter Summary 104
Chapter 5 Task Mapping and Scheduling in WSNs 106
5.1 Chapter Introduction 106
5.2 Biological Task Mapping and Scheduling (BTMS) Algorithm 106
5.2.1 Application Model 106
5.2.2 Problem Formulation 107
5.2.3 BTMS Algorithm 108
5.2.4 Decision-Making Algorithm 112
5.2.5 Computational Complexity Analysis 112
5.2.6 Biological Inspired Aspects in BTMS Algorithm 113

5.3 A Biological Independent Task Allocation (BITA) Algorithm 113
5.4 Chapter Summary 115
Chapter 6 Simulation Environment 116
6.1 Chapter Introduction 116
6.2 Event Driven Simulation 116
6.3 Simulation Framework 116
6.4 Object Oriented Programming 117
6.5 CSMA/CA Event List 119
6.6 Chapter Summary 120
Chapter 7 Simulation Results 121
7.1 Chapter Introduction 121
7.2 Simulation Assumptions 121
7.3 MS-ASTT Scheme Evaluation 122
7.3.1 Simulation Setup 122
7.3.2 Recovery Mechanism Evaluation 123
7.3.3 Impact of Adaptive Node Election 125
7.3.4 Impact of Group Size 126
7.3.5 Comparison with other STT Schemes Using Fixed Trajectory 129
7.3.6 Impact of Adaptive Group Size 133
7.3.7 Results Discussion 134
7.4 MS-DMTT Scheme Evaluation 136
7.4.1 Simulation Setup 136
7.4.2 Sensor Nodes Selection 136
7.4.3 Tracking Error and Sampling Interval 137
7.4.4 Results Discussion 138
7.5 MS-AMTT Scheme Evaluation 139
7.5.1 Simulation Setup 139
7.5.2 Sensor Node Selection 139
7.5.3 Targets' Importance and Group Size 141
7.5.4 Tracking Update Error 143
7.5.5 Local Search Iteration 144
7.5.6 Comparison with Other Well-Known MTT Schemes 145
7.5.7 Results Discussion 146
7.6 BTMS Algorithm Evaluation 147
7.6.1 Simulation Setup 147
7.6.2 Network Node Density 148
7.6.3 Real Example of Distributed Visual Surveillance 148
7.6.4 CET and Energy Consumption using Random DAG 149
7.6.5 Network Lifetime Performance 151
7.6.5 Results Discussion 151
7.7 BITA Algorithm Evaluation 151
7.7.1 Simulation Setup 152
7.7.2 Cooperative Execution Time (CET) 152
7.7.3 The Performance Metric (Pm) 152
7.8 Code Verification 153
7.9 Chapter Summary 153
Chapter 8 Discussion and Conclusions 155
8.1 Chapter Introduction 155
8.2 Discussion 155
8.2.1 Target Tracking in WSNs 155

8.2.2 Task Mapping and Scheduling in WSNs 156
8.3 Conclusions 157
Chapter 9 Future Work 160
Appendix A Simulation Framework 162
A.1 Appendix Introduction 162
A.2 Detailed Description of Simulation Framework 162
A.2.1 Target Tracking Model 162
A.2.2 Task Mapping and Scheduling Models 164
A.3 Event Handling Pseudo Code 166
A.3.1 TArrive Event 166
A.3.2 LOCALIZATION Event 166
A.3.3 PREDICTION Event 166
A.3.4 NextSnapshot Event 167
A.3.5 Ready Event 167
A.3.6 UPDATE Event 168
A.3.7 RECOVERY Event 168
A.3.8 Wait DIFS (waitDIFS) Event 169
A.3.9 Back off (Backoff) Event 169
A.3.10 Transmit (TX) Event 170
A.3.11 Collision (waitACK) Event 171
A.3.12 Receive (RX) Event 171
A.3.13 TICK Event 172
A.4 Appendix Summary 172
Appendix B Code Verification 173
B.1 Target Tracking Verification 173
B.1.1 Analytical Analysis 173
B.1.2 Simulation Results 176
B.1.3 Multi Target Tracking and the Optimal Solution 178
B.2 BITA Algorithm Verification 179
B.2.1 Analytical Analysis 179
B.2.2 Simulation Results 180
B.3 BTMS Algorithm Verification 180
B.3.1 BTMS Algorithm and GA Algorithm 180
B.3.2 Analytical Analysis 181
B.3.3 Simulation Results 182
References 185

List of Figures

Figure 1 Single Target Tracking in WSNs....16
Figure 2 MTT in WSNs....18
Figure 3 WSN Architecture....25
Figure 4 Sensor Node Main Components....25
Figure 5 Hidden Terminal Problem....29
Figure 6 RTS/CTS Handshake....30
Figure 7 Target Tracking Stages....33
Figure 8 2D Localization....37
Figure 9 Bayesian Model....39
Figure 10 Network Framework [16]....47
Figure 11 Leader Volunteering using Voronoi Diagram[69]....48
Figure 12 The illustration of Presented Scenario in [92]....48
Figure 13 WSN Tracking Architecture [5]....49
Figure 14 WSN Tracking Network Architecture [93]....50
Figure 15 Tracking Network Structure [68]....51
Figure 16 WSN Tracking Framework [81]....52
Figure 17 The uncertainty of the Target Position [98]....55
Figure 18 Radio Consumption Model....64
Figure 19 MS-ASTT Framework in WSNs....68
Figure 20 Target Metadata....69
Figure 21 Target Location Metadata....71
Figure 22 Adaptive Sampling Interval....72
Figure 23 Sampling Interval as a Function of Location Metadata....72
Figure 24 Chemical Diffusion Strength....74
Figure 25 Adaptive Group Size Algorithm....76
Figure 26 Target Lost Scenario....78
Figure 27 Target Recovery Levels....79
Figure 28 Node Deployment....81
Figure 29 Algorithm Running in the Helper Node....82
Figure 30 Algorithm Running in the Main Node....83
Figure 31 Conflict Nodes in Multi-Target Tracking....90
Figure 32 MS-DMTT Framework in WSNs....91
Figure 33 Chemical Diffusion Strength....92
Figure 34 The DMS Algorithm....93
Figure 35 MS-AMTT WSN Framework....96
Figure 36 Algorithm Running in the Leader Node....97
Figure 37 Adaptive Target Importance....98
Figure 38 Target Importance as a Function of Location Metadata....98
Figure 39 neighbourhood Structure....101
Figure 40 Maximum Allowable Iteration as a Function of Minimum Location Metadata....103
Figure 41 An Example DAG....107
Figure 42 BTMS Algorithm....110
Figure 43 Level-Based DAG....110
Figure 44 Arrangement the Tasks in Non-increasing Order....111
Figure 45 The Decision Making Rules....112
Figure 46 Overall Simulator Structure....117
Figure 47 The Simulation World Model....118

Figure 48 CSMA/CA Event Graph 119
Figure 49 Target Trajectory using T_{max}=0.1 min and Velocity=10m/s 123
Figure 50 Number of Recovery Events Variations with T_{max} 124
Figure 51 Energy Consumption Variations with T_{max} 124
Figure 52 Total Recovery Time Variations with T_{max} 125
Figure 53 Number of Recovery Events versus Group Size with a 95% Confidence Interval 126
Figure 54 Total Recovery Time versus Group Size with a 95% Confidence Interval 126
Figure 55 Overhead Message Characteristics 127
Figure 56 Total Messages versus Group Size 127
Figure 57 Average Overhead Time versus Group Size 128
Figure 58 Energy Consumption versus Group Size with a 95% Confidence Interval 128
Figure 59 Number of Retransmissions versus Group Size with a 95% Confidence Interval 129
Figure 60 Real and Estimated Target Trajectories 129
Figure 61 Sampling Interval for Different Schemes 130
Figure 62 Location Metadata Variations 130
Figure 63 Tracking Update Error for Different Schemes 131
Figure 64 Tracking Prediction Error for Different Schemes 131
Figure 65 Energy Consumption for Different Schemes 132
Figure 66 Overhead and Recovery Times 133
Figure 67 Real and Estimated Target Trajectories 133
Figure 68 Group Size Variation 134
Figure 69 Updated Tracking Error using Adaptive Group Size 134
Figure 70 Selected Sensors considering Target Importance 136
Figure 71 Selected Sensors without considering Target Importance 137
Figure 72 (a) Real and Estimated Trajectories and (b) Sampling Interval for Different Targets 137
Figure 73 Tracking Update Error for Different Targets with Target Importance 138
Figure 74 Tracking Update Error for Different Targets without Target Importance 138
Figure 75 Selected Sensors using Closest-Sensor Selection 140
Figure 76 Selected Sensors using MS-AMTT considering Target Importance 140
Figure 77 Selected Sensors using MS-AMTT without considering Target Importance 140
Figure 78 Real and Estimated Trajectories for Different Targets 141
Figure 79 Target Importance for Different Targets 142
Figure 80 Target Metadata for Different Targets 142
Figure 81 Group Size using MS-AMTT without considering Target Importance 142
Figure 82 Group Size using MS-AMTT considering Target Importance 143
Figure 83 Location and Velocity Errors using MS-AMTT without (red curves) and with (blue curves) considering of the Target Importance 144
Figure 84 DAG of Visual Surveillance 148
Figure 85 CET versus Deadline 150
Figure 86 Consumed Energy versus Deadline 150
Figure 87 Lifetime Ratio versus Number of Tasks with a 95% confidence interval 151
Figure 88 CET versus Number of Tasks (*N*) 152
Figure 89 Performance Metric versus Node Group Membership Size (n_g) 153
Figure 90 Target Tracking Handling within the Simulator 163
Figure 91 Sending *RDis* Packets 165

Figure 92 Routing Algorithm 1 165
Figure 93 Routing Algorithm 2 165
Figure 94 LOCALIZATION Event 166
Figure 95 PREDICTION Event 167
Figure 96 NextSnapshot Event 167
Figure 97 Ready Event 168
Figure 98 UPDATE Event 168
Figure 99 RECOVERY Event 169
Figure 100 waitDIFS Event 169
Figure 101 Backoff Event 170
Figure 102 TX Event 170
Figure 103 waitACK Event 171
Figure 104 RX Event 171
Figure 105 Target Tracking Snapshot at Time= 9.6998 seconds 173
Figure 106 CSMA/CA Contention 174
Figure 107 Target Tracking Snapshot at Time 9.6998 Seconds 176
Figure 108 Simulation Snapshot: *TRang* Packet Transmission from HN1 177
Figure 109 Simulation Snapshot: *TRang* Packet Transmission from HN2 177
Figure 110 Update and Prediction Stages 177
Figure 111 Election Algorithm 178
Figure 112 Transmission of *GTrig* Messages 178
Figure 113 BITA Algorithm Verification 179
Figure 114 CET versus Number of Tasks Simulation Results 180
Figure 115 BITA: Simulation Results for N=100 180
Figure 116 Level-Based DAG for Code Verification 181
Figure 117 BTMS Analytical Analysis 182
Figure 118 Task Generator Results 183
Figure 119 Decomposition Fitness Function Results 183
Figure 120 Task Mapping 184
Figure 121 Summary of BTMS Results 184

List of Tables

Table 1 CSMA/CA FHSS Parameters 122
Table 2 Lifetime, Load Balancing Performance and Energy Consumption for Different δ Values 125
Table 3 Recovery Results for Different Schemes 132
Table 4 Average Location and Velocity Errors 144
Table 5 Average Iterations, Computational Time and Tracking Loss 144
Table 6 Average Iterations and Computational Time 145
Table 7 Average Location and Velocity Errors 145
Table 8 Average Iterations and Computational Time using 500 Sensors 146
Table 9 Average Iterations and Computational Time using1000 Sensors 146
Table 10 Results for Visual Surveillance DAG 149
Table 11 Target Estimated Locations Database 175
Table 12 Election Algorithm Information 175

Glossary of Terms

2D	Two Dimensions
3D	Three Dimensions
ACK	Acknowledge
ACO	Ant Colony Optimization
AGF	Approximate Grid-based Filter
APTEEN	Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network
BS	Base Station
BITA	Biological Independent Task Allocation
BTMS	Biological Task Mapping and Scheduling
BU	Bottoms Up
CDMA	Code Division Multiple Access
CET	Collaborative Execution Time
CH	Cell Head or Cluster Head
CoRAl	Collaborative Allocation Algorithm
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CTS	Clear-to-Send
CW	Contention Windows
DAG	Directed Acyclic Graph
DBF	Distributed Bellman-Ford
DCA	Distributed Computing Architecture
DIFS	Distributed Inter-Frame Space
DMS	Distributed Multi-target Selection
DOA	Direction of Arrival
DSDV	Destination Sequenced Distance Vector Routing
DTW	Dynamic Time Warping
DVS	Dynamic Voltage Scaling
EcoMapS	Energy-Constrained Task Mapping and Scheduling
EKF	Extended Kalman Filter
FDMA	Frequency Division Multiple Access
FHSS	Frequency-Hopping Spread Spectrum

FSM	Finite-State Machine
FTM	Fast Tracking Mode
GA	Genetic Algorithm
GAF	Geographic Adaptive Fidelity
GEAR	Geographic and Energy Aware Routing
GF	Grid-based Filter
GOAFR	Greedy Other Adaptive Face Routing
GPS	Global Positioning System
HMM	Hidden Markov Model
HN	Helper Node
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
KF	Kalman Filter
LADAR	LAser Detection And Ranging
LB	Lower Bound
LEACH	Low Energy Adaptive Clustering Hierarchy
LIDAR	Light-Imaging Detection And Ranging
LOS	Line of Sight
LWT	Levelized Weight Tuning
MAC	Media Access Control
MACA	Multiple Access Collision Avoidance
MACAW	Multiple Access Collision Avoidance with Acknowledgment
MCC	Mega Clock Cycles
MCFA	Minimum Cost Forwarding Algorithm
MECN	Small Minimum Energy Communication Network
MN	Main Node
MS-AMTT	Multi-Sensor Adaptive Multi-Target Tracking
MS-ASTT	Multi-Sensor Adaptive Single Target Tracking
MS-DMTT	Multi-Sensor Distributed Multi-Target
MTMS	Multihop Task Mapping and Scheduling
MTT	Multi Target Tracking
NFA	Nondeterministic-Finite-State Automaton
NP	Nondeterministic Polynomial-time

OSI	Open System Interconnection
PDA	Personal Digital Assistant
PDF	probability density function
PDP	Predicted Detection Probability
PEGASIS	Power-Efficient Gathering in Sensor Information Systems
PF	Particle Filter
PHY	Physical
PIR	Passive Infrared
PSO	Particle Swarm Optimization
RADAR	RAdio Detection And Ranging
RFID	Radio frequency Identification
RX	Receive
RNG	Random Number Generator
ROI	Region of Interest
RSSI	Received Signal Strength Indication
RT-MapS	Real-time Task Mapping and Scheduling
RTS	Request-to-Send
SIFS	Short Inter-Frame Space
SL	Simplified Lagrangian
S-MAC	Sensor-Media Access Control
SONAR	SOund Navigation And Ranging
SPIN	Sensor Protocols for Information via Negotiation
SRC	Source Address
SSM	State Space Model
STT	Single Target Tracking
TDMA	Time Division Multiple Access
TDNN	Time-Delay Neural Network
TDOA	Time Delay of Arrival
TEEN	Threshold-sensitive Energy Efficient sensor Network
TMM	Track Maintenance Mode
TMS	Task Mapping and Scheduling
WSN	Wireless Sensor Network

Chapter 1 Introduction

1.1 Chapter Introduction

This chapter provides a brief introduction about the research topic. The problem definition is firstly established. After this the motivation and objectives of the research are summarized. Next, the main contributions of this thesis are summarised. The thesis structure and the publications are finally presented.

1.2 The Research Problem Definition

The problem of energy-efficient collaborative single and multiple target tracking in Wireless Sensor Networks (WSNs) is considered for sensor nodes with limited energy resources and abruptly manoeuvring targets of different importance. Additionally, the problem of task mapping and scheduling in WSNs is also considered.

The biological aspect of this research is to treat the target as a virtual chemical emitter and to construct influence contours whose strength decreases with distance from the target. The nodes that are influenced the strongest are more likely to be chosen to track the target. Furthermore, as with differentiation observed in biological zygotes, the sensor group differentiates, with specific nodes specializing to perform the required functionalities.

In following sections, the problems of target tracking, task mapping, and scheduling in WSNs are examined in more detail.

1.2.1 Brief Introduction about WSNs

Wireless Sensor Networks (WSNs) have become an emerging phenomenon in industry, both for civil and military purposes. WSNs provide virtual snapshots of the physical world by interpreting the physical events. WSNs consist of tiny electronic nodes connected to each other via wireless communication protocols [1]. Each node is equipped with embedded processors, sensor devices, storage, and radio transceivers. Nevertheless, the sensor nodes typically have limited resources in terms of battery-supplied energy, processing capability, communication bandwidth, and storage [2][3]. WSNs have attractive commercial applications such as healthcare, target tracking, monitoring, smart homes, surveillance and intrusion detection [4].

1.2.2 Target Tracking in WSNs

Target tracking in WSNs is a process of estimating the location, trajectory, velocity and/or acceleration of a mobile target. It often needs accurate estimation and prediction of the target state. Collaborative target tracking uses a multi-sensor scheme to improve the tracking accuracy compared with single-sensor tracking [5]. Figure 1 shows the Single Target Tracking (STT) scenario in WSNs. The Base Station (BS) or sink is responsible for forwarding the desired information from the WSN to the headquarters (i.e., main controller) through the Internet, via satellite or other wireless technology. The target can be a human being, moving vehicle, animal, tank, enemy or any interesting object that needs to be tracked. The target is usually mobile. Target dynamics is the mathematical modelling of the target motion. Targets can move in unexpected manner and this causes noise in the dynamic model. Target state is the location, velocity and/or the acceleration. The trajectory of the target is the path that target draws during its travel. Calculating the mobile target state and trajectory in the presence of noise in its dynamic characterisation is one of the main challenges for target tracking in WSNs.

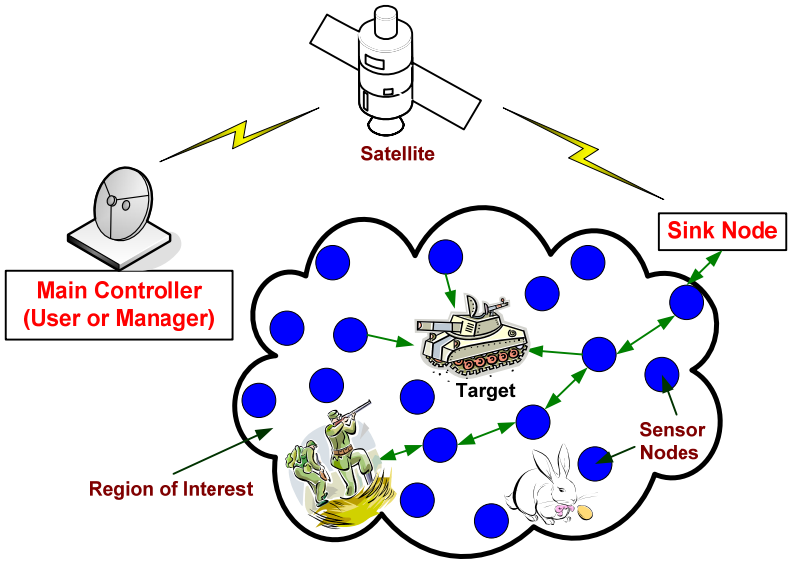


Figure 1 Single Target Tracking in WSNs

Each sensor node has a sensor device to sense or detect the presence of the target in the Region of Interest (ROI). Detection of the target is always handicapped by the presence of noise. Therefore, using the noisy sensor readings to calculate the target state is a further challenge. Due to the limited battery-supplied energy of the sensor nodes and the difficulty to physically access them, energy-efficient target tracking is a crucial aim. Additionally, hundreds or thousands of sensor nodes are deployed in the ROI. Thus,

several sensor nodes may detect the target at the same time. Therefore, selecting the necessary sensor nodes to track the target is a common problem in target tracking in WSNs. To reduce the energy consumption, sensor nodes are scheduled to be in active or sleeping modes. However, the target is mobile and requires sensor nodes to be in active mode to detect and calculate its state. Therefore, predicting the future state of the target, to proactively form the group of necessary sensor nodes to continue the tracking and how to allocate duties within the group are pertinent research problems.

The tracking sampling interval or resolution is defined as the time between two successive tracking events. If the sampling interval is set too large the tracking accuracy, which indicates about the difference between the real and estimated states, is degraded and the target may be unmonitored for long periods. Moreover, the target may be lost if it travels in an unpredictable manner. On the other hand, decreasing the sampling interval leads to increase the energy consumption because the tracking events will be increased. Therefore, choosing a suitable sampling interval during the tracking process is challenging. Furthermore, the tracking system should support to a mechanism to recover the target state in the case of target loss.

Designing tracking schemes for Multi-Target Tracking (MTT) is more complex than considering STT. Figure 2 shows a MTT WSN scenario. Targets can travel with different movement patterns. Some targets may move in a uniform and predictable fashion whilst others manoeuvre in a random manner. Therefore, for MTT tracking continuity and tracking accuracy robustness sensor selection, the management of the group of targets, the sampling interval calculation for each target, are additional challenges that must be addressed.

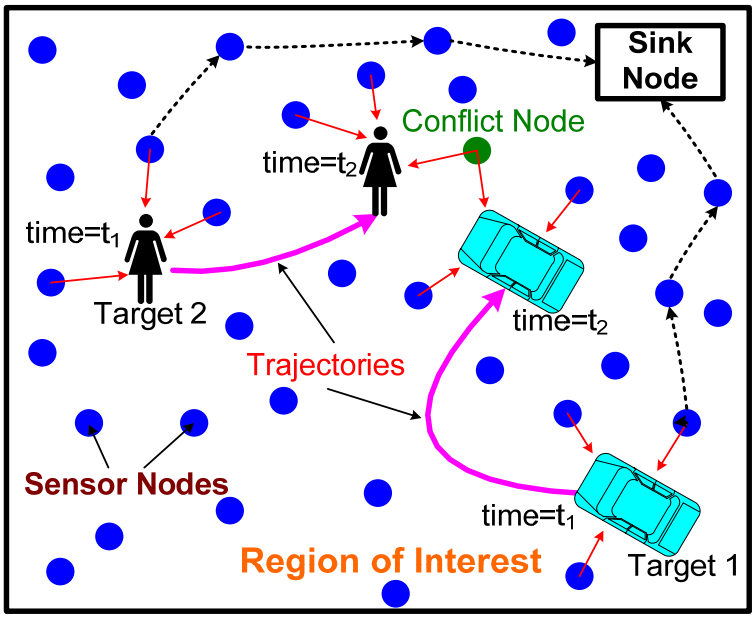


Figure 2 MTT in WSNs

In MTT, targets can be assigned different importance according to particular criteria. Finding a technique to evaluate the target importance is an interesting issue in MTT. As shown in Figure 2, if a sensor node detects more than one target, it has to decide which target it will serve. We refer to this sensor node as being a "conflict" node.

1.2.3 Task Mapping and Scheduling in WSNs

Many WSNs applications such as target tracking and camera-based applications [6] require real time execution, sensor node collaboration and computationally intensive operations. Since an individual sensor node does not have the enough processing power and possibly battery life to execute a complex application and meet the application deadline, one solution to execute the complex application using a group of sensor nodes. Task mapping assigns resources to tasks and task scheduling determines the execution sequence of the tasks, to try to maximize performance objectives. It is well known that optimal task mapping is an NP-complete problem [7]. Therefore, heuristic techniques are needed to obtain near optimal solutions. In high performance computing [8], task mapping and scheduling are deeply explored. However, the design objectives for WSNs are different due to the limitations of the resources.

1.3 The Research Motivation

As physical limitations of sensor nodes in terms of battery-supplied energy, processing performance, communication bandwidth, and storage become main challenges in

designing WSNs, this research is explores cooperative target tracking given these constraints. Briefly, the following points are the main requirements for this research:

- Improve the energy-efficiency and network lifetime for target tracking in WSNs
- Maintain tracking accuracy and reliability
- Accommodate the random motion of targets
- Provide support for target importance or priority
- Reduce the execution time of complex applications in WSNs

1.4 The Research Objectives

The main objectives of this research are listed below:

(1) Develop a reliable, accurate, energy-efficient, collaborative and self-organized target tracking scheme in WSNs.

- Design WSN framework for multi-sensor target tracking
- Develop target tracking scheme in WSNs
- Develop an adaptive sampling interval mechanism
- Use adaptive sensors selection
- Implement adaptive group election
- Develop a target recovery scheme to recapture lost targets
- Provide a sensor node density calculation
- Support both STT and MTT
- Support the target importance in MTT
- Optimize or nearly optimize sensor selection in the case of MTT
- Tackle the problem of conflict nodes in the case of MTT

(2) Design algorithms for task mapping and scheduling in WSNs to parallelize the execution of an application among a group of sensor nodes.

- Develop an algorithm to execute an application across a group of sensor nodes. The application is assumed to be divided into independent equal-weighted subtasks
- Develop an algorithm to execute an application across a group of sensor nodes. The application is assumed to be decomposed into smaller tasks with different computation weights and dependencies

1.5 Novelty & Contributions

This thesis proposes a novel framework to design biologically inspired self-organized communication networks. However, this thesis explores WSNs applications more widely and uses target tracking as an example. The thesis makes the following unique contributions:

1. This research introduces the formalization of target metadata pertaining to the target's past locations, by which the movement pattern of the target is computed. Target metadata is employed to adaptively calculate the tracking sampling interval, the targets' importance in the case of MTT and the number of local search iterations for the local search algorithm used in MTT.
2. This thesis introduces the first formulation associated with the conflict node concept in MTT taking into account target importance. Novel strategies for choosing the initial solution and neighbourhood structure are proposed in this thesis for the local search to solve in real-time the combinational optimization problem of sensor selection in MTT.
3. This research introduces an energy-efficient framework for STT and MTT in WSNs. Adaptive sensor selection and "leader node" election algorithms are proposed. A mechanism is developed to recover the tracking process in the event of of target loss.
4. Two algorithms are introduced for task mapping and scheduling in WSNs. The first algorithm assumes that the application can be divided into independent equally-weighted subtasks. The other assumes that the application can be decomposed into smaller tasks with different computation weights and dependencies.
5. The principle of differentiation found in biological zygotes is applied to the proposed tracking, task mapping and scheduling schemes. Furthermore, this is the

first research to treat the target to be served by the WSN as a virtual chemical emitter that has different influence strengths on the sensor nodes.

1.6 Notation Conventions

In this thesis, matrices and vectors are denoted by bold letters. Variables and functions are denoted by italic letters. The transpose of matrix $\mathbf{A} = [A_{ij}]$ is denoted by $\mathbf{A}' = [A_{ji}]$ where A_{ij} is the element at row i and column j. The inverse of the matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{-1}$. The diagonal matrix of $\mathbf{A}$ is denoted by $diag(\mathbf{A})$. The identity matrix is denoted by $\mathbf{I}$. The expectation of a random variable x is denoted by $E[x]$.

1.7 Thesis Structure

This thesis is organized as follows:

Chapter 2 presents the background and literature review of the research. It includes a general introduction about sensor network. Then, details about wireless routing and Media Access Control (MAC) protocols are discussed. WSNs target tracing techniques, frameworks and stages are presented. After that, biologically inspired researches and self-organised networks are introduced. State-of-the art in literature concerning biological inspired systems, target tracking in WSNs, and task mapping and scheduling in WSNs are explored.

Chapter 3 introduces the proposed Single Target Tracking (STT) scheme in details. The target dynamic, sensor detection, measurement and energy consumption models are presented. Then, the Extended Kalman Filter (EKF) for STT in WSNs is introduced. The framework and the assumptions for the proposed STT scheme are explained. The target metadata representation is illustrated. The sampling interval, and sensor nodes selection and election are presented. Recovery mechanism and sensor nodes deployment strategies are introduced. Complete algorithms and protocols for the proposed STT scheme are proposed.

Chapter 4 introduces the Multi-Target Tracking (MTT) in WSNs. Two proposed MTT schemes in WSNs are introduces. Firstly, a Multi-Sensor Distributed Multi-Target Tracking (MS-DMTT) scheme is proposed based on the assumption that the sensor node can only detect and serve a single target at the same time. Secondly, a Multi-Sensor Adaptive Multi-Target Tracking (MS-AMTT) scheme is introduced based on the

assumption that the sensor node can detect and serve more than one target at the same time.

Chapter 5 presents the Task Mapping and Scheduling (TMS) in WSNs. Firstly, a Biological Task Mapping and Scheduling (BTMS) algorithm is proposed. In BTMS algorithm, the application is assumed to be decomposed into dependent tasks with different computation weights. Secondly, Biological Independent Task Allocation (BITA) algorithm is introduced. In BITA algorithm, the application is assumed to be decomposed into equal-weighted independent tasks.

Chapter 6 explains the simulation models used to evaluate the proposed target tracking and TMS in WSNs. Event driven simulation is introduced. The main simulator flow chart and used random number generator are presented. The data structure and different event types with their pseudo code are discussed. Appendix A is included at the end of this thesis to explain the simulation events and framework in details.

Chapter 7 proposes the performance and evaluation of the MS-ASTT, MS-DMTT, MS-AMTT, BTMS and BITA schemes that proposed in Chapter 3, 4 and 5. A critical assessment and discussion for the simulation results are also provided. Additionally, the proposed schemes are compared against well-known schemes. Appendix B is included at the end of this thesis to verify the proposed simulation.

Chapter 8 provides critical discussions for the presented results. It also summarizes this thesis, the results and the original contributions of this research.

Chapter 9 presents the future work in target tracking, task mapping, and scheduling in WSNs.

1.8 Authorship

The following research publications have been published or submitted by the author.

Journal Papers

1. Yousef E. M. Hamouda and Chris Phillips, "*Adaptive Sampling for Energy-Efficient Collaborative Multi-Target Tracking in Wireless Sensor Networks*", IET Wireless Sensor Systems, 2011, Accepted for publication.
2. Yousef E. M. Hamouda and Chris Phillips, "*Metadata Based, Optimal Sensor Selection for Multi-Target Tracking in Wireless Sensor Networks*", International

Journal of Research and Reviews in Computer Science, 2010, Accepted for publication.

Conference Papers

1. Yousef E. M. Hamouda and Chris Phillips, "*Metadata-Based Adaptive Sampling for Energy-Efficient Collaborative Target Tracking in Wireless Sensor Networks*", The 10th IEEE International Conference on Computer and Information Technology (CIT 2010), Bradford, UK.
2. Yousef E. M. Hamouda and Chris Phillips, *"Biological Task Mapping and Scheduling in Wireless Sensor Networks"*, 2009 IEEE International Conference on Communication Technology and Applications (ICCTA2009), pp. 914-919, Beijing, October 2009.
3. Yousef E. M. Hamouda and Chris Phillips, *"Biologically Inspired, Cooperative Target Tracking Framework for Wireless Sensor Networks*", LCS 2009, University College London, September 2009.
4. Yousef E. M. Hamouda and Chris Phillips, *"Biologically Inspired, Self Organizing Communication Networks"*, PGNet2007 & EPRC, June 2008.

Chapter 2 Background and Literature Review

2.1 Chapter Introduction

In Chapter 1, the main contributions and motivations of this research are identified. This chapter explores the background and state-of-the art related to Wireless Sensor Networks (WSNs), biological inspired target tracking, task mapping and scheduling. It commences with an introduction to WSNs and their application. Then, Media Access Control (MAC) and routing protocols are addressed and target tracking in WSNs is examined. After that, biologically inspired self-organising networks are introduced. The state-of-the art in literature concerning biological inspired systems, target tracking in WSNs, task mapping and scheduling are considered. Finally, the chapter is summarised.

2.2 Wireless Sensor Networks (WSNs) Overview & Applications

WSNs are receiving much attention in industry, both for civil and military purposes. WSNs provide virtual snapshots of the physical world by interpreting the physical events. As shown in Figure 3, WSNs consist of electronic network nodes connected to each other via wireless communication protocols [9][10]. WSNs have many advantages such as easy random deployment, low-cost and small-size. WSNs contain hundreds or thousands of tiny sensor nodes that are scattered in the sensor field which is the area in which the sensor nodes are deployed. Deployment of the sensor nodes can be either in random fashion such as in disaster situation where for example sensor nodes are dropped from an airplane [9] or in planned manner such as deployment of WSNs in smart homes or for fire alarm systems. The Base Station (BS) or sink is responsible for forwarding the desired information from the WSN to the headquarters (i.e., main controller) through the Internet, via satellite or other wireless technology. Most of WSNs have fixed sensor nodes and BS. However, mobility of sensor nodes or BS is desirable in many applications [11]. The sensor nodes cooperate [12] together to sense, compute and transmit the information from harsh physical environments to external BS or sink. For example, the sensor nodes cooperate to localise the target shown in Figure 3. Each sensor node can collect and route the data either to other sensor nodes or back to the sink node via the path between them [13][14].

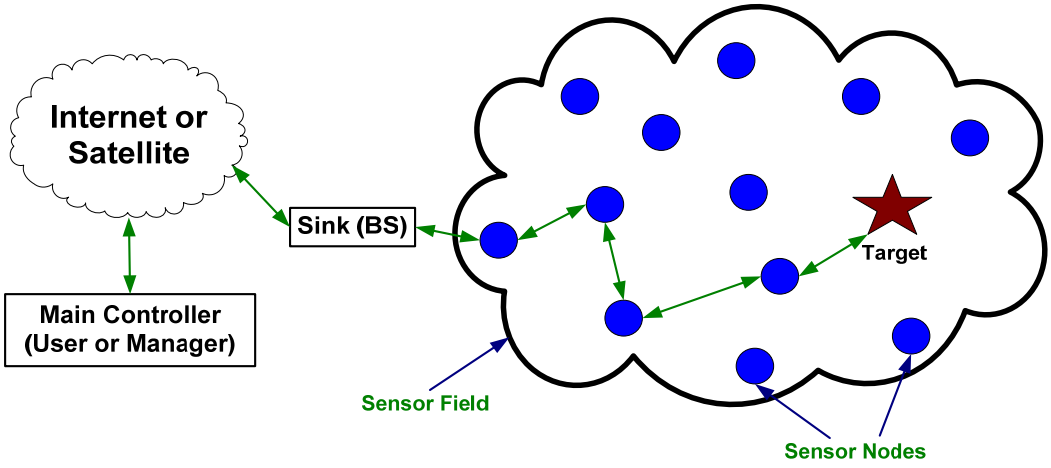


Figure 3 WSN Architecture

As shown in Figure 4, each sensor node is equipped with an embedded processor and storage to process the data, sensor devices to measure ambient conditions related to the environment surrounding the node and transform them to electrical signals, and radio transceivers to send and receive electromagnetic waves. Nevertheless, the sensor nodes have very limited resources in terms of battery-supplied energy, communication bandwidth, and computational processing and storage capabilities [2][3]. Therefore, sensor nodes are typically cheap devices.

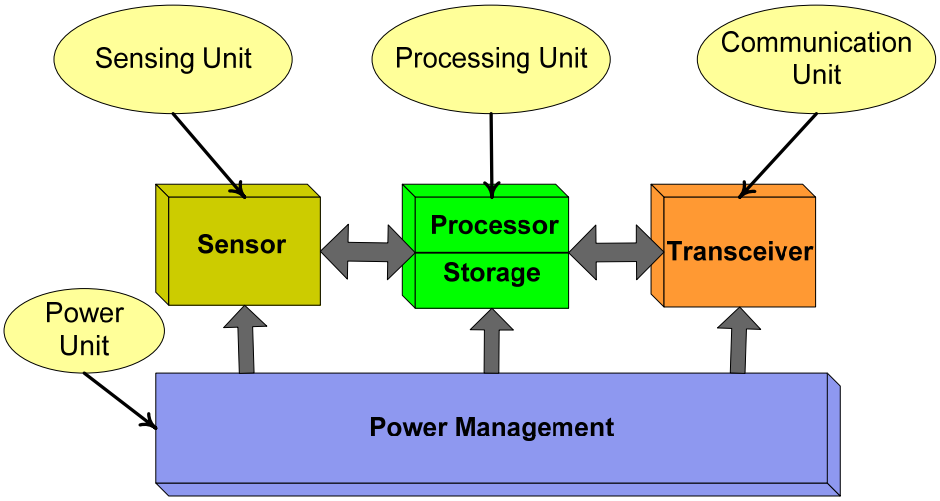


Figure 4 Sensor Node Main Components

Nowadays, commercial and industrial fields employ WSNs for a wide range of applications such as healthcare, machine condition monitoring, environmental monitoring including pollution monitoring, surveillance of people or vehicle (e.g. access control, crowd flux statistics, congestion analysis, anomaly detection, bio-chemical material detection such as diffused poison gas, alarming and person-specific identification [15]), structural monitoring, navigation and control of moving vehicle, wildlife habitat monitoring, tracking the movement of wild animals in wildlife

preserves, forest fire, manufacturing job flow, home applications (e.g. smart homes), detecting environmental ambient conditions (e.g. temperature, movement, sound, light, activity and the presence of certain objects), inventory control, weather monitoring, Single Target Tracking (STT), Multi-target Tracking (MTT) and disaster management [1][16]. WSNs can also be used in military applications including target field imaging, intrusion detection, enemy vehicles, detecting illegal crossings, security and tactical surveillance.

2.3 Energy Consumptions Factors in WSNs

WSNs are usually deployed in harsh environments such as space, forests and battlefields. Therefore, it is difficult to physically access the wireless sensor nodes after deployment. In many cases, it is impossible to change or recharge the depleted sensor node battery [17]. Therefore, maintaining battery life as long as possible is one of the most crucial issues in WSNs because it increases the useful network lifetime [18]. Energy is consumed from the battery during sensing, communication and processing. Sensor nodes wastes energy due to reasons outlined in [17][19][20], namely:

(1) **Collisions:** The collision takes place when two sensor nodes within the same coverage area transmit packets at the same time (i.e., full collision). Therefore, the two packets interfere with each other at the receiving sensor node which cannot distinguish between them. However, a collision can also happen if one sensor node transmits packets before the current transmitting sensor node finishes its transmission. This is called a partial collision. In these collisions, the receiver will discard both packets because they will be corrupt. Therefore, both of the transmitting sensor nodes will try to retransmit again which increases the energy consumption.

(2) Idle Listening: Basically, each sensor node in WSNs can be in active, idle or sleep modes. In active mode, a sensor node consumes energy in transmitting or receiving data. In idle mode, the sensor node consumes energy to listen to the channel. In the sleep mode, the sensor node sleeps and turns off the radio transceiver. The sensor node in the idle state listens to the channel status to initiate transmission or to wait for traffic from other sensor nodes. Thus, sensor nodes consume energy in channel listening. According to the measurements obtained by Katz and Stemm [21], the idle: receiving: transmission power consumption ratios are 1:1.05:1.4 on 915MHz.

(3) Over-hearing: The sensor node may receive the packets destined for others which is considered another energy wasting factor.

(4) Protocol Overhead: These are the control packets used by different communication protocols. For example, contention-based protocols such as Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) use control packets to manage channel access and reduce the collisions. In contrast, with collision-free or scheduled protocols such as Time Division Multiple Access (TDMA), the control traffic is less.

(5) Over-emitting: Energy is wasted when a sensor node sends packets although the receiver is not ready to receive them.

Many methods have been adopted to minimize the consumed energy from WSNs such as designing energy efficient MAC protocols [17], routing protocols and power control. However, this research tackles this issue by: (1) controlling the sensor node activity by scheduling sleep, idle and active modes for the sensor nodes, (2) controlling the processing demands and time by executing applications only when necessary, (3) parallelizing the execution of an application among a group of sensor nodes, and (4) designing energy-efficient protocols and algorithms that use low overhead packets. The use of an adaptive sampling interval in STT and MTT, adaptive MTT, and biologically task mapping and scheduling, introduced in Chapter 3, 4 and 5, address these issues together.

2.4 WSN MAC Protocols

Sensor nodes within communication range share the same physical channel or medium. Medium Access Control (MAC) protocols have been developed to coordinate the channel access and thus to avoid the collisions resulting from two sensor nodes accessing the same medium to send packets at the same time. MAC is a sub-layer of data link layer of the Open Systems Interconnection (OSI) model. MAC protocols let the sensor nodes decide when and how to access the channel. In the literature, this mechanism is also called channel allocation or multiple access.

Broadly speaking, MAC protocols are categorised into two classes which are collision-free or scheduled and contention-based protocols. Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA) and Code Division Multiple Access (CDMA) are examples of collision-free MAC protocols. The basic

concept of their operation is to avoid collisions and interference by assigning users or sensor nodes to separate sub-channels which are separated by time, frequency or orthogonal codes [22][23]. On the other hand, sensor nodes in contention-based protocols compete for access to the shared medium by using probabilistic coordination. Therefore, collisions may arise with these protocols. Two common contention-based protocols are ALOHA [24] and CSMA [25]. In ALOHA, a sensor node starts transmission when needed without any coordination and it reschedules another transmission in the event of a collision. There are two main types of ALOHA protocol, namely: slotted ALOHA, in which sensor node transmits at the next available slot, and pure ALOHA, in which the sensor node transmits a packet as soon as it is generated. On the other hand, a CSMA protocol senses the channel before transmitting. If the channel is busy, the sensor node delays access and retries later. Several extensions have been developed for CSMA to support different environmental conditions. Furthermore, CSMA is adopted for IEEE 802.11 [26]. As mentioned in Section 2.2, WSNs differ from traditional wireless networks because WSNs have limited battery energy of the sensor node and WSNs have to be employed large number of sensor nodes in ad hoc fashion. Therefore, MAC protocols have to consider collision avoidance, energy efficiency, scalability of the WSN and adaption to network topology changes due to sensor node death and movement [19]. CSMA is an important contention-based MAC protocol because it is considered the basic approach for contention-based MAC protocols for use in WSNs. The main principle of its operation is to listen before attempting transmission. In [27], the performance of CSMA is assessed for WSNs. Basically, WSNs are multi-hop wireless networks. Two well-known problems in multi-hop wireless networks are the hidden terminal and exposed terminal problems [28][154]. Exposed terminal problem occurs when a node can not send packets because one of its neighbours is transmitting. In Figure 5, the hidden terminal problem is explained. Assume three sensor nodes, node 1, 2 and 3, form a two-hop wireless network, in which node 1 and 2 are neighbours, and node 2 and 3 are neighbours. When node 1 sends data to node 2, node 3 will not hear that transmission. Therefore, node 3 may start sending at the same time in which node 1 is sending to node 2 (i.e., resulting in a Full Collision) or during the transmission of node 1 to node 2 (i.e., Partial Collision). In both cases, node 2 will received collided or corrupted packets. Because to this problem, CSMA/CA [29], where CA refers to Collision Avoidance has been developed.

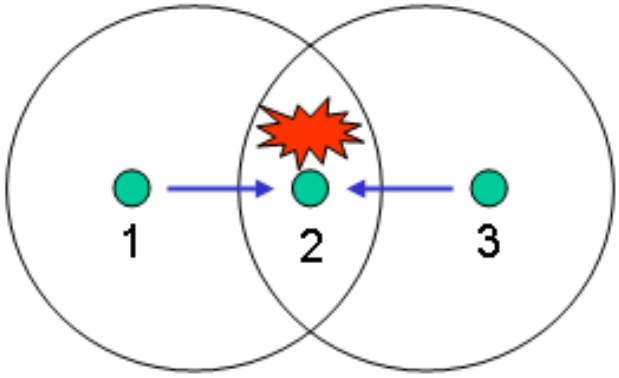


Figure 5 Hidden Terminal Problem

As shown in Figure 6, CSMA/CA uses a short handshake between the source and the destination before the actual transmission. The source (i.e., node 2) sends a Request-to-Send (RTS) packet to the destination (i.e., node 3). The source's neighbours (i.e., node 1) hear the RTS packet and they defer their own transmission until the current transmission finishes. When the destination receives an RTS, it replies with a Clear-to-Send (CTS) packet. Like before, the destination's neighbours (i.e., node 4) hear the CTS packet and they defer their own transmission until the current transmission is finished. After that, the source starts to send the actual data. However, CSMA/CA does not address completely the hidden terminal problem because the collision may happen on RTS packets. Since an RTS packet is very short, the collision from RTS will be very short as well.

Multiple Access Collision Avoidance (MACA) [30] uses the same concept of CSMA/CA but it adds a duration field in RTS and CTS packets to indicate the amount of time to transmit the data so that the neighbours of the sender have to wait this amount of time before their own transmissions. Further improvements have been carried out with the MACA protocol and a new protocol called Multiple Access Collision Avoidance with Acknowledgment (MACAW) [31] has been developed. MACAW adds an acknowledgement (ACK) packet after each actual data packet transmission. Therefore, the handshake between source and destination in MACAW is RTS-CTS-DATA-ACK. In Figure 6, the destination replies by acknowledging the source for each data packet received from the source. IEEE 802.11 [26] has adopted all the features of CSMA/CA, MACA and MACAW in its MAC layer.

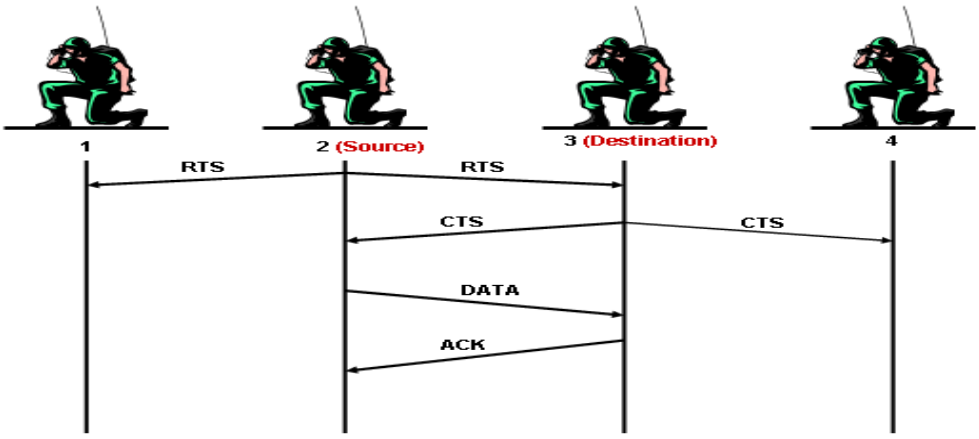


Figure 6 RTS/CTS Handshake

Sensor-MAC (S-MAC) is a MAC protocol designed especially for WSNs [32][33]. It is a contention-based protocol that has been developed to improve the energy efficiency in multi-hop WSNs. S-MAC has mechanisms to reduce the energy wastage from the consumption factors mentioned in Section 2.3.

In this thesis, CSMA/CA will be used as MAC protocol in the simulation models to evaluate the proposed tracking, task mapping and scheduling schemes. Further details are introduced in Chapter 7 and Appendix A.

2.5 WSNs Routing Protocols

Routing protocols are the algorithms used for the Layer-3 of the OSI model. They determine the best path or route between the source and the destination. In Internet Protocol (IP) networks [34], routers are usually fixed and provide the routing information to fixed clients as well. On the other hand, ad hoc networks and some WSNs [35][36] are infrastructure independent in which each sensor node can operate as a client and a router. Sensor nodes in ad hoc networks and some WSNs can move randomly and the network topology dynamically changes. Therefore, mobility is the main design consideration for routing protocols in ad hoc networks and WSNs. Other design challenges for routing protocols in ad hoc networks and WSNs include the limited wireless bandwidth and sensor node resource constraints. However, routing in WSNs is very challenging [13][14] due to their needed requirements. First, it is impractical to build global addressing scheme like the IP-based one due to the large number of sensor nodes that make the addressing overhead too great. Therefore, in WSNs, getting the data is more important than knowing the addresses of which sensor nodes sent the data. Second, a sensor node has very limited resources in term of battery

and processing power. Therefore, very careful resource management is required to design routing protocols for WSNs. Third, WSNs are designed based on the required application. For example, a tactical surveillance application design is different from a periodic weather monitoring. Fourth, data collected by many sensor nodes in particular WSN is based on common phenomena. Therefore, data redundancy can arise. Finally, WSNs are location aware because the data collection normally depends on position. Global Positioning System (GPS) [37] hardware is not feasible to be equipped in all sensor nodes due to sensor node resource constraints. Techniques based on triangulation that, for example, allows sensor nodes to determine approximately their positions using radio strength, triangulation or multilateration [38] perform quite well under conditions where a few sensor nodes know their position using, for example, GPS.

Routing protocols are classified into proactive, reactive and hybrid protocols based on how and when the source searches for a route to the destination. In proactive protocols, each sensor node has the routes to all destinations regardless of whether or not it is needed, while in reactive protocols the routes are computed on demand when they are needed. Hybrid protocols are a combination of these two ideas. In addition, WSN routing protocols can be classified according to the network structure as flat, hierarchical, or location-based [13]. The flat architecture introduces a fully peer-based distributed network where each terminal acts as an ordinary sensor node and a gateway at the same time. Therefore, all the sensor nodes play the same roles. Data centric routing is used in flat protocols instead of using a global identifier to each sensor node. The BS sends queries to sensor nodes located in certain regions and waits for the reply from the sensor nodes. The data is specified using attribute-based naming that includes the data properties. There are many flat routing protocols including Sensor Protocols for Information via Negotiation (SPIN) [39] and [40], Directed Diffusion [41], Rumor routing [42] and The Minimum Cost Forwarding Algorithm (MCFA) [43]. On the other hand, the hierarchal architecture or cluster-based routing classifies the sensor nodes into ordinary nodes, cluster head nodes or gateway nodes. The main function of the cluster node is to control the other nodes inside the cluster and relay the traffic within the cluster. The gateway node connects the clusters together to relay or forward the data and control traffic between clusters. The routing information and overhead can be reduced in the case of hierarchal architectures; especially in large-scale networks [44]. Therefore, lifetime and energy efficiency can be improved. The cluster head is selected

as a higher energy sensor node. It processes and sends the data. The other sensor nodes in the cluster perform the sensing. Hierarchal routing is typically a two layer routing scheme where one layer selects the cluster head and the other is for routing. There are many hierarchal routing protocol including Low Energy Adaptive Clustering Hierarchy (LEACH) [18], Power-Efficient Gathering in Sensor Information Systems (PEGASIS) [45], Threshold-sensitive Energy Efficient sensor Network (TEEN) [46], Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network (APTEEN) [47] and Small Minimum Energy Communication Network (MECN) [48]. The location-based routing protocols employ sensor node position information to build the routing table. There are many location-based routing protocol including Geographic Adaptive Fidelity (GAF) [49], Geographic and Energy Aware Routing (GEAR) [50] and The Greedy Other Adaptive Face Routing (GOAFR) [51].

However, Destination Sequenced Distance Vector routing (DSDV) [52], which is a proactive ad hoc routing protocol, has been implemented in the simulation model. It is based on classical Distributed Bellman-Ford (DBF) algorithm. In DBF, each sensor node maintains the first sensor node (hop) on the shortest path to every other sensor node in the network. Each sensor node maintains routing table for all possible destinations and the number of routing hops to reach that destination. A sequence numbering system (labelling the routes) is used to differentiate stale routes from the new routes.

2.6 WSNs Target Tracking

Target tracking is one of the most useful and used applications in both civil and military applications. The main purpose of target tracking is to monitor the location of the target in two or three-dimensional coordinates [53]. In automated visual or video surveillance, target tracking goals are advanced to not only determine the target's location but also to obtain a description of what is happening in the region of interest (ROI) and then to perform suitable actions according to the interpretation obtained from the ROI [54]. Therefore, the tracking system can detect abnormal behaviour and hostile intent. Two popular target tracking infrastructures in WSNs are a camera-based approach which relays on image analysis and computer vision [55] and an acoustic-based one [56] which uses the strength of acoustic signal received from the target to calculate the target range and direction angle. Some systems mix both approaches. Generally as shown in

Figure 7, every target tracking system includes some or all of the following stages: target detection, target classification, nodes selection, group election, target localization, target tracking, behaviour and activity analysis, personal identification, and handover [15][54][57].

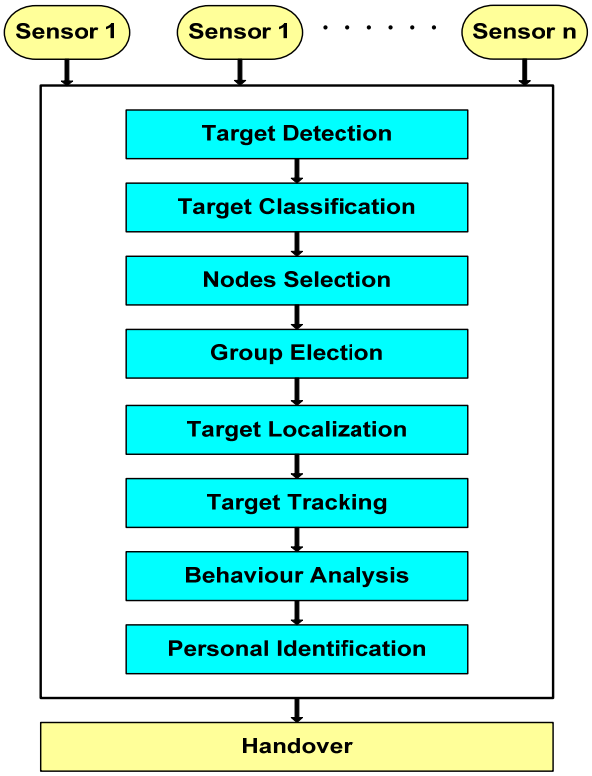


Figure 7 Target Tracking Stages

2.6.1 Target Detection

The main objective of detection is to detect the presence of the target in the ROI by sensor networks. Therefore, the system has to discriminate between the target absence and presence. Loosely speaking, targets emit signals characterized their presence in the ROI which can be sensed from sensor nodes [58]. The sensors can be classified based on the type of measurement information or modalities they read from the world (i.e., target's signal emissions). Passive sensors detect the target using target's natural energy. Vision-based, magnetic-based, seismic-based, thermal-based and acoustic-based sensors are passive sensors [59][60]. More advanced detection techniques combine these methods such as using camera and microphone arrays (i.e., audiovisual sensor network) detection [61]. For example, vision-based or camera-based sensors work similarly like human eyes through using the electromagnetic spectrum to generate the image. In vision-based WSNs [15][54][55], target detection is achieved using motion and object detection which aims to separate the region corresponding to the moving

target from the rest of the image. Motion and object detection requires environmental (i.e., background) modelling and motion segmentation. Temporal differencing, background substation and optical flow are the most used approaches for motion segmentation. In acoustic-based sensor networks as another example, a vehicle produces sounds when travelling on a road. On the other hand, active sensors provide their own energy to detect the targets and that energy is reflected by the target in the ROI. RADAR (RAdio Detection And Ranging), LADAR (LAser Detection And Ranging) or LIDAR (Light-Imaging Detection And Ranging), Ultrasonic and SONAR (SOund Navigation And Ranging) [59][60] are examples of active sensors. For instance, a radar sensor radiates a series of pulses from antennas. When the pulses reach the desired target, some of the pulses' energy will be reflected back toward the radar antennas. The reflected energy will be measured and timed. The distance or range to the target is calculated from the time required for the pulses to travel to the target and come back again to the sensor. All these detection techniques are under the umbrella of tokenless detection approaches in which the target does not carry any additional device. In a token-based detection approaches, the target carries a token or tag which is a device such as laptop, Personal Digital Assistant (PDA), Radio Frequency Identification (RFID) tag or wireless device. The token assists in the detection and tracking. The target can also be classified by the unique token identifier [62].

In this thesis, passive sensor devices are used to detect the acoustic signals produced from the targets. The target to be tracked is assumed to be an isotropic sound source. The emitted acoustic density from the sound source (i.e., the target) is assumed to be known. The target acoustic power intensity received by the sensor nodes is modelled as decreasing with the distance from the target according to power n which is the attenuation decay factor and is typically between 2 to 5 according to the environment and atmospheric conditions [27]. Further details are presented in Chapters 3 and 4.

2.6.2 Target Classification

In this thesis, target class, importance or priority is involved. Target class can be obtained from target classification [63]. Target classification techniques aim to classify the target in terms of its type or importance. For instance, a target could be human, animal, moving vehicle or any objects of interest in the scene. Moreover, human objects are classified for example based on their historical behaviours or importance. Target classification can be regarded as pattern recognition task. Therefore, two main

approaches are used for classifications which are shape-based and motion-based classifications [15]. Shape information such as points, boxes, silhouettes and blobs are used to classify the target using shape-based classification. In motion-based classification, a strong clue to classify the object is salient features in its motion. Another form of target classification is token or tag tracking [62]. The token carried by the target is an electronic device such as RFID tag, PDA, laptop and wireless sensor device. All target information is stored and obtained from its tag [64].

However, in this thesis target importance is adaptively calculated based on the historical movement pattern of the target. A target is more important if it moves in random fashion with sharp bends. However, the proposed MTT algorithm also supports offline assignment of the target importance according to the target class or type. More details are provided in Chapters 3 and 4.

2.6.3 Node Selection

Due to the high number of the deployed sensor nodes in the ROI, typically several sensor nodes can detect the target. However, some of these sensor nodes provide useful information that improves the accuracy of the target state estimation. Moreover, some of sensor node's information might be useful but redundant while some might contain a lot of measurement errors. In order to prolong the network lifetime and save resources in terms of energy, bandwidth and processing, the target tracking task should only use the necessary sensor nodes that optimally reduce uncertainly of the target state. In other words, nodes selection techniques aim to activate the best necessary sensor nodes at each snapshot to perform the target tracking. Additionally, sensors selection requires communication between sensor nodes and this consumes energy. Therefore, a balance between the accuracy of the tracking and the cost to perform sensors selection is a main goal. In summary, sensors selection that belongs to the category of sensor network management field [65] is essential for tracking continuity, resource economy and tracking accuracy. In Chapter 3 and 4, proposed algorithms and techniques to select the best tasking sensor nodes to track the target at each tracking snapshot will be presented.

2.6.4 Target Localization

The objective of the localization is to estimate the current location of a moving target in the ROI by the sensor network. Mainly, the localization techniques are based on three kinds of physical measurement obtained or derived from sensor node readings. The following paragraphs present these measurement techniques in the context of acoustic-

based passive sensors. Firstly, in the Time Delay of Arrival (TDOA) [66][61] technique, the detection mechanism relays the estimation of arrival angle for audio signal using TDOA or steered beamforming. A TDOA method requires accurate time delay measurements. It is suitable for broadband signals. Secondly, Direction of Arrival (DOA) [67] requires a costly antenna array in each sensor node and is suitable for narrow band signals. It has lower quality compared to the other approaches [68]. TDOA and DOA are time delay-based localization approaches. Finally, in the Received Signal Strength Indication (RSSI) method, the sensor nodes in ROI use the received or measured acoustic energy emitted from the target (i.e., audio source) to locate it [56]. Therefore, RSSI is energy-based localization approach. This method is primarily based on the fact that the acoustic energy level decays with increasing distance between the audio source (i.e., target) and the listener (i.e., sensor node). Therefore, the target location can be determined using the acoustic energy readings from different known sensor node locations. In [56], it shows RSSI is a suitable choice for WSNs because it reduces the computational and communication costs. Furthermore, no accurate time synchronization is required between the sensor nodes. Therefore, energy-based localization approaches are more robust than time delay-based localization approaches which are sensitive to errors in time synchronization and echo effects [69]. Hence, target tracking algorithms proposed in this thesis adopt the RSSI technique. The RSSI mathematical model used in this research is presented in Chapters 3 and 4.

Triangulation is one method to obtain the target location using the known sensor node locations and the distance or range between these sensor nodes and the target [38]. In 3D localization, at least four sensor nodes readings are required, while three sensor nodes readings are required for 2D localization. In 2D localization shown in Figure 8, the set of equations to compute the unknown target location $\xi_T = (x_T, y_T)$ using n_g sensor nodes $\{\xi_S = (x_i, y_i),\ 1 \le i \le n_g\}$ is given as [38]:

$$\begin{bmatrix} (x_1 - x_T)^2 + (y_1 - y_T)^2 \\ \cdot \\ \cdot \\ (x_{n_g} - x_T)^2 + (y_{n_g} - y_T)^2 \end{bmatrix} = \begin{bmatrix} R_1^2 \\ \cdot \\ \cdot \\ R_{n_g}^2 \end{bmatrix} \tag{2.1}$$

where R_i is the measured distance or range between the sensor node i and the target. By subtracting the first row in (2.1) from the rest, a liner system of $n_g - 1$ equations is obtained as follows:

$$\mathbf{Au} = \mathbf{B} \tag{2.2}$$

where,

$$\mathbf{B} = 0.5\times\begin{bmatrix} R_2^2 - R_1^2 - x_2^2 + x_1^2 - y_2^2 + y_1^2 \\ \cdot \\ \cdot \\ R_{n_g}^2 - R_1^2 - x_{n_g}^2 + x_1^2 - y_{n_g}^2 + y_1^2 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} (x_1 - x_2) & (y_1 - y_2) \\ \cdot & \cdot \\ \cdot & \cdot \\ (x_1 - x_{n_g}) & (y_1 - y_{n_g}) \end{bmatrix} \quad \textbf{and} \quad \mathbf{u} = \begin{bmatrix} x_T \\ y_T \end{bmatrix}.$$

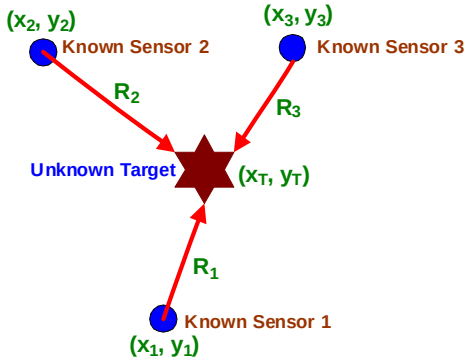


Figure 8 2D Localization

Equation (2.2) can be solved using least squares method [64][70] as follows:

$$\mathbf{u} = \begin{bmatrix} x_T \\ y_T \end{bmatrix} = (\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'\mathbf{B} \tag{2.3}$$

2.6.5 Target Tracking

The primary goal of target tracking is to estimate the trajectory, velocity and acceleration of a mobile target. Therefore, tracking is a series of localization problems. In camera-based target tracking [15][54][55], target tracking is achieved from one frame to another in the image sequence. Region-based, active contour-based, feature-based and model-based tracking are the primary tracking categories in camera-based target tracking. In region-based tracking, the motion region is obtained by subtracting the background from the image region that varies from frame to another. In active contour-based tracking, the contour of the target is tracked instead of involved the whole image region. Feature-based tracking extracts the target features using recognition and matches the features between images to track the target. For example, a target is bounded with rectangular box whose centroid is selected as the feature used for target

tracking. In model-based tracking, the target tracking is achieved by matching a known projected target model to the image data. In acoustic-based WSNs which will adopt in this thesis, target tracking aims to determine the target states such as location and velocity at every sampling interval which is the time between the two tracking snapshots. The localization method described in Section 2.6.4 can be used to track a target in acoustic-based WSNs.

2.6.5.1 Bayesian Networks

The target-tracking problem is considered a dynamic system. A dynamic system [71][72] is defined as the system where its states change over time. A state-space approach is used to model the dynamic system to estimate the system states using noisy measurements obtained from the system. The system states are encapsulated into a state vector that contains all required information to describe the system under investigation. For instance, kinematic characteristics including position, velocity and acceleration of the target are the information required to describe the tracking problems. The noisy measurements (i.e., observations) are related to the system states and encapsulated into the measurement vector. Generally, the dimension of the measurement vector is less than or equal the dimension of the state vector. Basically, two discrete-time models are required to describe the dynamic system in order to obtain the current and predicted system states. The first model is called the system model, state transition or evolution model in which the evolution of the system states over the time is described. The second model is called the measurement model, which is relating the noisy observations of the system states. For target tracking, the system model equation that describes the evolution of the target state $\left\{\mathbf{X}(k+1) \in \mathbf{R}^{n_x},\ k \in \mathbf{N}\right\}$ with respect to the time k is given by:

$$\mathbf{X}(k+1) = f[k+1, \mathbf{X}(k), \mathbf{w}(k)] \tag{2.4}$$

In Equation (2.4), f is the evolution function or system transition function and possibly nonlinear and time-varying function that relates the current state $\mathbf{X}(k+1)$ with the previous state $\mathbf{X}(k)$. Therefore, Equation (2.4) is considered a first order Markov process. $\mathbf{w}(k) \in \mathbf{R}^{n_w}$ is the process noise (or state noise) in the interval between k and $k+1$ with known distribution which is independent of time. n_x and n_w are the dimensions of the state and process noise (or state noise) vectors respectively, $\mathbf{N}$ is the natural numbers and $\mathbf{R}$ is the real numbers. The measurement model at time $k+1$ that

relates the target noisy observations or measurements $\mathbf{z}(k+1) \in \mathbf{R}^{n_z}$ with the target states $\mathbf{X}(k+1)$ is given by:

$$\mathbf{z}(k+1) = h[k+1, \mathbf{X}(k+1), \mathbf{v}(k+1)] \tag{2.5}$$

where, h is the measurement function and is possibly a nonlinear and time-varying function, $\mathbf{v}(k+1) \in \mathbf{R}^{n_v}$ is the measurement noise whose know distribution is independent of both process noise and time, n_z and n_v are dimensions of the measurements and measurement noise vectors, respectively. The measurements $\mathbf{z}(k+1)$ are conditionally independent. A survey of target measurement models is found in [73]. The dimensions n_x, n_v, n_z and n_w can be different. In fact, filtered estimates of $\mathbf{X}(k+1)$ are calculated recursively based on all available measurements or observations up to time $k+1$. As shown in Figure 9, the models described in Equation (2.4) and (2.5) are described as a Hidden Markov Model (HMM) or state space model (SSM) in which the unobserved states (i.e., hidden states) are filtered from the measurements. Therefore, it is a recursive filter that sequentially updates the previous estimates.

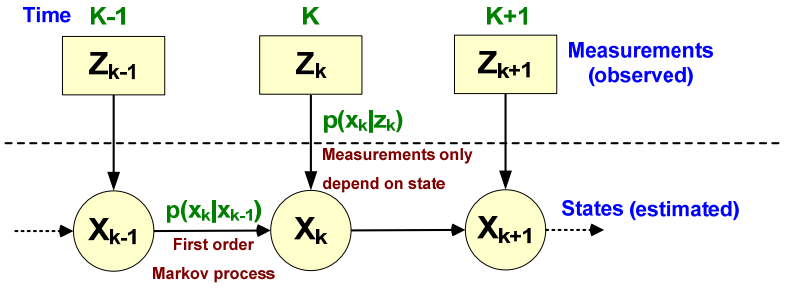


Figure 9 Bayesian Model

Target estimated states are calculated in a probabilistic distribution form, which called belief. Therefore, the state-space approach is suited for recursive Bayesian filtering. This means that degree of belief in the state $\mathbf{X}(k+1)$ at different time $k+1$ is calculated given all measurements $Z_{k+1} = \{\mathbf{z}(i), i \le k+1\}$ up to time $k+1$. Therefore, it is required to calculate recursively in time the posterior Probability Density Function (PDF) $p[\mathbf{X}(k+1) \mid Z_{k+1}]$ by assuming that initial PDF (i.e., prior) $p[\mathbf{X}(0) \mid Z_0] = p[\mathbf{X}(0)] = p(0 \mid 0)$ (where Z_0 is the set of no measurement) and the state vector $\mathbf{X}(k+1)$ are known. Thus, $p[\mathbf{X}(k+1) \mid Z_{k+1}]$ at time $k+1$ is obtained recursively using two stages known as prediction and update. Generally, estimating the state $p[\mathbf{X}(k+1) \mid Z_l]$ is called prediction if $l \prec k+1$, update if $l = k+1$ and smoothing if $l \succ k+1$.

In the prediction stage, the prior PDF $p[\mathbf{X}(k+1)\,|\,Z_k]$ of the system state at time $k+1$ is calculated (i.e., predicted) without knowing the measurement $\mathbf{z}(k+1)$ and using the system model described in Equation (2.4) via Chapman-Kolmogorov equation which is give by:

$$p[\mathbf{X}(k+1)\,|\,Z_k] = \int p[\mathbf{X}(k+1)\,|\,\mathbf{X}(k)]p[\mathbf{X}(k)\,|\,Z_k]d\mathbf{X}(k) \tag{2.6}$$

where, $p[\mathbf{X}(k)\,|\,Z_k]$ at time k is available and $p[\mathbf{X}(k+1)\,|\,\mathbf{X}(k)]$ (i.e., transition distribution of the first order Markov process) is the probabilistic model of state evolution defined in Equation (2.4) given the known statistics of $\mathbf{w}(k)$. $p[\mathbf{X}(k+1)\,|\,\mathbf{X}(k)]$ is referred to as the prior distribution. The update stage is achieved at time $k+1$ where a measurement $\mathbf{z}(k+1)$ is available. Using Bayes' rule, the prior PDF $p[\mathbf{X}(k+1)\,|\,Z_k]$ calculated using Equation (2.6) is updated to get the posterior PDF $p[\mathbf{X}(k+1)\,|\,Z_{k+1}]$ of the current state via the following equation:

$$p[\mathbf{X}(k+1)\,|\,Z_{k+1}] = \frac{p[\mathbf{z}(k+1)\,|\,\mathbf{X}(k+1)]p[\mathbf{X}(k+1)\,|\,Z_k]}{p[\mathbf{z}(k+1)\,|\,\mathbf{z}(k)]} \tag{2.7}$$

where the likelihood function, $p[\mathbf{z}(k+1)\,|\,\mathbf{X}(k+1)]$, (i.e., marginal distribution of Markov process) is defined in Equation (2.5) given the known statistics of $\mathbf{v}(k+1)$ and $p[\mathbf{z}(k+1)\,|\,\mathbf{z}(k)]$ (i.e., the normalized constant) is defined as:

$$p[\mathbf{z}(k+1)\,|\,\mathbf{z}(k)] = \int p[\mathbf{z}(k+1)\,|\,\mathbf{X}(k+1)p[\mathbf{X}(k+1)\,|\,Z_k]dx_k \tag{2.8}$$

Therefore, the HHM or SSM are described by:

$$p[\mathbf{X}(k+1)\,|\,\mathbf{X}(k)] \text{ for } k \geq 0 \tag{2.9}$$

$$p[\mathbf{z}(k+1)\,|\,\mathbf{X}(k+1)] \text{ for } k \geq 0 \tag{2.10}$$

The recurrence Equations (2.6) and (2.7) are the optimal and exact Bayesian solution to compute the posterior PDF. However, this solution is a conceptual solution and cannot be calculated analytically because the integrals in Equations (2.6) and (2.7) are not tractable. Kalman Filter (KF) [72][74][75] and Grid-based Filter (GF) [76] approaches are optimal and analytically possible and exact under the following assumptions: (1) $f[k+1,\mathbf{X}(k),\mathbf{w}(k)]$ and $h[k+1,\mathbf{X}(k+1),\mathbf{v}(k+1)]$ are known and linear functions, and (2) $\mathbf{w}(k)$ and $\mathbf{v}(k+1)$ are drawn from known Gaussian distribution. Therefore, the PDF of distribution $p[\mathbf{X}(k+1)\,|\,Z_l]$ for $l \prec k+1$, $l = k+1$ and $l \succ k+1$ are Gaussian at all times. On

the other hand, if these assumptions do not hold (i.e., one or both of the models in Equation (2.4) and (2.5) are nonlinear or/and one or both of the system and measurement noises are non-Gaussian), Extended Kalman Filter (EKF) [72][74][75], Approximate Grid-based Filter (AGF) [76] and Particle Filter (PF) [77][78][79] approximate the optimal Bayesian solution to get suboptimal solutions. However, EKF will be used in this thesis. In the Section 2.6.5.2, EKF will be introduced and in Chapters 3 and 4 further mathematical details about EKF are provided.

2.6.5.2 Extended Kalman Filter (EKF)

EKF [72][74][75] is based on the linearization of the nonlinearities in the dynamic and/or the measurement models. It deals with conditional mean and covariance. In Equations (2.4) and (2.5) $\mathbf{w}(k)$ and $\mathbf{v}(k+1)$ are assumed to be adaptive, zero-mean and white with $Q(k)$ and $R(k+1)$ covariance matrices respectively such that:

$$E[\mathbf{w}(k)] = 0 \ \& \ E[\mathbf{w}(k)\mathbf{w}(k)'] = \mathbf{Q}(k) \tag{2.11}$$

$$E[\mathbf{v}(k+1)] = 0 \ \& \ E[\mathbf{v}(k+1)\mathbf{v}(k+1)'] = \mathbf{R}(k+1) \tag{2.12}$$

The main purpose of EKF is to calculate the predicted and update states and their covariance matrices in both prediction and update stages. In the prediction stage at time k, the predicted state, $\hat{\mathbf{X}}(k+1|k)$ to time $k+1$ is driven from the dynamic model and the measurements (Z_k) up to time k, such that $\hat{\mathbf{X}}(k+1|k) \approx E[\mathbf{X}(k+1)|Z_k]$. The state prediction error of $\hat{\mathbf{X}}(k+1|k)$ is defined as $\tilde{\mathbf{X}}(k+1|k) = \mathbf{X}(k+1) - \hat{\mathbf{X}}(k+1|k)$. Thus, the state prediction covariance matrix is $\mathbf{P}(k+1|k) = E\{[\tilde{\mathbf{X}}(k+1|k)][\tilde{\mathbf{X}}(k+1|k)]'|Z_k\}$. Similarly, in the update stage at time $k+1$, the updated state, $\hat{\mathbf{X}}(k+1|k+1)$ of time $k+1$ is defined as $\hat{\mathbf{X}}(k+1|k+1) \approx E[\mathbf{X}(k+1)|Z_{k+1}]$. The state updated error of $\hat{\mathbf{X}}(k+1|k+1)$ is defined as $\tilde{\mathbf{X}}(k+1|k+1) = \mathbf{X}(k+1) - \hat{\mathbf{X}}(k+1|k+1)$. Thus, the state updated covariance matrix is $\mathbf{P}(k+1|k+1) = E\{[\tilde{\mathbf{X}}(k+1|k+1)][\tilde{\mathbf{X}}(k+1|k+1)]'|Z_{k+1}\}$. The mathematical equations to calculate the sates and their covariance matrices are presented in Chapters 3 and 4 for single and multi target tracking in WSNs.

2.6.6 Behaviour Analysis

After target tracking, understanding target behaviour is the next stage [15][54]. Behaviour understanding is to classify the target feature data by matching it to a group of labelled reference behaviours. This concept allows the detection of suspicious human

behaviour via automated visual surveillance. Video understanding can be performed using two main steps, which are lower-level processing and higher-level artificial intelligence. Target detection and classification are lower level computer vision functions. Behaviour recognition gained from tracking is higher level processing. Many methods are used for behaviour understanding and analysis such as HMM, Dynamic Time Warping (DTW), Finite-State Machine (FSM), Nondeterministic-Finite-State Automaton (NFA), Time-Delay Neural Network (TDNN), Syntactic/Grammatical Techniques, Self-organized Neural Network, Agent-Based Techniques and Artificial Immune Systems. However, behaviour analysis is beyond the scope of this thesis.

2.6.7 Person Identification

Personal identification [15][54] is a special behaviour understanding. The main biometric features used in personal identification are face and gait recognition. Face detection, face tracking, face feature detection and face recognitions are the main steps in face recognition. Model-based, statistical, physical-parameter-based, spatio-temporal motion-based and fusion of gait with other biometric are main methods for gait recognition. However, personal identification is beyond the scope of this thesis.

2.6.8 WSNs Target Tracking Architecture

Target tracking in WSNs can be classified into centralized or distributed approaches [68]. In a centralized target tracking system, the sensor nodes detect the target and send its signature to the BS or fusion centre. The fusion centre performs the detection and tracking algorithms based in the information fetched from the sensor nodes. However, since hundreds or thousands of sensor nodes are spread over ROI, a lot of information about the target signature sensed from the sensor nodes will be sent to the fusion centre at the same time. Therefore, the centralized approach causes the lifetime of WSNs to be limited due to the huge communication overhead between the sensor nodes and the fusion centre. On the other hand, distributed approach systems [80] are designed so that the WNS is divided into regions, cells or clusters via clustering algorithms. One leader, cluster head, cell head or manager node for each region is selected. Other sensor nodes in the cluster are members. The processing is performed cooperatively between the leader and sensor nodes.

There are static clustering and dynamic clustering architectures in WSNs [68]. In the static clustering techniques, the clusters are formed at the time of network

deployment. Since the clusters attributes, leader and members are static all the time. Static clustering improves the energy consumption of the WSNs. However, static clustering does not offer fault tolerance. If the cluster head dies due to energy depletion for example, the network may not have enough sensor nodes to carry the tracking. Additionally, sensor nodes in different clusters cannot share their information and collaborate on data processing.

In dynamic clustering architectures, the formation of the cluster is triggered by certain events such as detection of a target or prediction of the target next location. An election algorithm is performed to elect the leader of the cluster. The group leader has the responsibility of management the current group and formation of the next group [57][69][81][82]. However, dynamic clustering consumes the energy in forming and disbanding the clusters for seamless tracking.

In this thesis, a dynamic distributed architecture is adopted. A group of tasking sensor nodes is selected to cooperatively track the target in WSN. One of the group sensor nodes is elected to be the leader. More details are presented in Chapters 3 and 4.

2.7 Biologically Inspired Research & Self-Organized Networks

Many possible definitions have been proposed for self-management or self-organized networks. A self-management system [83] can be described according to eight characteristics, which are self-configuration, self-healing, self-optimization, self-protection, self-awareness, environment-awareness, openness and transparency. Generally, a famous idiom that indicates these elements is the self-* property. Self-configuration means that the communication system has to dynamically configure and reconfigure itself based on the dynamic changes of the network or environment. The self-healing property allows the communication system to detect the failure and thus contain it, replace it with another feature or eliminate it without affecting the system. Additionally, the system has to behave proactively in term of prediction of the problems. Maximizing the resource management and utilization, and achieving efficient load balancing are the main two functions of self-optimization systems. Self-protection element has the responsibilities of protection of the systems from the attacks and performing all the system security aspects. Self-awareness or self-knowledge allows the system to know itself in terms of available resources, applied load etc. Additionally, the communication system must be aware of the execution environment to cope with any

environmental changes. Openness means that communication system has to operate in different environments and conditions. Finally, communication system should operate in a transparent fashion with respect to the users. Its complexity has to be hidden.

Characteristics and behaviours inspired by biological and ecological systems have become enthusiastic research methodologies in communication and information technologies. For example, advanced self-management phenomena can be found in the nature itself [84]. Ants in their colony cooperate to find the shortest path to the food source; schools of fish swim in patterns to improve the response time to attacks; human autonomic nervous system works cooperatively to identify the things.

As shown in Chapters 3, 4 and 5, the target tracking, and task mapping and scheduling schemes proposed in this thesis are under the umbrella of the biologically inspired, self-organized communication networks. Further details are explored in these chapters.

2.8 Task Mapping and Scheduling in WSNs

Task mapping is defined as the mechanism to assign available resources to tasks or jobs. Recourses include the processing capability, available storage, remaining energy and network bandwidth. Task scheduling is the execution sequence of the tasks or jobs so that the performance objectives are optimized. The performance objectives are defined according to the underlying system requirements and they can include execution time and/or energy consumption.

Applications can be divided either into independent or dependent tasks. It is well known that optimal task mapping is an NP-complete problem [7], where NP stands for Nondeterministic Polynomial-time. The polynomial time refers to the algorithm's running or execution time. An algorithm is a polynomial time if its running time is no larger than a polynomial function of its input size. No fast solution of an NP-compete problem is known. As the size of the NP-compete problem increases, the time required to solve it using a given algorithm increases. Therefore, heuristic techniques or approximately algorithms are needed to obtain near-optimal solutions.

In this thesis, two algorithms are developed for task mapping and scheduling in WSNs. The first algorithm assumes the application can be divided into independent

tasks and the second assumes that there are dependencies between the application tasks. Chapter 5 explores these algorithms in depth.

2.9 State-of-the Art Review

In this section, a literature review of relevant biological research, target tracking in WSNs and task mapping and scheduling in WSNs will be explored and analysed. The advantages and limitations of the current approaches to target tracking, and task mapping and scheduling in WSNs are considered. The proposed target tracking, task mapping and scheduling schemes in WSNs are presented in detail in Chapters 3, 4 and 5 to address the limitations of the current work.

2.9.1 Biological Inspired Techniques and Algorithms

In this thesis, the proposed target tracking, task mapping and scheduling in WSNs schemes are inspired from biological principles. However, a number of researchers have already considered applying biological and ecological principles within communication networks for different purposes. Therefore, this section will only mention the biological aspects of their research insofar as they relate to target tracking, task mapping and scheduling in WSNs. However, the reader can refer to the given references for further background.

In [85][86][87], a biological-inspired architecture has been introduced to allow network services to adopt and scale with dynamic network conditions. In these papers, the network service such as HTTP is designed as biological abstract entities, which can perform biological behaviours such as migration, replication, reproduction and death. Additionally, the authors in [85][86][87] assume that the service can autonomously die, migrate and replicate depending on the locally available information such as their neighbours' resource availability. In [88], the authors have developed novel models inspired from molecular biology to achieve autonomic capabilities in communication networks.

In [89], a biologically inspired system has been implemented to provide autonomous adaptation to the environmentally dynamic changes such as network traffic, user location and resource availability. The authors in this paper have developed a middleware platform that has biological features as well as the application services. In [89], the platform can migrate, replicate, reproduce, exchange energy and die accordingly to the network conditions.

In [90], the authors map the network service that is executed using platforms to ants and bees. Ants and bees work continually in their colonies from birth to death. They move to other location to find food. In times of crisis, they stop looking for food and protect their colonies from intruders. In [90], network service behaves like ants and bees. When a network service is created, it registers with the system. After that, it begins to perform activities. It stops its work if another high or priority network service wants to work. It also can migrate among platforms.

In [91], a combination of Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) has been developed in one algorithm to be used in distributed multi-agent systems to search for multiple targets. In [91], the system is inspired from biological swarm intelligence so that the agents interact locally. The agent can detect the target within its local sensing range. In [91], agent can behave like ants. The ants work cooperatively to determine the shortest path to the source of the food. Each ant lays down chemical trails of pheromones. The ants follow the high intensity pheromones. This biological behaviour allows the ants to adapt with the environment changes. The same principle of ants is applied to the agents in [91]. In [91], the agent builds virtual pheromone data structure whenever it detects a target. It broadcasts this target information to its neighbours. The agent may move toward the target if it receives information about the target and the pheromone intensity is strong enough.

2.9.2 Target Tracking in WSNs

In fact, many researchers focus on target tracking in WSNs. As described in Section 2.6, several operational steps are involved in tracking system. In this section, the literature review about tracking framework, sensor nodes selection, sensor nodes election and target tracking techniques are proposed. In Chapters 3 and 4, the proposed tracking schemes for STT and MTT are presented.

2.9.2.1 Target Tracking Framework

In [16][69][5][92] dynamic clustering architectures are proposed. In [16], there is one leader at a time in the vicinity of the target. Based on information-driven approach and the cost, the current leader selects the new leader. The current target state is sent to the new leader. As shown in Figure 10, during the target movement from the left to the right, the leader node hand offs from sensor node to sensor node [16]. In [16], initially, the user query arrives in the network from sensor node “Q”. The query is directed to the

region of interest. Sensor node “a” is the initial leader. It computes the initial estimate of the vehicle state, determines the next best leader (b) and hands off the state information to “b”. Sensor node “b” uses its measurement to update the state estimation using Bayesian filter, for example. The process is repeated through the next leaders “c”, “d”, “e” and “f”. Sensor nodes “d” and “f” send back the state estimation to the queering sensor node “Q”. However, the framework in [16] does not consider the energy-efficiency of sensor nodes that can be achieved by scheduling the sensor nodes to be in active or sleeping modes as necessary. Further to this, all sensor nodes should be in active mode to be able to receive the queries sent by the user. Additionally, a single target is assumed to be tracked.

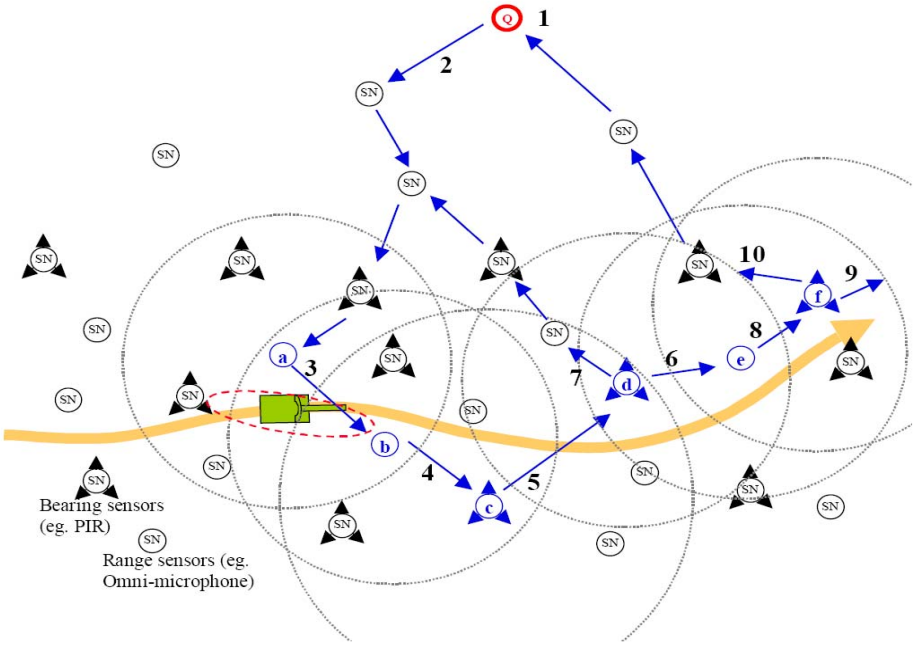


Figure 10 Network Framework [16]

In [69], dynamic clustering for target tracking in WSN is proposed. A sensor node can volunteer to be a cluster head (CH) if it receives signal from the target exceeds a predefined threshold. Other sensor nodes around the target are invited to become cluster members. Cluster members send the target signature to the cluster head and then cluster head performs the processing. Two or more sensor nodes can receive a good sound signal above the threshold and volunteer as cluster head. As shown [69] in Figure 11, a probabilistic leader volunteering is achieved using Voronoi diagram. If the target inside the inner dotted circle, the target will be inside the Voronoi cell of CHi . After the CH localizes the target, it sends the target state to the sink node. The simulation results in

[69] show that the dynamic clustering proposed in [69] reduces the target localization error compared to the static clustering approaches. However, like the scheme in [16], the scheme in [69] does not consider the sensor node scheduling between the sleeping and active modes. Therefore, the framework in [69] is not energy-efficient.

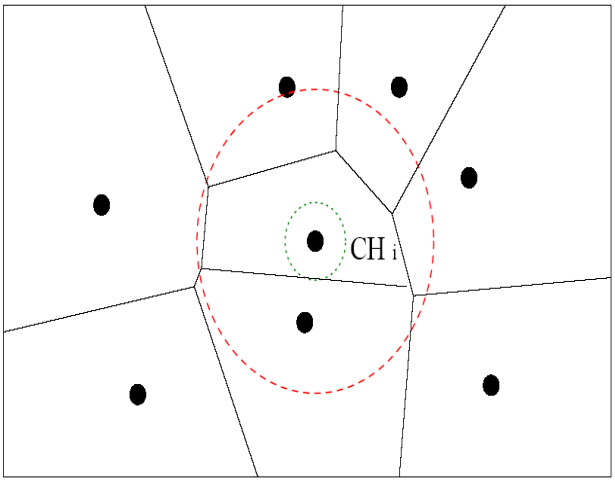


Figure 11 Leader Volunteering using Voronoi Diagram[69]

In [92] as shown in Figure 12, the elected leader node forces all its neighbours to sense the target by broadcasting a message to them. However, hundreds or thousands of sensor nodes are deployed in the area of interest. Therefore, communication between sensor nodes neighbours to inform them to sense the target is expensive in term of energy consumption.

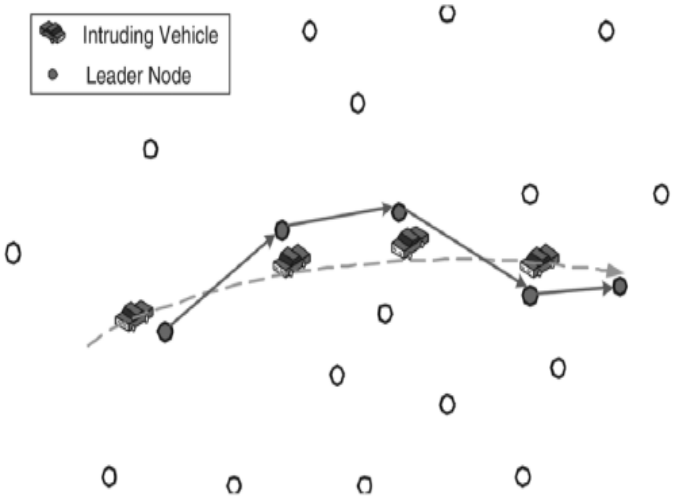


Figure 12 The illustration of Presented Scenario in [92]

In [5], dynamic clustering for WSNs target tracking is proposed. The framework assumes that the sensor nodes are in the sleeping mode and triggers by using ultralow power channel to perform the sensing tasks. In [5], when the target entering the ROI, it

is initially detected using some low-power sensor nodes such as passive infrared (PIR) sensor nodes. As the authors in [5] explain, in Figure 13 the sensor nodes in the current cluster send the measurement information to the CH. The CH calculates the target states using the information received from the sensor nodes. The CH chooses the next cluster nodes and the next CH. Although, MTT is not considered in [5], we believe that the framework in [5] is energy-efficient.

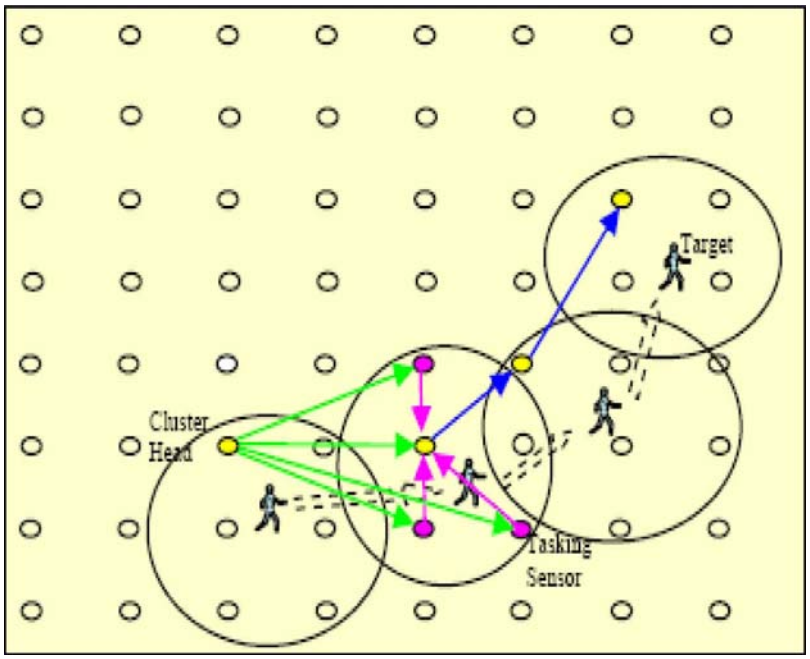


Figure 13 WSN Tracking Architecture [5]

In [93] and [94], a distributed tracking algorithm is presented in a static clustering architecture. As shown in Figure 14, each cluster head (CH) knows all information of all sensor nodes inside its cluster all the time. Three sensor nodes are assumed to be enough to determine the target state (i.e., location and velocity) using triangulation. The target current location and predicted next location is calculated. The current CH migrates the target states to the next predicted CH. Using the information of the sensor nodes inside the CH database, the next CH selects from its cluster three sensor nodes that have the smallest distances from the predicted target location under condition that the target will be in their sensing range. The CH asks its neighbours for help if it cannot find three sensor nodes in its cluster. However, we believe that the scheme presented in [93] and [94] is not energy-efficient because the CH is in communication with all sensor nodes inside its cluster to ascertain their information at all times. Additionally, the target state probability distribution is not considered in the tracking scheme presented in [93] and [94].

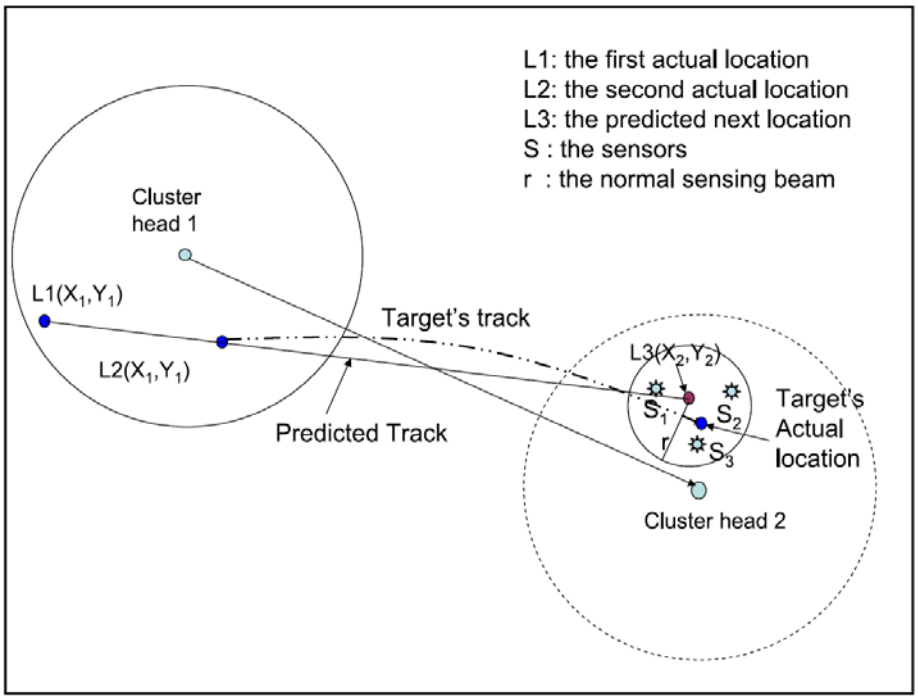


Figure 14 WSN Tracking Network Architecture [93]

The static architecture in [68] is shown in Figure 15. The target may enter the area across one of the four corners of the sensing area. The specialized photo sensor nodes in the corner are active all the time and they can sense any target entering the sensing area. The acoustic sensor nodes are only in active mode if there is a target. The sensing area is divided into equal regions and one cluster head (i.e., processing node) is located in the centre of each region. The cluster heads collect the sensing information from the specialized photo and acoustic sensor nodes, process the data and send the target states to the base station. The results in [68] show that the static clustering approach reduces the energy consumption compared with the dynamic architecture schemes. However, there are no techniques in [68] to explain the formation of this static clustering and the energy consumption required to form it. Moreover, it is difficult to build static architecture in areas that are difficult to be physically accessed such that forests, mountains and under water. The scheme in [68] does not include a method by which the sensor nodes can be informed to be in active mode at the time of target arrival.

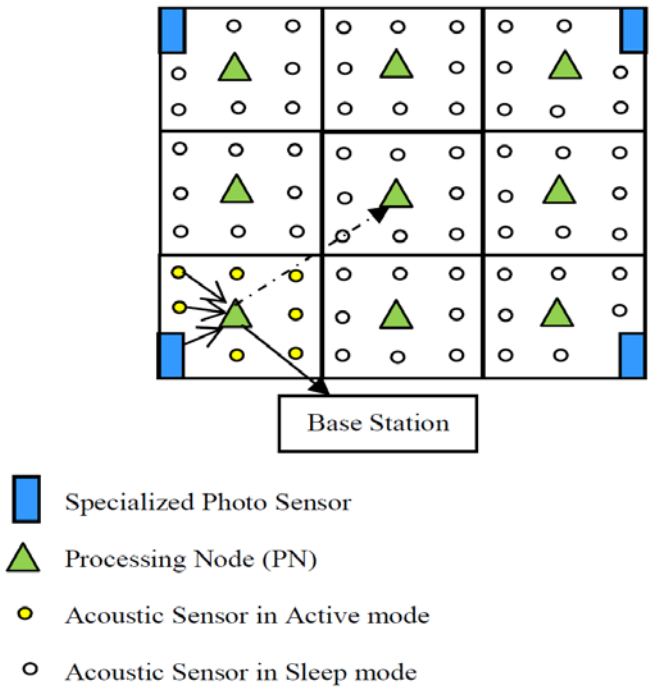


Figure 15 Tracking Network Structure [68]

In [81], a static WSN architecture for target applications is presented. As shown in Figure 16, sensor nodes are arranged in triangular form where the sensing range is equal to the side length of the triangles. All sensor nodes are in sensing mode all the time. The sensor nodes detect the target either using passive sensing such as the received signal strength or active sensing by emitting a signal to the target and calculating the time of the reflected signals from the target. Each sensor node has to know at least the information about the sensor nodes that are two hops away from it. In [81], three sensor nodes are assumed to be sufficient to calculate the target location using trilateration algorithm. One of these three sensor nodes will be the master and the other two nodes are slaves. The master node runs the tracking algorithms and the slave nodes send the target range to the master. The sensor nodes tracking the target are changed when the target moves through the triangles. For example, when the target enters the area A1 instead of the slave S2, the slave S6 starts to track the target with the master S0 and other slave S1. However, setting the sensor nodes to be in sensing mode all the time increases the energy consumption of the sensor nodes and in turn reduces the network lifetime. Additionally, energy is consumed during the information gained by each sensor node about the sensor nodes that are two hops away from it. Moreover, in [81] there is no technique to show how the sensor nodes can be arranged in triangular form.

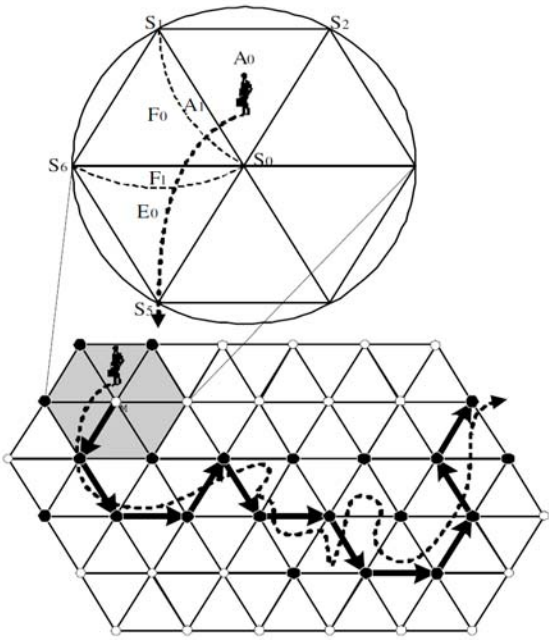


Figure 16 WSN Tracking Framework [81]

2.9.2.2 Sensor Nodes Selection Algorithms

In [93] and [94] the nearest three sensor nodes to the target predicted location are proactively selected to track the target at next tracking snapshot. However, these sensor nodes have to be inside the cluster where the target is located. Therefore, the sensor nodes inside other clusters will not be selected even if they are closer to the predicted location of the target than the sensor nodes inside the cluster in which the target is currently located. In [81], the nearest three sensor nodes to the target are selected to track it. These approaches to select the closest sensor nodes to the target are computationally efficient. However, they do not take into account the uncertainty in the target state calculation and this degrades the accuracy in the prediction and tracking [82].

In [92], the number of sensor nodes to track the target is calculated so that the tracking accuracy is improved by a predefined improvement value. In [92], detection fusion coefficients of all sensor nodes that detect the target are computed based on their received target signal power. After that, the sensor nodes that have the highest detection fusion coefficients are selected. Nevertheless, this approach is practically difficult and costly in term of energy and processing because all sensor nodes within the range of the target will perform measurements from which the useful ones are then selected.

In [95], the sensors selection is based on the proximity of the sensor node to the target and the co-linearity between the sensor nodes. The co-linearity between a set of sensor

nodes increases when the lines connected the sensor nodes are more likely a straight line. [95] shows that the tracking error is reduced with decreasing co-linearity between the sensor nodes and decreasing the proximity of the sensor node to the target. In [95], three sensor nodes are selected to track the target at each tracking event. The selection algorithm consists of two steps. Firstly, two sensor nodes are selected from the set of the sensor nodes that their distances to the predicted target location is less than or equal the sensing range. The selection of the two sensor nodes is based on the angle that the target is located between them, and the proximity to the predicted location of the target. Secondly, the third sensor node is selected so that it has the minimum co-linearity with the selected two sensor nodes in the first step. However, [95] does not consider the uncertainty in the target state calculation and this degrades the accuracy in the prediction and tracking [82]. Furthermore, as declared in [95], the computational complexity in first step of sensors selection is $O(N^2)$ where N is the number of sensor nodes that can detect the target. Therefore for $N = 20$, it is required to evaluate $N!/(N-2)!2! = 190$ combinations to get the first two sensor nodes and another $N = 20$ operations to get the third one. Obviously, this is expensive in terms of computing and energy consumption especially for large value of N and considering MTT.

The algorithms for sensor selection based on the most informative sensor nodes is set by how much the amount of information the sensor node can bring (i.e., how much this sensor node data is useful) or how much uncertainly (i.e., opposite to accuracy) of the target state the sensor node can reduce. In [16] and [82] the next sensor node to track the target is selected based on maximizing the information utility measure. The information utility measure is calculated in [16] and [82] for each sensor node based on the Mahalanobis distance between the sensor node and the predicted location of the target taking into the account the covariance of the target state. The results in [16] and [82] show that sensor selection based on Mahalanobis distance effectively reduces the uncertainty in the target state compared with the sensor selection based on the nearest neighbour criteria by which the next sensor nodes is the nearest neighbour to the current sensor node. However, the tracking algorithm used in [16] and [82] selects only one sensor node at each time step. Moreover, it does not show how to select the sensor nodes whose measurements are potentially useful.

In [96] the next sensor node is selected so that the expected conditional entropy of the posterior target location is minimized. The statistical entropy measures how much

the randomness of a given random variable. Nevertheless, this approach is practically difficult and costly in term of energy because all sensor nodes within range of the target will perform measurements from which the useful ones are then selected. Furthermore, only one sensor node at each time step is selected in [96].

In [97], the sensor nodes selection is based on the mutual information between the target state and the measurement to determine how much this measurement is useful. The mutual information is the amount of the information obtained from one random variable by observing another. However, like [96], this method requires the measurements to be available before the sensor nodes selection.

In [98], sensors selection technique based on information utility measurement for bearing-only sensor nodes is presented. As shown in Figure 17, $S_i = (x_i, y_i)$ is the sensor node location. The covariance ellipsoid is used to represent the covariance matrix. The covariance ellipsoid of the target prior position PDF (i.e.,, $p(\mathbf{X}_{k+1} \mid Z_{1:k})$) is shown in Figure 17. Its long and short axes are $3\sigma_x$ and $3\sigma_y$ respectively. In the bearing sensor nodes [98], the sensor nodes along the shorter axis of the ellipsoid reduce the target position uncertainty of the prior PDF. The shadowed area is the modified uncertainty of target position and it results from the intersection between the sensor node bearing errors and the original covariance ellipsoid. This area is approximated to Gaussian distribution. The covariance matrix of this new a Gaussian distribution is computed. The determinant of the covariance matrix is proportional to the rectangular region enclosing the covariance ellipsoid representing the covariance matrix. Therefore, in [98], the sensor node is selected so that it minimizes the determinant of the covariance matrix of the new Gaussian distribution. However, the selection algorithm used in [98] selects only one sensor node for each tracking snapshot. Additionally, the energy cost to select the next sensor node is not considered. The rectangular region of the covariance ellipsoid will be zero in case of shrinking the smallest principal axis to zero while the uncertainty longest principal axis might remain large [82].

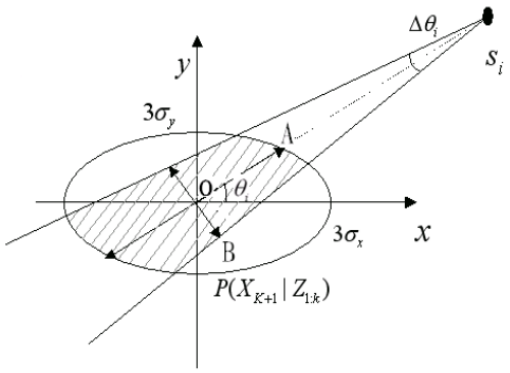


Figure 17 The uncertainty of the Target Position [98]

In [5], the probability for the sensor node to detect the target within its detection region is introduced. The PDF of the predicted target location is computed. Then, the predicted detection probability (PDP) of the sensor node is computed as the integration of the probability that the sensor node detects the target and the probability of the target to be in the predicted location. At each time step, the sensor nodes with the highest PDP are selected to track the target. However, this technique is concerned with the detection probability rather than the usefulness of the measurements obtained from the sensor nodes.

All the above approaches only consider single target tracking. Most of the research into MTT [99][100-102] focuses on the data association problem that is the techniques to know which measurements were generated by which targets. Sensor nodes selection management in MTT receives less attention [103]. In [103] and [104], a near-optimal sensor node subset is found to track multiple targets. Sensor nodes in [103] and [104] are assumed to be able to detect more than one target at a time. The objective function to select the near-optimal sensor node subset is to maximize the overall tracking accuracies of the targets. Off-the-shelf convex optimization and local search techniques are employed to obtain a near-optimal solution. However in [103], the targets move in a predictable fashion and the optimization problem uses all the sensor nodes to find the near-optimal solution. This is computational expensive. Moreover, the system does not provide a mechanism to predict the sensor nodes that may detect the target at each time step; furthermore there is no mechanism to set the target priorities and the maximum allowable iterations for the local search algorithm.

The above research in MTT does not consider the concept of conflict nodes whereby a sensor node that is not capable of detecting and serving more than one target at the same time has to decide to which target it will serve.

2.9.2.3 Sensor Nodes Election Algorithms

In [69] and [81], the cluster leader is elected so that it is the nearest sensor node to the target. In [92], the leader node is the sensor node that has the minimum statistical expectation of the difference between the target and the sensor node location. However, the communication is always between the sensor nodes in the cluster and the target. Therefore, the election in [69][81][92] does not reduce the communication energy consumption between the cluster nodes.

In [82], the leader node is chosen so that it is the closest node to the centroid of the cluster. In [5], the cluster leader is elected so that the total energy consumption used for all cluster members to transmit their measurements to it is minimized. The election in [82] and [5] will improve the communication energy efficiency of the cluster nodes. However, all these election techniques do not consider load balancing between the cluster nodes in terms of resources availability. Therefore, if the elected leader node has a low battery level, it will die quickly leaving gaps in the WSN and in turn reduces the network lifetime.

2.9.2.4 Target Tracking Techniques

Most of the existing research into STT in WSNs adopts a uniform or fixed sampling interval, which is the time between two successive tracking events [93][94][82], [96][16][92][95][97][105 - 108]. In case of MTT in WSNs, most of the research focuses on the data association problem and uses fixed sampling interval [99][100 - 104]. However, the target can be lost if its motion includes abrupt changes and the sampling interval is chosen to be a large value. On the other hand, energy consumption is increased if the sampling interval is chosen to be a small value. The rest of this section explores the STT schemes in WSNs that use an adaptive sampling interval.

In [109], a simple prediction model is used to locate a moving object. The sampling interval is changed based on the average historical target speed. However, changes in the target direction are not considered in the sampling interval calculation. Additionally, the sampling interval for each averaged speed of the target is calculated offline.

In [5] and [110], adaptive sampling interval target tracking schemes are proposed but they do not consider randomness in the motion of the target, i.e., undergoing “sharp bends” in its path. As shown in Chapter 7, the proposed tracking scheme is compared against the uniform schemes and the tracking schemes presented in [5] and [110].

In the tracking scheme proposed by Xiao in [110], the sampling interval is computed so that the updated tracking accuracy is satisfied. A single sensor node is selected to track the target at each tracking event. The sampling interval is chosen in [110] to be between minimum and maximum values. Two operational modes are proposed in [110] which are Fast Tracking Mode (FTM) and Track Maintenance Mode (TMM). FTM mode is used when the current tracking error, that is defined as the “trace" of the updated state covariance matrix, is not satisfied or non of the sensor nodes that may detect the target at the next tracking snapshot can achieve a satisfactory updated tracking error using any allowable sampling interval. In this case, the sampling interval is set to its minimum value and the sensor node is selected so that it minimizes the next updated tracking error and energy consumption used for communication between it and the current sensor node. TMM mode is used when the current tracking error is satisfied and at least one sensor node can achieve an update tracking error within acceptable limits using a certain sampling interval. In this case, the authors in [110] developed a discrete search algorithm to select the sensor node and the sampling interval so that the energy consumption used for communication between the next potential sensor node and the current sensor node is minimized using a biggest value of the sampling interval. The discrete algorithm divides the allowable values of the sampling interval into a discrete set of numbers. Then, for each value of the sampling interval starting from the biggest value, the energy consumption used for communication between the next potential sensor node and the current sensor node is calculated. If some sensor nodes satisfy the next updated tracking error for a given sampling interval, the sensor node that has the minimum updated tracking error is selected and the discrete search algorithm is terminated. However, in [110], one tasking sensor node at each time step is selected. Furthermore, choosing the tracking accuracy threshold after which the updated tracking error is not satisfied dramatically affects the total energy consumption and it is not easy to set it as it depends on the motion pattern of the target. Additionally, Xiao’s scheme assumes the sensor nodes that can detect the

target at each time step are known without including a method or technique to identify them.

In Lin's approach [5], the sampling interval is calculated based on the predicted tracking accuracy. This scheme can guarantee the predicted position accuracy to be less than or equal the predefined threshold. Like [110], FTM and TMM operational modes are adopted. FTM is used when current updated position uncertainty, which is defined as the "trace" of the updated position covariance matrix, cannot be satisfied. In this case, the sampling interval is set to its minimum value. On the other hand, TMM is operated when both the current updated and predicted position uncertainties are satisfactory. In this case, the sampling interval is calculated so that the predicted position uncertainty is equal a predefined threshold value. The sensors selection strategy in [5] is illustrated in the Section 2.9.2.2. However, it is difficult to decide in [5] the value of the threshold because the target can sometimes make unexpectedly abrupt changes in motion. Therefore, choosing a large threshold value may cause loss of the target. On the other hand, choosing a small threshold value wastes energy if the target travels in a uniform manner.

2.9.3 Task Mapping and Scheduling in WSNs

Task mapping and scheduling are considered in depth in traditional parallel computing environments including high performance computing, heterogeneous computing, grid computing and distributed computing systems [8][111 - 120]. However, the design objectives of these traditional parallel processing systems are different from those of WSNs. For example, in [112] and [114] the goal is to minimize the execution time of the applications. However, the execution time of the application in WSNs has to meet the application deadline (i.e., time constraint) after which the execution of the application will not be useful anymore. Battery energy and wireless communication constraints are not considered in traditional parallel processing systems. Thus, task mapping and scheduling in traditional parallel processing systems cannot directly apply to WSNs. A number of researchers have already considered task mapping and scheduling in WSNs. In the following paragraphs the state-of-the art for task mapping and scheduling in WSNs are introduced.

In [121], a fast online collaborative allocation algorithm (CoRAl) is proposed to dynamically reconfigure WSNs according to the sensor node's activity changes (i.e., sleep versus active modes) or new hot spots occurring (e.g. new target is detected).

CoRAl allocates the resources to the tasks so that the system utility is maximised. However, CoRAl does not consider the battery level as a part of sensor node resources and it does not address the energy consumption problem.

In [122], six heuristic task mapping and scheduling techniques, Min-Min, Levelized Weight Tuning, Bottoms Up, Genetic Algorithm, Simplified Lagrangian, Lower Bound and A* are compared and evaluated in heterogeneous ad hoc grid environment. In [122], the application is modelled using the Directed Acyclic Graph (DAG). Min-Min is a task mapping and scheduling technique used in traditional parallel computing [112][113]. In [122], the fitness value is defined as the weighted sum of the execution time and energy consumption required to execute a task in a particular sensor node. In Min-Min used in [122], for each task the fitness value is calculated across all sensor nodes and thus the sensor node that has minimum fitness value is temporality selected and stored with the corresponding task in a pair. Among all node/task pairs, the pair that has the minimum fitness value is permanently selected for mapping. After that, the energy and time availabilities of the selected sensor node and any other sensor nodes that are involved to send/receive any dependencies to/from the selected machine are updated. The procedures are repeated until all tasks are mapped. Levelized Weight Tuning (LWT) is a task mapping and scheduling technique used in distributed heterogeneous environments [114]. In LWT used in [122], a DAG representing the application is arranged into levels according to the data precedence constraints. At each level, each task is assigned a priority based on the size of its output data items. For each task from the low level to the high level and from the high priority to the low priority in each level, the LWT is run to map the tasks to the sensor nodes. The Bottoms Up (BU) heuristic proposed on [122] combines Min-Min and LWT. However, it starts from the highest level to the lowest level. A* is a tree search technique that starts from the root node. A* has been found to be a highly effective method for searching a tree or graph [122]. Simplified Lagrangian (SL) proposed in [122] is a simple version of Lagrangian approaches that have been used for job scheduling in industrial environments [111]. Genetic Algorithm (GA) is used to search a large solutions space to find exact or approximate optimized solution. The GA used in [122] is a modified version of GA used in [123]. With Lower Bound (LB) proposed in [122], the sensor node that has the minimum percentage of energy consumption is chosen. The simulation results in [122] show that GA gives the best performance. The performance metric using GA is better

than Min-Min by 7%. The performance metric in [122] is defined as the summation of the percentage of energy consumed by each sensor node to complete the mapped tasks, averaged across all sensor nodes. On the other hand, the time required to perform GA is high compared to Min-Min. However, unlike the case of WSNs, [122] assumes individual channels for each sensor node and each sensor node can transmit and receive data at the same time. Moreover, [122] ignores the energy consumption to receive a data item and the cost of the initial data item.

In [124], a task allocation heuristic algorithm that consists of three operational phases has been developed to provide energy-balanced task allocation in a single-hop cluster of homogeneous sensor nodes. In the first phase, the tasks are serialized into clusters so that the execution time of the application is minimized. In the second phase, the task clusters are assigned to the sensor nodes that have minimum normalized energy dissipation, which is the sum of energy dissipation of the clusters assigned to the sensor node normalized by the sensor node remaining energy. In the third phase, the CPU voltage levels of tasks are adjusted with the goal of maximizing the system lifetime subject to the application deadline. The process of changing the CPU voltage level is known as Dynamic Voltage Scaling (DVS) [125]. The operation of DVS is mainly based on the fact that the processing energy consumption is proportional with the cube of the CPU voltage [125]. On the other hand, the execution time is reduced with decreasing the CPU voltage. However, [124] assumes the energy consumption to transmit a data item is the same in the sender and receiver, which is not realistic. Additionally, [124] does not employ the broadcast nature of WSNs where sensor nodes are equipped with Omni-directional antennas.

The authors in [126 - 128] proposed a different algorithm for task mapping and scheduling in WSNs. In [126], Energy-Constrained Task Mapping and Scheduling (EcoMapS) algorithm is implemented for energy-constrained application in single-hop clustered. The objective of EcoMapS is to find a task mapping and scheduling solution so that the schedule length is minimised under energy consumption constraint. However, EcoMapS does not guarantee the in-time completion of the application before the application deadline. In [127], a real-time task mapping and scheduling (RT-MapS) algorithm is proposed for collaborative in-network processing in single-hop cluster WSN with enabling DVS feature. In [128], Multihop Task Mapping and Scheduling (MTMS) solution is presented to map and schedule application tasks in multi-hop

cluster WSN. The main goal of RT-MapS and MTMS is to minimize energy consumption subject to meeting the application deadline. MTMS and RT-MapS use DVS and Min-Min algorithms in its operation. However, MTMS and RT-MapS do not allow mapping the task to its immediate predecessors. Additionally, they involve all sensor nodes in the task mapping decision-making. Moreover, the Min-Min algorithm adopted by them is initially introduced in traditional parallel computing for mapping and scheduling independent tasks. Therefore, there are no any dependencies among tasks and in turn there is no communication cost between the processors. In Min-Min, the fitness value for each task is calculated across all sensor nodes and thus the sensor node that has minimum fitness value is temporality selected and stored with the corresponding task in a pair. Among all node/task pairs, the pair that has the minimum fitness value is selected for mapping. Therefore in Min-Min approach, only the selected pair will be permanently mapped and the procedures will be repeated to map other tasks. In case of an application that can be divided into tasks with dependencies, the first pair is permanently selected based on the other pairs and communication between the pairs to exchange the dependencies. If the same procedures are repeated to permanently map the next pair, the calculations that are used to map the first pair will not valid anymore because the pairs that are produced to permanently map the second pair may not be the same as the pairs generated to permanently map the first pair.

Task mapping and scheduling of an application that can be divided into independent tasks is introduced in traditional parallel processing system [112][113]. However, it receives less attention in WSNs.

2.10 Chapter Summary

This chapter introduces the structure, applications, and MAC and routing protocols of WSNs. In this thesis, CSMA/CA is selected as the MAC protocol and DSDV as the routing protocol and both of them are implemented in the simulator to evaluate the proposed tracking, task mapping and scheduling schemes. The structure and requirements for target tracking in WSNs are presented. Background material concerning relevant biological analogies, task mapping and scheduling in WSNs are considered. In the next chapter, the proposed single target tracking scheme is introduced in detail.

Chapter 3 Single Target Tracking in WSNs

3.1 Chapter Introduction

In the previous chapter, background material and a literature review relating to WSNs, target tracking, task mapping and scheduling and biological systems are considered. This chapter presents the proposed Single Target Tracking (STT) scheme in detail. The target dynamic, sensor detection, measurement and energy consumption models are presented. Then, the Extended Kalman Filter (EKF) for STT in WSNs is introduced. After that, the framework and the assumptions for the proposed STT scheme are explained. The target metadata representation is illustrated; then, sampling interval determination, sensor node selection and election are presented. The recovery mechanism and sensor node deployment strategies are then introduced. After that, complete algorithms and protocols for the proposed STT scheme are proposed. Finally, a chapter summary is provided

3.2 Target Dynamic Model

In this chapter, a STT is considered. The target (T) state vector at time step k consists of the target coordinates (i.e., $x_T(k)$ and $y_T(k)$) and velocities (i.e., $\dot{x}_T(k)$ and $\dot{y}_T(k)$) in xy plane and is written in a vector form as follows:

$$\mathbf{X}(k) = \left[x_T(k) \quad \dot{x}_T(k) \quad y_T(k) \quad \dot{y}_T(k)\right]' \tag{3.1}$$

The target location at time step k and the sensor node (s_i) location can be expressed by the following vectors:

$$\mathbf{L}_T(k) = [x_T(k) \quad y_T(k)]' \tag{3.2}$$

$$\mathbf{L}_{s_i} = [x_{s_i} \quad y_{s_i}]' \tag{3.3}$$

The target (T) dynamic is modelled using the discrete-time white noise acceleration model [72][74][75]. Therefore, the system model described in Equation (2.4) at time step k can be written as:

$$\mathbf{X}(k+1) = \mathbf{A}(k)\mathbf{X}(k) + \mathbf{w}(k) \tag{3.4}$$

where $\mathbf{A}(k)=\begin{bmatrix}\mathbf{A}_s(k) & \mathbf{Z}\\ \mathbf{Z} & \mathbf{A}_s(k)\end{bmatrix}$, $\mathbf{A}_s(k)=\begin{bmatrix}1 & \Delta t(k)\\ 0 & 1\end{bmatrix}$, $\mathbf{Z}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$, $\Delta t(k)=t_{k+1}-t_k$ is the sampling interval which is the time between two successive tracking snapshots (i.e., at time steps k and $k+1$) and is calculated at time step k and $\mathbf{w}(k)$ is the process noise which models the target velocity variations (i.e., acceleration) and is assumed to possess a zero-mean White Gaussian Distribution with $\mathbf{Q}(k)$ covariance matrix:

$$\mathbf{Q}(k)=q\begin{bmatrix}\mathbf{Q}_s(k) & \mathbf{Z}\\ \mathbf{Z} & Q_s(k)\end{bmatrix} \tag{3.5}$$

where $\mathbf{Q}_s(k)=\begin{bmatrix}\Delta t^3(k)/3 & \Delta t^2(k)/2\\ \Delta t^2(k)/2 & \Delta t(k)\end{bmatrix}$ and q is scalar that represents the amount of randomness in the process noise.

3.3 Sensor Detection and Measurement Model

As mentioned in Section 2.6.1, passive sensor devices are used to detect the acoustic signals produced form the targets. Therefore, the target is assumed to be an isotropic sound source. Recall from Section 2.6.4, The RSSI method is used to model the acoustic signals. The acoustic power intensity received by sensor node s_i at time step $k+1$ is calculated according to the following model [58][129][130]:

$$P_{s_i}(k+1)=\frac{S(k+1)}{R_{s_i}^n(k+1)} \tag{3.6}$$

where $S(k+1)$ is the emitted acoustic density from the sound source (i.e., the target) at time step $k+1$ which is assumed to be known , $R_{s_i}(k+1)$ is the noisy geometric distance between the sensor node s_i and the target at time step $k+1$ and n is the attenuation decay factor which is typically between 2 to 5 according to the environment and atmospheric conditions [131]. Therefore, by measuring $P_{s_i}(k+1)$, $R_{s_i}(k+1)$ can be calculated using Equation (3.6). Using a group of $n_g(k+1)$ tasking sensor nodes denoted by $S_g(k+1)=\{s_1,s_2,...,s_{n_g(k+1)}\}$ to track the target T at time step $k+1$ and according to Equation (2.5), the measurement model at time step $k+1$ is given by:

$$\mathbf{z}(k+1)=\mathbf{h}[k+1,\mathbf{X}(k+1)]+\mathbf{v}(k+1) \tag{3.7}$$

where $\mathbf{v}(k+1)$ is the measurement noise which is assumed a zero-mean White Gaussian Distribution with $\mathbf{R}(k+1)$ covariance matrix, $\mathbf{z}(k+1)$ is the target noisy measurements vector which is given by:

$$\mathbf{z}(k+1) = \left[R_{s_1}(k+1) \quad R_{s_2}(k+1) \quad \ldots \quad R_{s_{n_g}(k+1)}(k+1) \right]' \tag{3.8}$$

and $\mathbf{h}[k+1, \mathbf{X}(k+1)]$ is the measurement function which is calculated as:

$$\mathbf{h}[k+1, \mathbf{X}(k+1)] = \begin{bmatrix} \sqrt{[\mathbf{L}_T(k+1) - \mathbf{L}_{s_1}]'[\mathbf{L}_T(k+1) - \mathbf{L}_{s_1}]} \\ \sqrt{[\mathbf{L}_T(k+1) - \mathbf{L}_{s_2}]'[[\mathbf{L}_T(k+1) - \mathbf{L}_{s_2}]} \\ . \\ . \\ \sqrt{[\mathbf{L}_T(k+1) - \mathbf{L}_{s_{n_g}(k+1)}]'[\mathbf{L}_T(k+1) - \mathbf{L}_{s_{n_g}(k+1)}]} \end{bmatrix} \tag{3.9}$$

where $\sqrt{[\mathbf{L}_T(k+1) - \mathbf{L}_{s_i}]'[\mathbf{L}_T(k+1) - \mathbf{L}_{s_i}]}$ is the Euclidean distance in matrix-form between the target T and sensor node s_i at time step $k+1$. The measurement noise variances of the sensor nodes are assumed to be independent. Therefore, the measurement noise covariance matrix is defined as $\mathbf{R}(k+1) = diag(\sigma^2_{s_1}, \sigma^2_{s_2}, ..., \sigma^2_{s_{n_g}(k+1)})$. The process and measurement noises are assumed to be independent of time and with respect to each other.

3.4 Energy Consumption Model

Energy is consumed during sensing, communication and processing activities. As in [132] and [133], the energy the transmitter consumes is from the dissipated energy to run the radio electronics and the power amplifier while the receiver consumes energy to run the radio electronics. This is shown in Figure 18.

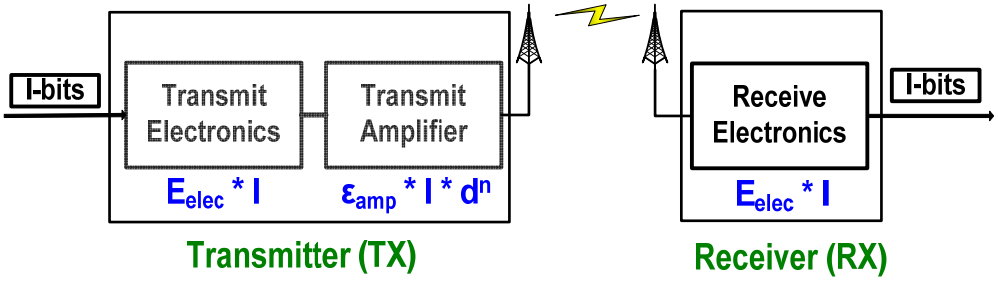


Figure 18 Radio Consumption Model

Therefore, the energy consumption to transmit l-bit message over a distance d is:

$$E_{TX}(l,d) = \begin{cases} E_{elec}.l + \varepsilon_{FS}.l.d^2 & d \prec d_o \\ E_{elec}.l + \varepsilon_{MP}.l.d^4 & d \geq d_o \end{cases} \tag{3.10}$$

where d_o is the threshold value which is the border between the free space transmission (i.e., $d \prec d_o$) and the multipath fading transmission (i.e., $d \geq d_o$). d^2 and d^4 are used to model the power loss in free space and multipath channels, respectively. E_{elec} is the electronic energy that depends on factors such as coding, modulation and filtering, and ε_{FS} and ε_{MP} are the amplifier energy (ε_{amp}) for free space and multipath channels respectively. The energy consumption to receive l-bit message is:

$$E_{RX}(l) = E_{elec}.l \tag{3.11}$$

For a CPU with clock frequency f, the energy consumption to execute N clock cycles [133][134] is:

$$E_{comp}(N) = NCV_{dd}^2 + V_{dd}\left(I_o e^{\frac{V_{dd}}{n.V_T}}\right)\left(\frac{N}{f}\right) \tag{3.12}$$

where $\mathrm{V_{dd}}$ is the supply voltage, V_T is the thermal voltage, the CPU clock speed is modelled as a function of the supply voltage through the equation $f \approx K(\mathrm{V_{dd}} - c)$ and c, C, I_0, n, K are CPU dependent parameters.

3.5 Extended Kalman Filter for Single Target Tracking

In the EKF [72][74][75], the predicted target state is given by:

$$\hat{\mathbf{X}}(k+1 \mid k) = \mathbf{A}(k)\hat{\mathbf{X}}(k \mid k)) \tag{3.13}$$

with associated predicted covariance matrix given by:

$$\mathbf{P}(k+1 \mid k) = \mathbf{A}(k)\mathbf{P}(k \mid k)\mathbf{A}'(k) + \mathbf{Q}(k) \tag{3.14}$$

The predicted measurement vector is calculated as follows:

$$\hat{\mathbf{z}}(k+1 \mid k) = \mathbf{h}[k+1, \hat{\mathbf{X}}(k+1 \mid k)] \tag{3.15}$$

The Jacobian matrix of $\mathbf{h}$ at $\mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k)$ is:

$$\begin{aligned}\mathbf{H}(k+1) &= \frac{\partial \mathbf{h}[k+1, \mathbf{X}(k+1)]}{\partial \mathbf{X}(k+1)}, \quad \text{at} \quad \mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k) \\ &= [\mathrm{H_{ij}}] \qquad\qquad 1 \le i \le n_g(k+1), \quad 1 \le j \le 4\end{aligned} \tag{3.16}$$

where

$$\mathrm{H_{ij}} = \begin{cases} \dfrac{\partial \mathbf{h}_{\mathrm{i1}}[k+1, \mathbf{X}(k+1)]}{\partial x_T(k+1)} & \text{at } \mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k) \quad \mathrm{j}=1 \\ \dfrac{\partial \mathbf{h}_{\mathrm{i1}}[k+1, \mathbf{X}(k+1)]}{\partial \dot{x}_T(k+1)} & \text{at } \mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k) \quad \mathrm{j}=2 \\ \dfrac{\partial \mathbf{h}_{\mathrm{i1}}[k+1, \mathbf{X}(k+1)]}{\partial y_T(k+1)} & \text{at } \mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k) \quad \mathrm{j}=3 \\ \dfrac{\partial \mathbf{h}_{\mathrm{i1}}[k+1, \mathbf{X}(k+1)]}{\partial \dot{y}_T(k+1)} & \text{at } \mathbf{X}(k+1) = \hat{\mathbf{X}}(k+1 \mid k) \quad \mathrm{j}=4 \end{cases} \tag{3.17}$$

By performing the partial integrations for Equation (3.9), Equation (3.17) leads to:

$$\mathrm{H_{ij}} = \begin{cases} \dfrac{\hat{x}_T(k+1 \mid k) - x_{s_i}}{\sqrt{[\hat{\mathbf{L}}(k+1 \mid k) - \mathbf{L}_{s_i}]'[\hat{\mathbf{L}}(k+1 \mid k) - \mathbf{L}_{s_i}]}} & \mathrm{j}=1 \\ 0 & \mathrm{j}=2 \\ \dfrac{\hat{y}_T(k+1 \mid k) - y_{s_i}}{\sqrt{[\hat{\mathbf{L}}(k+1 \mid k) - \mathbf{L}_{s_i}]'[\hat{\mathbf{L}}(k+1 \mid k) - \mathbf{L}_{s_i}]}} & \mathrm{j}=3 \\ 0 & \mathrm{j}=4 \end{cases} \tag{3.18}$$

where $\hat{\mathbf{L}}_T(k+1 \mid k)$ is the target predicted location and is calculated as:

$$\hat{\mathbf{L}}_T(k+1 \mid k) = [\hat{x}_T(k+1 \mid k) \quad \hat{y}_T(k+1 \mid k)] \tag{3.19}$$

and for any variable x:

$$\frac{\partial}{\partial x}\left(\sqrt{(x-a)^2 + (y-b)^2}\right) = \frac{x-a}{\sqrt{(x-a)^2 + (y-b)^2}} \tag{3.20}$$

$$\frac{\partial}{\partial y}\left(\sqrt{(x-a)^2 + (y-b)^2}\right) = \frac{y-b}{\sqrt{(x-a)^2 + (y-b)^2}} \tag{3.21}$$

In the update stage where the measurements at time step $k+1$ are available, the measurement residual which is the difference between the actual and predicted measurements can be calculated as follows:

$$\mathbf{r}(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1 \mid k) \tag{3.22}$$

with associated residual or innovation covariance matrix:

$$\mathbf{S}(k+1) = \mathbf{R}(k+1) + \mathbf{H}(k+1)\mathbf{P}(k+1 \mid k)\mathbf{H}'(k+1) \tag{3.23}$$

The update state estimate is give by:

$$\hat{\mathbf{X}}(k+1 \mid k+1) = \hat{\mathbf{X}}(k+1 \mid k) + \mathbf{K}(k+1)\mathbf{r}(k+1) \tag{3.24}$$

with associated updated covariance matrix:

$$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{K}(k+1)\mathbf{S}(k+1)\mathbf{K}'(k+1) \tag{3.25}$$

where the filter gain is defined as:

$$\mathbf{K}(k+1) = \mathbf{P}(k+1 \mid k)\mathbf{H}'(k+1)\mathbf{S}^{-1}(k+1) \tag{3.26}$$

A detailed derivation of the EKF equations and further explanation about EKF are found in [72], [74] and [75].

3.6 Multi-Sensor Adaptive Single Target Tracking Framework

In this chapter, a Multi-Sensor Adaptive Single Target Tracking (MS-ASTT) scheme is proposed for STT in WSNs. Like [5], the STT framework introduced in this section is energy efficient and similar in some respects to the framework proposed in [5]. However, a comparison between the STT scheme in [5] and the proposed MS-ASTT scheme is provided in Chapter 7. Figure 19 shows the framework of the proposed MS-ASTT scheme. The sensor nodes are randomly deployed according to a uniform distribution in the sensing area to track the target. Each sensor node knows its location using GPS [37] or triangulation [38] where some sensor nodes called anchor or beacon nodes determine their positions using GPS and other sensor nodes use triangulation to calculate their positions by using the known anchor positions and their distances to these anchors. Each sensor node knows the location and battery level of its neighbours. Therefore, if any sensor node performs communication or processing activities which lead to significantly reduced battery level, it informs its neighbours about the updated battery level. The sensor nodes have the capability to measure the target range, for example by using an acoustic signal emitted from the target. To improve the energy efficiency, three operational modes are assumed for the sensor node which are sensing, communication and sleeping. When the target enters the sensing area, the border sensor nodes detect it. Therefore, the border sensor nodes are in both sensing and communication modes all the time. All other sensor nodes are in sleeping mode. Therefore, these sensor nodes have to be triggered to “wakeup” if they are needed for communication and/or sensing. Sensor nodes in sleeping mode use a low energy

communication channel [134] to receive trigger message from other sensor nodes. The BS or sink is responsible for forwarding the desired information from the WSN to the headquarters (i.e., main controller) through the Internet, via satellite or other wireless technology.

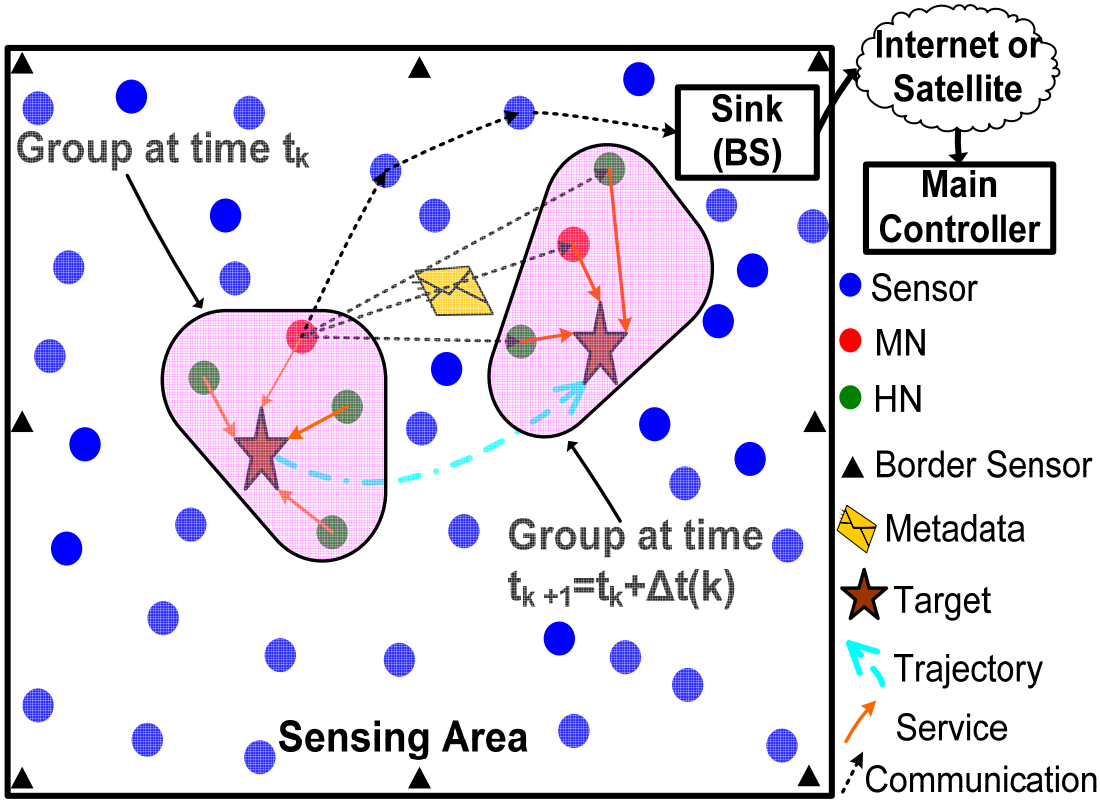


Figure 19 MS-ASTT Framework in WSNs

As shown in Figure 19, at each tracking time step, one member of the group that is formed is elected to manage the tracking scheme. This is the Main Node (MN) and the other members are called Helper Nodes (HNs). The main goals of the proposed MS-ASTT scheme are to proactively select the next group, elect one sensor node to be the MN, calculate the next sampling interval and perform recovery in the case of target loss so that the network lifetime, energy efficiency and tracking accuracy are improved. The tracking error is used to indicate the tracking accuracy. Two definitions for tracking error are used in this thesis. Firstly, the tracking error can be defined as the difference between the real state and the updates or predicted state of the target. Secondly, the tracking error is defined based on the uncertainty associated with the updated or predicted covariance matrix. The tracking initialization is started when a mobile target enters the sensing area. The border sensors sense the target, localize the target using a triangulation technique, which is presented in Section 2.6.4, and set the initial error covariance. Thus, the border sensors predict the next target state using EKF, select the

next group, perform the election of the next MN, initiate the sampling interval to its minimum value and trigger the next group to wakeup. The group of sensor nodes track the target cooperatively. The algorithms that run in the group of sensor nodes are presented in Section 3.13.

3.7 Target Metadata

In this thesis, behavioural data obtained while tracking the target including the target's previous locations is recorded as metadata. As shown in Figure 20, At each tracking time step k, the target metadata $TMD(k)$ consists of information about the target's previous movement pattern, predicted state $\hat{\mathbf{X}}(k+1|k)$ and its covariance matrix $\mathbf{P}(k+1|k)$ (i.e., predicted error) and next tracking time (t_{k+1}), which is the current time plus the current sampling interval (i.e., $t_k + \Delta t(k)$). The predicted state and its covariance matrix are used to calculate the updated state and its covariance matrix using EKF. The next tracking time allows the next tracking group to be made aware of the target arrival time in their vicinity.

The location metadata $M(k, K_m)$ that includes the last $K_m \leq k$ target update locations is calculated from the target's previous movement pattern. As shown in Section 3.8, the target location metadata is used to calculate the sampling interval at each tracking time step. Additionally, as shown in Chapter 4, the target location metadata is employed to compute the target importance and the number of allowable iterations for the local search in the case of MTT. Therefore, the location metadata is calculated by the sensor nodes in distributed manner. In this case, a sensor node requires information about the past movement pattern of the target. This could lead to an increase in the size of target metadata message. However, message coding and compression can be used to reduce the size [155] if necessary. On the other hand as shown in Section 3.6, since the main controller has all the information about the target tracking states, it can be employed to calculate the target location metadata in a centralized fashion. In this thesis, the target metadata message is assumed to be coded to a small size so that the CSMA/CA can be used without the need of using RTS/CTS handshaking.

Movement Pattern	Predicted State	Predicted Error	Next Tracking Time

Figure 20 Target Metadata

The location metadata of the target indicates the historical movement pattern of the target. It can be calculated from the previous target locations that are computed from the

tracking algorithms such as EKF. At time step k, the location metadata of the last K_m tracking snapshots (i.e., tracking events) are modelled as follows:

$$M(k,K_m)=\frac{d_n(k,K_m)}{d_t(k,K_m)} \tag{3.27}$$

where $d_t(k,K_m)$ and $d_n(k,K_m)$ are the total and the net travel of the target during the last K_m tracking snapshots, respectively. The net travel of the target, $d_n(k,K_m)$ is the distance between the updated target location at time step $k-K_m$ and the current target location at time step k. The updated target location is expressed in the following vector form:

$$\hat{\mathbf{L}}_T(k\mid k)=[\hat{x}_T(k\mid k)\quad \hat{y}_T(k\mid k)] \tag{3.28}$$

Therefore, the net travel of the target, $d_n(k,K_m)$ can be calculated according to:

$$d_n(k,K_m)=\sqrt{[\hat{\mathbf{L}}_T(k\mid k)-\hat{\mathbf{L}}_T(k-K_m\mid k-K_m)]'[\hat{\mathbf{L}}_T(k\mid k)-\hat{\mathbf{L}}_T(k-K_m\mid k-K_m)]} \tag{3.29}$$

The total travel of the target, $d_t(k,K_m)$ is the overall distance between the last K_m tracking snapshots. The total travel $d_t(k,K_m)$ is calculated according to:

$$d_t(k,K_m)=\sum_{j=k-K_m}^{k-1}\sqrt{[\hat{\mathbf{L}}_T(j\mid j)-\hat{\mathbf{L}}_T(j+1\mid j+1)]'[\hat{\mathbf{L}}_T(j\mid j)-\hat{\mathbf{L}}_T(j+1\mid j+1)]} \tag{3.30}$$

As shown in Figure 21 (i), the maximum value of the net travel is equal to the total travel where the target is moving in straight line. In this case, the location metadata of the target at point "D" is equal to one. This indicates that the target is moving in a uniform fashion. On the other hand as shown in Figure 21 (ii), the minimum value of the net travel is zero when the target returns to the same point it started from. In this case, the location metadata of the target at point "D" is equal to zero which indicates that the target is moving in random manner. Otherwise as shown in Figure 21 (iii), the location metadata of the target is between one and zero. Therefore, $M(k,K_m)$ is reduced when the target starts to move in bends. The location metadata of the target is thus bounded in the interval $0\le M(k,K_m)\le 1$.

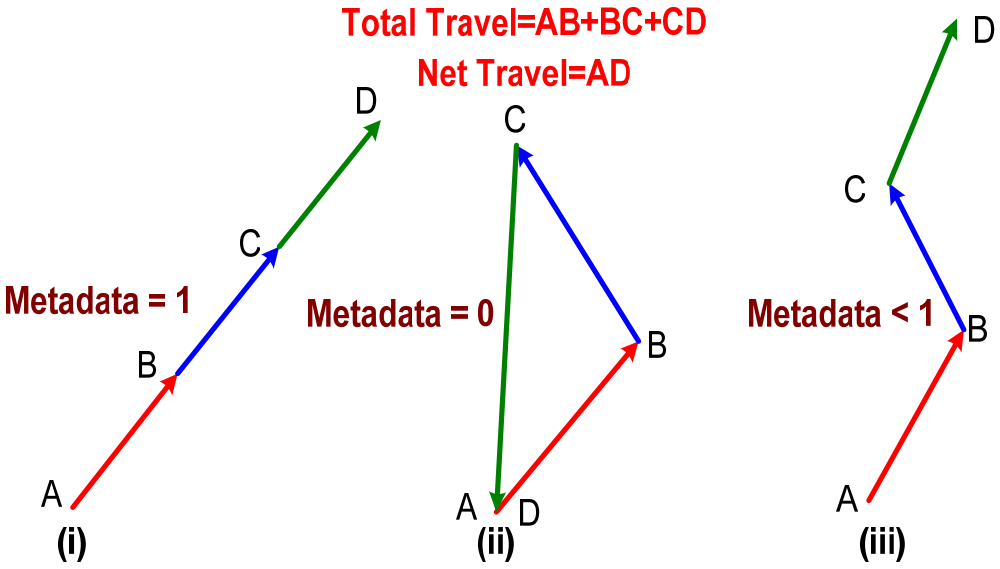


Figure 21 Target Location Metadata

3.8 Adaptive Sampling Interval Selection Algorithm

Unlike STT schemes using a uniform sampling interval, which are introduced in Chapter 2, the proposed MS-ASTT scheme adaptively calculates the sampling interval to improve the energy efficiency and maintain a good accuracy with seamless target tracking. The value of the sampling interval is reduced when the target manoeuvres in an unpredictable fashion. This improves the tracking accuracy with seamless target tracking. Moreover, the prediction of the next target state presented in Section 3.5 is more likely to succeed. Conversely, the sampling interval is set to a larger value when the target travels in a uniform, predictable manner. This improves the energy efficiency of the WSN. One example is shown in Figure 22. Figure 22 (a) illustrates the trajectory of a mobile target. The sampling interval is large when the target travels in straight line (i.e., a uniform manner) while it should be small during the target manoeuvrings. In Figure 22 (b), the location metadata at time t_4 and t_7 are calculated using three previous tracking snapshots. The location metadata at time t_7, $M(7,3)$ is equal to 1 and in turn the sampling interval is set to a large value because the target is moving in a uniform manner. On the other hand, the location metadata at time t_4, $M(4,3)$ is less than 1 and in turn the sampling interval is set to a smaller value because the target is moving in an unpredictable manner.

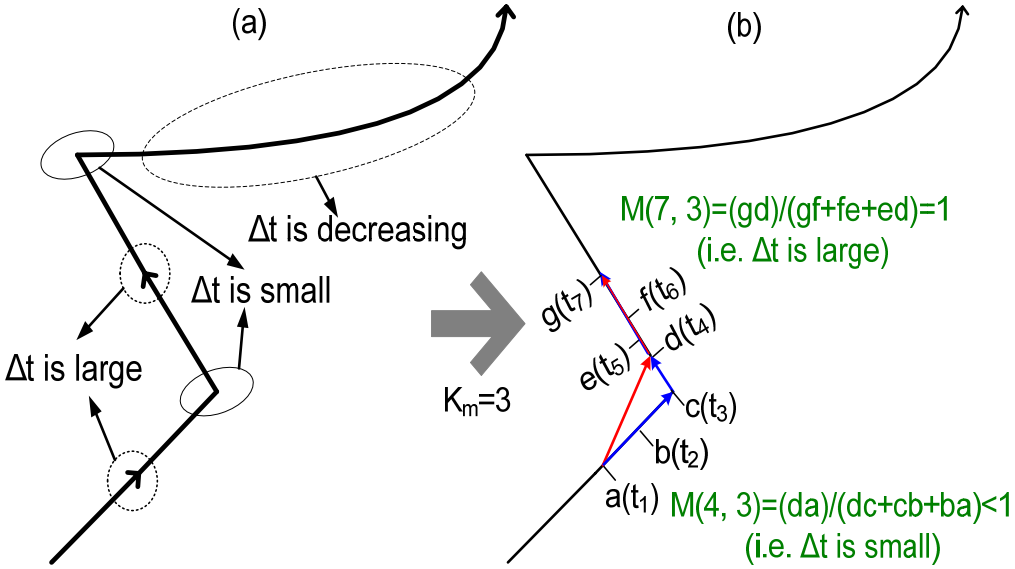


Figure 22 Adaptive Sampling Interval

The sampling interval at time step k, $\Delta t(k)$ is permitted to adaptively change in the interval $T_{\min} \le \Delta t(k) \le T_{\max}$. Therefore, $T_{\min}$ is the minimum sampling interval, which should be less than the time required for channel access, propagation delay and any necessary data processing. $T_{\max}$ is the maximum sampling interval, which is determined according to the amount of the motion randomness and manoeuvring of the target [135]. The sampling interval, $\Delta t(k)$ is calculated based on the current location metadata, $M(k, K_m)$ and the previous sampling interval, $\Delta t(k-1)$. The measured sampling interval, $\Delta t_m(k)$ is defined to model the impact of the location metadata, $M(k, K_m)$ such that $T_{\min} \le \Delta t_m(k) \le T_{\max}$. As shown in Figure 23, $\Delta t_m(k)$ is modelled as a liner function of the location metadata, $\Delta t_m(k) = f(M(k, K_m))$, according to the following equation:

$$\Delta t_m(k) = (T_{\max} - T_{\min}) M(k, K_m) + T_{\min} \tag{3.31}$$

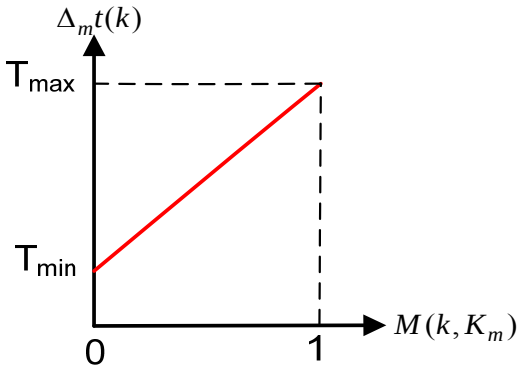


Figure 23 Sampling Interval as a Function of Location Metadata

Therefore the current sampling interval $\Delta t(k)$ is the weighted sum of the current the measured sampling interval, $\Delta t_m(k)$ and the previous sampling interval, $\Delta t(k-1)$ as follows:

$$\Delta t(k) = \alpha \Delta t(k-1) + (1-\alpha)\Delta t_m(k) \tag{3.32}$$

where, $\alpha \in [0 \ \ 1]$. Therefore, if the measured sampling interval changes from a low value to high one, the sampling interval smoothly increases to reach the high value and vice versa. This gives extra confidence that the target completes its previous motion pattern. Therefore, the sampling interval increases smoothly when the target changes its movement pattern from a manoeuvring pattern to a uniform pattern to avoid the unexpected movement of the target during changing its movement pattern.

Higher speed targets require a smaller sampling interval. Therefore, in Figure 23, the ratio between $T_{\max}$ and $T_{\min}$ is selected according to the speed of the target. If the target moves at variable speed, the ratio between $T_{\max}$ and $T_{\min}$ should be adaptively calculated according to the current target speed. However, within Chapter 7 the proposed tracking algorithms are evaluated during periods when the target is travelling with constant speed. Nevertheless, the proposed tracking algorithms are evaluated for different target speeds.

3.9 Sensor Nodes Selection Management

In this section, the sensor nodes selection algorithm is presented. At the beginning, the target model is presented. After that, the selection strategy of the sensor nodes is introduces. Finally, the adaptive group size of the sensor nodes to track the target is presented.

3.9.1 Target Model

In this thesis, as shown in Figure 24, the target to be tracked, such as human being or a moving vehicle, is treated as a virtual chemical emitter that influences the sensor nodes with a varying strength which is determined according to the target importance and the target proximity to the sensor nodes. Therefore, the target's influence on the sensors nodes is referred as chemical diffusion strength (G). The chemical diffusion strength of the target decreases with distance from the target to the sensor node. More details are provided in Section 3.9.2.

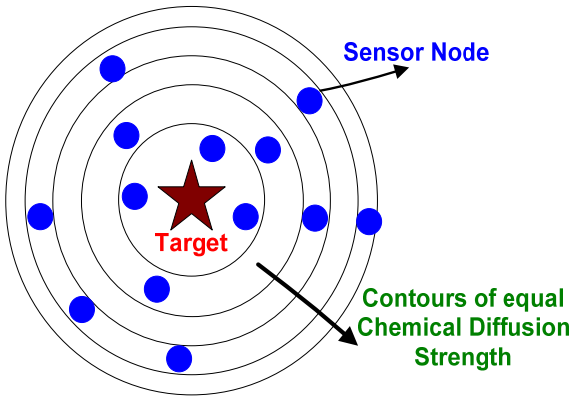


Figure 24 Chemical Diffusion Strength

3.9.2 Sensor Nodes Selection Algorithm

Unlike [16], [82], [96] and [98], the MS-ASTT scheme uses multi-sensors to track the target to improve the tracking accuracy and continuity [5]. The sensor nodes selection algorithm is primarily based on the information associated with the predicted target location PDF. At each tracking time step k, using the predicted target location PDF, the sensor nodes that are most influenced by the target are proactively selected to form the group, $S_g(k+1)$ that will track the target at the time step $k+1$. The group, $S_g(k+1)$ is selected from the neighbours ($S_n(k)$) of the current main node, $MN(k)$. The Mahalanobis distance $D(k+1|k,s_i)$ [136], which considers the predicted target location covariance $\mathbf{P}(k+1|k)$ in its calculations, is obtained between the target predicted location PDF and each sensor node $s_i \in S_n(k)$ as follows:

$$D(k+1|k,s_i)=\sqrt{[\mathbf{L}_{s_i}-\hat{\mathbf{L}}_T(k+1|k)]'\boldsymbol{\Sigma}_T^{-1}(k+1|k)[\mathbf{L}_{s_i}-\hat{\mathbf{L}}_T(k+1|k)]} \tag{3.33}$$

$\boldsymbol{\Sigma}_T(k+1|k)$ is the predicted target location covariance matrix. If $\mathbf{P}(k+1|k)$ is in a form of:

$$\mathbf{P}(k+1|k)=\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} \tag{3.34}$$

then, $\boldsymbol{\Sigma}_T(k+1|k)$ is calculated as:

$$\boldsymbol{\Sigma}_T(k+1|k)=\begin{bmatrix} P_{11} & P_{13} \\ P_{31} & P_{33} \end{bmatrix} \tag{3.35}$$

$D(k+1|k,s_i)$ and the target importance, $Z_T(k+1)$ are used to model the target chemical diffusion strength, $G(k+1|k,s_i)$ as follows:

$$G(k+1 \mid k, s_i) = \frac{Z_T(k+1)}{D(k+1 \mid k, \mathrm{s_i})} \tag{3.36}$$

$Z_T(k+1)$ is used for multi-target tracking to give priority to more important targets. Therefore, if the sensor node is located in the sensing areas of more than one target at the same time, preference is given to target that has the strongest chemical strength (G) as evaluated by the sensor node. This is presented in detail in Chapter 4. In sensor nodes selection, preference is given to the sensor nodes that have the strongest chemical diffusion strength (G) of the target. Therefore, the selection fitness function by which the sensor nodes will be selected is computed as follows:

$$f_S[k+1 \mid k, s_i, S_n(k)] = \frac{G(k+1 \mid k, s_i)}{\sum_{\forall j \in S_n(k)} G(k+1 \mid k, s_j)} \tag{3.37}$$

The ℓth sensor node, $g_\ell \in S_g(k+1)$ where $1 \le \ell \le n_g(k+1)$ is selected so that:

$$g_\ell = \arg_{s_i} \max \left\{ f_S[k+1 \mid k, s_i, S_n(k)] : \; s_i \in S_n \setminus \bigcup_{j=1}^{j=\ell-1} g_j \right\} \tag{3.38}$$

where $S_n \setminus \bigcup_{j=1}^{j=\ell-1} g_j$ is the set of $S_n(k)$ members excluding the ones from g_1 to $g_{\ell-1}$, and $n_g(k+1)$ is the group size at time step k+1.

3.9.3 Adaptive Group Size Algorithm

Unlike the STT researches proposed in Chapter 2, the size $n_g(k+1)$ of next sensor nodes group $S_g(k+1)$ is adaptively changed according to the tracking error at time $k+1$ to enhance the tracking accuracy, at each tracking time step k. However, increasing the number of group members leads to increase the energy consumption. Therefore, $n_g(k+1)$ is assumed to be bounded between a smallest value of $n_g^{\min}$ and a biggest value of $n_g^{\max}$ where $n_g^{\max} > n_g^{\min}$. The "trace" of the covariance matrix is proportional to the circumference of the rectangular region of covariance ellipsoid [82]. Therefore, the updated tracking error at time step $k+1$ is defined as follows:

$$\psi\{\mathrm{k}+1, S_g(k+1)\} = tarce\{\mathbf{\Sigma}_T(k+1 \mid k+1)\} \tag{3.39}$$

where $\mathbf{\Sigma}_T(k+1 \mid k+1)$ is the updated target location covariance matrix which is a part from $\mathbf{P}(k+1 \mid k+1)$ and can be calculated by the same way on which $\mathbf{\Sigma}_T(k+1 \mid k)$ is

calculated in Equation (3.35). $\mathbf{P}(k+1|k+1)$ can be calculated using Equation (3.25) without knowledge of the real measurements of $S_g(k+1)$ members. The adaptive group size algorithm shown in Figure 25 tries to reduce the tracking error by increasing the group size.

1. **if** $n_n(k) \geq n_g^{\min}$ **do:**
2. $n_g(k+1) = n_g^{\min}$;
3. **while** $n_g(k+1) \leq n_n(k)$ **do:**
4. Select $S_g(k+1)$ using Equation (3.37) and (3.38);
5. **if** $n_g(k+1) == n_g^{\max}$ **do:**
6. Go to Step 13;
7. Calculate $\psi\{\mathrm{k}+1, S_g(k+1)\}$ using Equation (3.39);
8. **if** $\psi\{\mathrm{k}+1, S_g(k+1)\} > \psi_0$ AND $n_g(k+1) \leq n_n(k)-1$ **do:**
9. Increment $n_g(k+1)$;
10. **end while;**
11. **else do:**
12. Failure to track the target;
13. Finish;

Figure 25 Adaptive Group Size Algorithm

In the Figure 25, in line 1, $n_n(k)$ is the size of the neighbours, $S_n(k)$, of the current main node, $MN(k)$. In line 8, ψ_0 is a predefined tracking error threshold. The tracking accuracy is considered good enough if the tracking error is less than a predefined tracking error threshold. To improve the energy consumption, $n_g(k+1)$ is initially set to $n_g^{\min}$. If the tracking accuracy is not satisfactory, $n_g(k+1)$ is incremented. The maximum value of $n_g(k+1)$ should not exceed $n_n(k)$ and $n_g^{\max}$.

3.10 Sensor Node Election

In this section, the group is classified into one MN and a number of HN(s). The MN typically performs more processing and communication activities. Hence, choosing the MN is a crucial issue to maximize the network lifetime and energy saving. Based on energy models proposed in Section 3.4, the data transmission energy consumption is proportional to the square of the distance between the source and the destination. Node centrality is defined to indicate how much the sensor node is in the group centre.

Therefore, the energy-efficient communication is maximized by selecting the MN to be the sensor node that has largest node centrality because the distances to the other sensor nodes are minimized. Node centrality of the sensor node, s_i in the group, $S_g(k+1)$ is calculated according to the following equation:

$$C(k+1,s_i)=\frac{1}{\sum_{\forall j\in S_g(k+1)}\sqrt{[\mathbf{L}_{s_i}-\mathbf{L}_{s_j}]'[\mathbf{L}_{s_i}-\mathbf{L}_{s_j}]}} \tag{3.40}$$

Unlike the election techniques proposed in Chapter 2 for the related work, load balancing among the group of nodes is another important factor to be considered in the election algorithm, especially when the remaining energy in the nodes is diverse. This means that load balancing is improved by selecting the node that has maximum remaining energy as the MN. Therefore, the next MN, $MN(k+1)$, is elected so that it has the largest election fitness function, f_E, which is computed as follows:

$$f_E[k+1,s_i,S_g(k+1)]=\delta\times\frac{C(k+1,s_i)}{\sum_{\forall j\in S_g(k+1)}C(k+1,s_j)}+(1-\delta)\frac{E_{s_i}}{\sum_{\forall j\in S_g(k+1)}E_{s_j}} \tag{3.41}$$

where E_{s_i} is the remaining energy of sensor node s_i and $\delta\in[0\ \ 1]$ is a weighting parameter used to balance the load with the energy consumption. The energy consumption in the network is reduced when δ is set to 1 (i.e., considering only the node centrality in the election algorithm). However, if δ is set to 1, a scenario where the elected MN has the smallest energy remaining could cause the battery to run out and bring about death of the MN. Death of nodes creates holes in the network causing connectively loss and reduction in the network connectivity lifetime. Therefore, considering the load balancing is crucial for improving the network lifetime.

Setting delta to 0 means only the residual energy of the nodes is considered by the election algorithm. Although this will delay the onset of node death and increase the network connectivity lifetime, by itself it can lead to less desirable election results. If the group of nodes involved in the election of the MN has roughly the same residual energy, then the centrality of the candidate MN should be given precedence even though this node might have marginally less residual energy than its peers. Its superior relative location could result in overall energy savings as the communication penalty between the MN and the HNs is reduced. Therefore, the weighting parameter (δ) is

adaptively chosen according to the variation in the remaining energy between the group members as follows:

$$\delta = \frac{E_{\min}}{E_{\max}} = \frac{\min(E_{s_i})}{\max(E_{s_i})} \qquad \forall s_i \in S_g(k+1) \tag{3.42}$$

where, $E_{\min}$ and $E_{\min}$ is the minimum and maximum remaining energy of the group nodes, respectively. Therefore, when the variation in the remaining energy between the group members is high, δ will be small and in turn the load balancing factor in Equation (3.41) is strongly considered, and vice versa. The MN at time $k+1$ is selected based on the following equation:

$$MN(k+1) = \arg_{s_i} \max f_E[k+1, s_i, S_g(k+1)] \tag{3.43}$$

3.11 Tracking Recovery Mechanism

One of the main requirements of target tracking applications is reliability such that the target is monitored at all times. Tracking can fail due to random and abrupt target manoeuvring. In this section a recovery mechanism is developed to provide seamless tracking reliability in case of target loss. Figure 26 shows a scenario of losing the target. At time step k, the next target location is predicted and the next sensor group is formed. However, the target changes its predictable direction sharply. This causes the tracking algorithm to lose the track of the target.

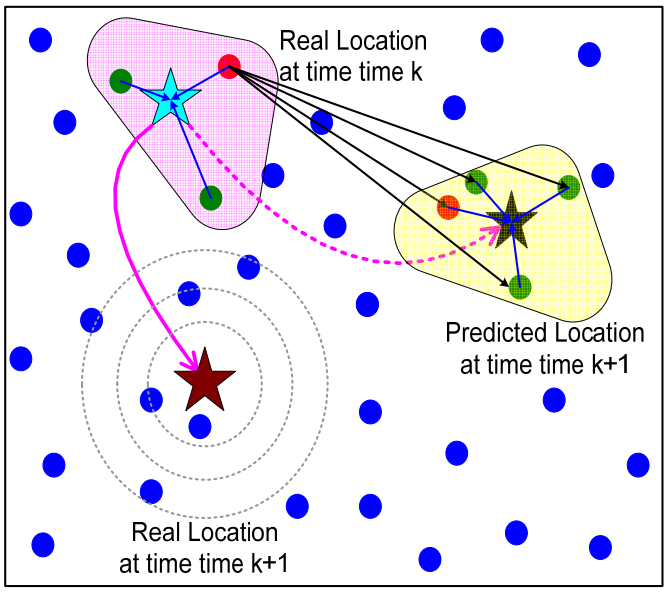


Figure 26 Target Lost Scenario

Generally, the recovery mechanism can be invoked in the following situations:

(1) Accuracy Failure: A low value of tracking accuracy indicates that the target may be lost in subsequent time steps. Tracking can be recovered in case of insufficient tracking accuracy. In this case, a tracking accuracy threshold has to be defined to allow the tracking system to decide when the recovery procedure should be invoked. Tracking accuracy can be predicted for the next tracking snapshot. Therefore, the tracking can be proactively adjusted to avoid losing of the target.

(2) Prediction Failure: The target sometimes changes its direction unexpectedly so that the next predicted group would not be able to detect it. Therefore, the target will be lost and a recovery operation has to be performed to recapture it.

(3) Selection Failure: In some cases, the predicted PDF of the target location is not accurate enough and in turn, the target will not be in the sensing region of all the next group nodes. Therefore, recovery is required to ensure more sensor nodes are involved to track the target.

(4) Node Failure: Recovery is needed as a result of sensor node failure within the tracking group. Node failure can be caused by battery drain, or failure of the software or hardware.

In this thesis, recovery is invoked in the case of prediction and selection failures. As shown in Figure 27, recovery is performed in levels to reduce the energy consumption.

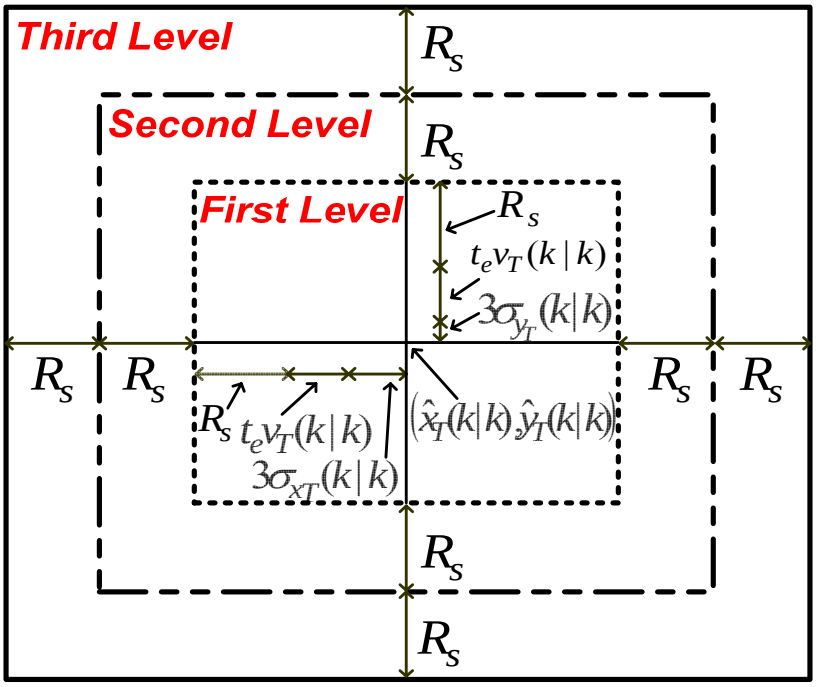


Figure 27 Target Recovery Levels

At each level, more nodes are involved to capture the lost target. At each tracking time step $k+1$, the $MN(k+1)$ activates a timer for recovery (*Timer_recovery*) which

determines when recovery will start if one or more nodes of the group $S_g(k+1)$ does not detect the target. If the timer "*Timer_recovery*" expires and $MN(k+1)$ does not receive a measurement from a particular HN, it will assume that the HN has not detected the target. Thus, after triggering $MN(k)$ to wakeup, the node $MN(k+1)$ informs the old $MN(k)$ about the loss of the target. The node $MN(k)$ will initiate first-level recovery by informing the first-level recovery nodes to wakeup to capture the target. In the first-level recovery, the recovery nodes (S_r^1) are the nodes inside the rectangle:

$$\begin{aligned} \left| y - \hat{y}_T(k \mid k) \right| &\le 3\sigma_{y_T}(k \mid k) + v_T(k \mid k) \times t_e + R_s \\ \left| x - \hat{x}_T(k \mid k) \right| &\le 3\sigma_{x_T}(k \mid k) + v_T(k \mid k) \times t_e + R_s \end{aligned} \tag{3.44}$$

where, R_s is the sensing range, t_e is the time elapsed since the target was last sensed, σ_{x_T} and σ_{y_T} is the standard deviation of updated target x and y locations, respectively, and $v_T(k \mid k)$ is the target updated speed which is defined.

$$v_T(k \mid k) = \sqrt{\dot{x}_T^2(k \mid k) + \dot{y}_T^2(k \mid k)} \tag{3.45}$$

In Equation (3.44), based on the normal distribution empirical rule [137], three standard deviations from the mean are considered to guarantee the recovery nodes are within of the predicted target location with 99.7% certainty. Each recovery node initiates sensing to find the target and sends its measurement to the node $MN(k)$ if it detects the target.

At the time of informing the first-level recovery nodes to wakeup and attempt to capture the target, the node $MN(k)$ activates the next recovery level timer (*Timer_levels*) after which the next level of recovery will be performed if $MN(k)$ does not receive at least three target measurement readings from current recovery nodes. In the second level recovery, the recovery nodes (S_r^2) are inside the rectangle:

$$\begin{aligned} \left| y - \hat{y}_T(k \mid k) \right| &\le 3\sigma_{y_T}(k \mid k) + v_T(k \mid k) \times t_e + 2R_s \\ \left| x - \hat{x}_T(k \mid k) \right| &\le 3\sigma_{x_T}(k \mid k) + v_T(k \mid k) \times t_e + 2R_s \end{aligned} \tag{3.46}$$

excluding the recovery nodes from the first level (S_r^1). Therefore, the recover nodes in ℓth level recovery are defined as:

$$S_r^{\ell} = S_t^{\ell} / \bigcup_{j=1}^{j=\ell-1} S_r^j \tag{3.47}$$

where, S_t^{ℓ} is the set of the nodes inside the rectangle:

$$\begin{aligned} \left| y - \hat{y}_T(k \mid k) \right| &\le 3\sigma_{y_T}(k \mid k) + v_T(k \mid k) \times t_e + \ell R_s \\ \left| x - \hat{x}_T(k \mid k) \right| &\le 3\sigma_{x_T}(k \mid k) + v_T(k \mid k) \times t_e + \ell R_s \end{aligned} \tag{3.48}$$

and $S_t^\ell / \bigcup_{j=1}^{j=\ell-1} S_r^j$ denotes the nodes inside S_t^ℓ excluding all the recovery nodes in the levels from 1 to $\ell - 1$.

3.12 Sensor Nodes Deployment

In this section, the sensor density to be deployed in a given area is considered. The sensor nodes group that tracks the target consists of n_g sensor nodes in each time step. Therefore, to increase the likelihood of sensing the target by n_g sensor nodes at all time steps, the number of sensor nodes deployed in the sensing area should be calculated correctly. Since the sensor nodes are assumed to be uniformly deployed over the sensing area (A) with sensor nodes density of ρ *sensors* $/ m^2$, the number of sensor nodes in any given area (A_0) is a Poisson process [138] with mean $\mu = A_0\rho$. This is shown in Figure 28.

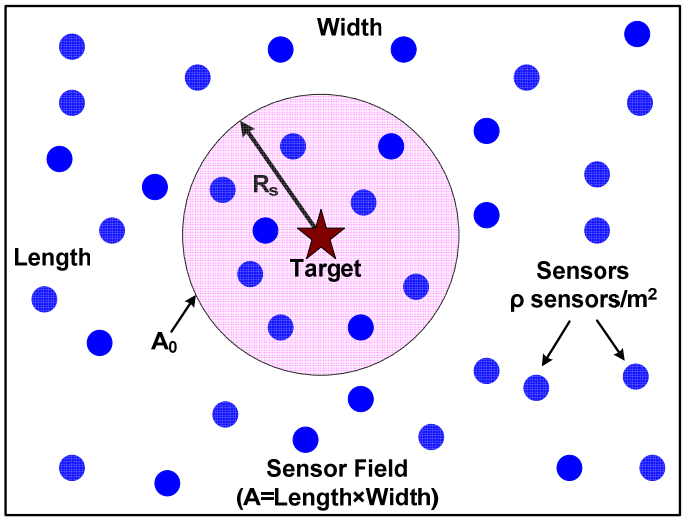


Figure 28 Node Deployment

The probability that the target is within the sensing range of n_g sensor nodes is calculated as follows:

$$p(n_g, \mu) = \frac{e^{-\mu} \mu^{n_g}}{n_g!} \tag{3.49}$$

where, $\mu = A_0\rho = \rho\pi R_s^2$. Therefore, the probability that the target is within the sensing range of n_g sensor nodes or more is calculated as:

$$\Pr(n_g) = 1 - e^{-\rho\pi R_s^2} \times \sum_{i=0}^{i=n_g} \frac{(\rho\pi R_s^2)^i}{i!} \tag{3.50}$$

For any given probability $\Pr(n_g)$, using trial and error the sensor density of ρ can be computed. For example, for $\Pr(n_g) = 1$, $R_s = 50\,m$ and $n_g = 3$ *nodes*, the value of $\rho = 5.6 \times 10^{-3}\, sensors/m^2$ can guarantee the given probability. Thus, for a given sensing area of $A = 300 \times 300 m^2$, the number of sensor nodes required to be uniformly deployed is $m = \rho A \approx 500$ *nodes*.

3.13 Complete Single Target Tacking Algorithms

Figure 29 shows the algorithm running in the HNs. The group MN proactively sends the target metadata to the next group along with the group election results using a group-triggering message (*GTrig*) so that the new group has knowledge of the target before it arrives in their vicinity. The HNs measure the target ranges and send the data to the MN using target range (*TRan*) messages.

```
1. while (true) do:
2.   switch (event) {
3.   Event 1: Receive GTrig or TRec message
4.     Turn on the communication channel (i.e., communication mode);
5.     Turn on the sensing circuits (i.e., sensing mode);
6.     Set Timer_awake;
7.   Event 2: Target Detection
8.     Get the target range measurement;
9.     Shutdown sensing circuit;
10.    Send TRang message to the current MN;
11.    Shutdown the communication channel;
12. Event 3: Timer_awake timeout
13.    Shutdown sensing circuit and the communication channel;
14. }
15. end while
```

Figure 29 Algorithm Running in the Helper Node

Figure 30 shows the algorithm running in the MN. If the current MN does not receive the target range measurement from all HNs after *Timer_recovery*, a target loss (*TLos*) message is sent from the current node to the previous one to inform it about the loss of the target. The previous MN performs the recovery in levels and informs the recovery

nodes to capture the target through target recovery message (*TRec*). The algorithms shown in Figure 29 and 30 are self-explanatory and should be easy to follow.

```
1. while (true) do:
2.  switch (event) {
3.  Event 1: Receive GTrig message
4.   Turn on the communication channel and the sensing circuits;
5.   Set Timer_recovery;
6.  Event 2: Target Detection
7.   Get the target range measurement and shutdown sensing circuit;
8.  Event 3: (Receive TRang from all HNs) or (at least from three nodes in case of recovery)
9.   if (Target is recovering) do:
10.     Localize the target using triangulation and initialize its covariance matrix;
11.     Set the sampling interval to its minimum value;
12.  else do:
13.     Use EKF to get the target updated state and its covariance matrix;
14.     Calculate the sampling interval;
15.     Use EKF to calculate the predicted target state and its covariance matrix;
16.     Update the target location metadata;
17.     Select the next group from the neighbours and perform the election of the next MN;
18.     Send GTrig message to the next group;
19.     Send the current target information to the sink;
20.  Event 4: Timer_recovery timeout
21.   if (All target range measurements are not received) do:
22.     Send Tlost to the previous MN ;
23.     Shutdown sensing circuit and the communication channel;
24.  Event 5: Receive Tlost
25.   Determine the first-level recovery nodes and send them TRec;
26.   Set Timer_levels;
27.  Event 6: Timer_levels timeout
28.   if (Target is still lost) do:
29.     Increment the recovery level, determine the recovery nodes and send them TRec;
30.     Set Timer_levels;
31.   else do:
32.     Shutdown sensing circuit and the communication channel;
33.  }
34. end while
```

Figure 30 Algorithm Running in the Main Node

3.14 Biologically Inspired and Self-Organizing Aspects

Although, other works, which are explored in Chapter 2, have been inspired by biological and ecological principles, none of these assume the differentiation or specialisation of sensor nodes functionalities that is inspired from the biological zygote or human embryonic stem cell. When the zygote or embryo is formed, it comprises a collection of similar cells. All the cells of the zygote are equal in terms of behaviours and capabilities. Over time, the zygote cells start to specialize with different functionalities. The embryo begins with about 150 cells. These cells divide into three layers that are internal, middle and outer layers. Each layer develops in an independent fashion. The internal layer or the endoderm specializes to form the respiratory and digestive system. It also forms the glands such as the pancreas, liver, thymus and thyroid. The middle layer or the mesoderm forms the bones and cartilage, muscles, excretory system, the circulatory system (i.e., heart and blood vessels), the inner skin layer (i.e., dermis), loins and genitalia, and the outer covering of the internal organs. The outer layer or ectoderm becomes the brain, nervous system and epidermis (e.g., skin, hair, nails). This biological behaviour is called differentiation or specialization [139].

The same principle is applied in the proposed MS-ASTT scheme; the sensor nodes start equally and then exhibit some kind of specialisation in order to perform the target tracking. The sensor nodes before the selection and election algorithms were all equal. The selection algorithm differentiates the jobs of the sensors nodes so that some of them will be selected to sense the target and others will remain in sleeping mode. The election algorithm classifies the selected group nodes into one MN and possibly multiple HN(s). Furthermore, this is the first research to treat the target as a virtual chemical emitter.

The proposed MS-ASTT scheme is self-configured and self-organizing. A recovery mechanism is designed to solve the problem of the tracking failures. Prediction is provided to determine the target's future location and prepare the tasking nodes before the target arrives in their vicinity. Load balancing is adopted in the election of the leader of the group of the tasking nodes to track the target. The sensors nodes are at all times aware of their remaining resources. The proposed MS-ASTT scheme in this research operates automatically without interaction with external administration. Therefore, the complexity is hidden from the users.

3.15 Chapter Summary

In this chapter, a MS-ASTT scheme for reliable target tracking in WSNs is presented. The operation of the MS-ASTT scheme can be summarized in four steps. Firstly, the sampling interval is computed according to the historical location metadata of the target such that the prediction is likely to succeed and the tracking accuracy is maintained. Secondly, the next tracking group is proactively selected. An adaptive group size configuration mechanism is presented to improve the tracking accuracy. Thirdly, one of the group nodes is elected as a MN so that the communication energy efficiency and load balancing are improved. Finally, target recovery is supported to improve tracking reliability in the case of target loss due to selection or prediction failures.

A node deployment strategy is introduced to guarantee the coverage of the target with at least predefined number of sensor nodes. The complete algorithms for the proposed MS-ASTT scheme are presented. The proposed MS-ASTT is self-configuring and self-organizing, and inspired from the differentiation or specialisation principles found in biological zygotes.

This chapter considers only the tracking of a single target. In Chapter 4 multi-target tracking (MTT) in WSNs is considered.

Chapter 4 Multi Target Tracking in WSNs

4.1 Chapter Introduction

Chapter 3 introduces the STT in WSNs. In this chapter, Multi-Target Tracking (MTT) in WSNs is presented. The target dynamics, sensor detection and measurement models, and EKF proposed in Chapter 3 are used in this chapter with slight modifications to make them suitable for MTT. In this chapter, two MTT schemes for WSNs are proposed. Firstly, a Multi-Sensor Distributed Multi-Target Tracking (MS-DMTT) scheme is proposed based on the assumption that the sensor node can only detect and serve a single target at the same time. Secondly, a Multi-Sensor Adaptive Multi-Target Tracking (MS-AMTT) scheme is introduced based on the assumption that the sensor node can detect and serve more than one target at the same time. After the description of both schemes in detail, a chapter summary is provided.

4.2 Target Dynamic Model

In this chapter, MTT is considered. The target (T_j) state vector at tracking time $t_{T_j}(k)$ consists of the target coordinates and velocities in xy plane and is written as follows:

$$\mathbf{X}_{T_j}(k)=[x_{T_j}(k) \quad \dot{x}_{T_j}(k) \quad y_{T_j}(k) \quad \dot{y}_{T_j}(k)]' \tag{4.1}$$

The target location at time $t_{T_j}(k)$ and the sensor node (s_i) location can be expressed as:

$$\mathbf{L}_{T_j}(k)=[x_{T_j}(k) \quad y_{T_j}(k)]' \tag{4.2}$$

$$\mathbf{L}_{s_i}=[x_{s_i} \quad y_{s_i}]' \tag{4.3}$$

The target (T_j) dynamics is modelled using the discrete-time white noise acceleration model [72][74][75]. Therefore, the system model described in Equation (2.4) at time $t_{T_j}(k)$ can be written as:

$$\mathbf{X}_{T_j}(k+1)=\mathbf{A}_{T_j}(k)\mathbf{X}_{T_j}(k)+\mathbf{w}_{T_j}(k) \tag{4.4}$$

where $\mathbf{A}_{T_j}(k)=\begin{bmatrix}\mathbf{V}_{T_j}(k) & \mathbf{Z}\\ \mathbf{Z} & \mathbf{V}_{T_j}(k)\end{bmatrix}$, $\mathbf{V}_{T_j}(k)=\begin{bmatrix}1 & \Delta t_{T_j}(k)\\ 0 & 1\end{bmatrix}$, $\mathbf{Z}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$ and $\mathbf{w}_{T_j}(k)$ is the process noise which models the target velocity variations (i.e., acceleration) and is assumed to possess a zero-mean White Gaussian Distribution with $\mathbf{Q}_{T_j}(k)$ covariance matrix:

$$\mathbf{Q}_{T_j}(k)=q_{T_j}\begin{bmatrix}\mathbf{\Lambda}_{T_j}(k) & \mathbf{Z}\\ \mathbf{Z} & \mathbf{\Lambda}_{T_j}(k)\end{bmatrix} \tag{4.5}$$

where, $\mathbf{\Lambda}_{T_j}(k)=\begin{bmatrix}\Delta t_{T_j}^3(k)/3 & \Delta t_{T_j}^2(k)/2\\ \Delta t_{T_j}^2(k)/2 & \Delta t_{T_j}(k)\end{bmatrix}$, q_{T_j} is scalar that represents the amount of randomness in the process noise and $\Delta t_{T_j}(k)=t_{T_j}(k+1)-t_{T_j}(k)$ is the sampling interval, which is the time between the two tracking snapshots (i.e., at times $t_{T_j}(k)$ and $t_{T_j}(k+1)$) and is calculated at time $t_{T_j}(k)$.

4.3 Sensor Detection and Measurement Model

As mentioned in Section 2.6.1, passive sensor devices are used to detect the acoustic signals produced from the targets. Therefore, the target T_j is assumed to be an isotropic sound source. Recall from Section 2.6.4, the RSSI method is used to model the acoustic signal. The acoustic power intensity received by sensor node s_i at time $t_{T_j}(k+1)$ is calculated according to the following model [58][129][130]:

$$P_{s_i}[k+1,T_j]=\frac{S_{T_j}(k+1)}{R_{s_i}^n[k+1,T_j]} \tag{4.6}$$

where $S_{T_j}(k+1)$ is the emitted acoustic density from the sound source (i.e., the target T_j) at time $t_{T_j}(k+1)$ which is assumed to be known , $R_{s_i}^n[k+1,T_j]$ is the noisy geometric distance between the sensor node s_i and the target T_j at time $t_{T_j}(k+1)$ and n is the attenuation decay factor which is typically between 2 to 5 according to the environment and atmospheric conditions [131]. Therefore, by measuring $P_{s_i}[k+1,T_j]$, $R_{s_i}^n[k+1,T_j]$ can be calculated using Equation (4.6). Using a group $S_{g(T_j)}(k+1)=\{s_1,s_2,...,s_{n_{g(T_j)}(k+1)}\}$ of $n_{g(T_j)}(k+1)$ tasking sensor nodes to track the target T_j at time $t_{T_j}(k+1)$ and according to Equation (2.5), the measurement model at time $t_{T_j}(k+1)$ is given by:

$$\mathbf{z}_{T_j}(k+1) = \mathbf{h}_{T_j}[k+1, \mathbf{X}_{T_j}(k+1)] + \mathbf{v}_{T_j}(k+1) \tag{4.7}$$

where $\mathbf{v}_{T_j}(k+1)$ is the measurement noise which is assumed a zero-mean White Gaussian Distribution with $\mathbf{R}_{T_j}(k+1)$ covariance matrix, and the target noisy measurement vector and measurement function are given by:

$$\mathbf{z}_{T_j}(k+1) = \left[R_{s_1}[k+1, T_j] \quad R_{s_2}[k+1, T_j] \quad \ldots \quad R_{s_{n_{g(T_j)}(k+1)}}[k+1, T_j] \right]' \tag{4.8}$$

$$\mathbf{h}_{T_j}[k+1, \mathbf{X}_{T_j}(k+1)] = \begin{bmatrix} \sqrt{[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_1}]'[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_1}]} \\ \sqrt{[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_2}]'[[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_2}]} \\ . \\ . \\ \sqrt{[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_{n_{g(T_j)}(k+1))}}]'[\mathbf{L}_{T_j}(k+1) - \mathbf{L}_{s_{n_{g(T_j)}(k+1)}}]} \end{bmatrix} \tag{4.9}$$

The process and measurement noises are independent of time and with respect to each other. The measurement noise variances of the sensor nodes are assumed to be independent and are assumed to be multiplicative noises [95]. Thus, the measurement noise variance associated with target T_j of sensor s_i at time $t_{T_j}(k+1)$ is calculated as:

$$\sigma_{s_i}^2[k+1, T_j] = \gamma R_{s_i}[k+1, T_j] \tag{4.10}$$

where $0 \le \gamma \le 1$. Therefore, the measurement noise covariance matrix is defined as:

$$\mathbf{R}_{T_j}(k+1) = diag\left[\sigma_{s_1}^2[k+1, T_j], \sigma_{s_2}^2[k+1, T_j], \ldots, \sigma_{s_{n_{g(T_j)}(k+1)}}^2[k+1, T_j] \right] \tag{4.11}$$

4.4 Extended Kalman Filter for Multi-Target Tracking

In the Extended Kalman Filter (EKF) [72][74][75], the predicted target state is given by:

$$\hat{\mathbf{X}}_{T_j}(k+1 \mid k) = \mathbf{A}_{T_j}(k)\hat{\mathbf{X}}_{T_j}(k \mid k)) \tag{4.12}$$

with associated predicted covariance matrix given by:

$$\mathbf{P}_{T_j}(k+1 \mid k) = \mathbf{A}_{T_j}(k)\mathbf{P}_{T_j}(k \mid k)\mathbf{A}'_{T_j}(k) + \mathbf{Q}_{T_j}(k) \tag{4.13}$$

The predicted measurement vector is calculated as follows:

$$\hat{\mathbf{z}}_{T_j}(k+1 \mid k) = \mathbf{h}_{T_j}[k+1, \hat{\mathbf{X}}_{T_j}(k+1 \mid k)] \tag{4.14}$$

The Jacobian matrix of $\mathbf{h}_{T_j}$ at $\mathbf{X}_{T_j}(k+1) = \hat{\mathbf{X}}_{T_j}(k+1|k)$ is:

$$\begin{aligned}\mathbf{H}_{T_j}(k+1) &= \frac{\partial \mathbf{h}_{T_j}[k+1, \mathbf{X}_{T_j}(k+1)]}{\partial \mathbf{X}_{T_j}(k+1)}, \text{ at } \mathbf{X}_{T_j}(k+1) = \hat{\mathbf{X}}_{T_j}(k+1|k) \\ &= [\mathrm{H_{ij}}] \quad 1 \le i \le n_{g(T_j)}(k+1), \quad 1 \le j \le 4\end{aligned} \tag{4.15}$$

where

$$\mathrm{H_{ij}} = \begin{cases} \dfrac{\hat{x}_{T_j}(k+1|k) - x_{\mathrm{s_i}}}{\sqrt{[\hat{\mathbf{L}}_{T_j}(k+1|k) - \mathbf{L}_{s_i}]'[\hat{\mathbf{L}}_{T_j}(k+1|k) - \mathbf{L}_{s_i}]}} & \mathrm{j} = 1 \\ 0 & \mathrm{j} = 2 \\ \dfrac{\hat{y}_{T_j}(k+1|k) - y_{\mathrm{s_i}}}{\sqrt{[\hat{\mathbf{L}}_{T_j}(k+1|k) - \mathbf{L}_{s_i}]'[\hat{\mathbf{L}}_{T_j}(k+1|k) - \mathbf{L}_{s_i}]}} & \mathrm{j} = 3 \\ 0 & \mathrm{j} = 4 \end{cases} \tag{4.16}$$

and $\hat{\mathbf{L}}_{T_j}(k+1|k)$ is the target predicted location and is calculated as:

$$\hat{\mathbf{L}}_{T_j}(k+1|k) = [\hat{x}_{T_j}(k+1|k) \quad \hat{y}_{T_j}(k+1|k)] \tag{4.17}$$

In the update stage where the measurements at time step $k+1$ are available, the measurement residual which is the difference between the actual and predicted measurements can be calculated as follows:

$$\mathbf{r}_{T_j}(k+1) = \mathbf{z}_{T_j}(k+1) - \hat{\mathbf{z}}_{T_j}(k+1|k) \tag{4.18}$$

with associated residual or innovation covariance matrix:

$$\mathbf{S}_{T_j}(k+1) = \mathbf{R}_{T_j}(k+1) + \mathbf{H}_{T_j}(k+1)\mathbf{P}_{T_j}(k+1|k)\mathbf{H}'_{T_j}(k+1) \tag{4.19}$$

The update state estimate is given by:

$$\hat{\mathbf{X}}_{T_j}(k+1|k+1) = \hat{\mathbf{X}}_{T_j}(k+1|k) + \mathbf{K}_{T_j}(k+1)\mathbf{r}_{T_j}(k+1) \tag{4.20}$$

with associated updated covariance matrix:

$$\mathbf{P}_{T_j}(k+1|k+1) = \mathbf{P}_{T_j}(k+1|k) - \mathbf{K}_{T_j}(k+1)\mathbf{S}_{T_j}(k+1)\mathbf{K}'_{T_j}(k+1) \tag{4.21}$$

where the filter gain is defined as:

$$\mathbf{K}_{T_j}(k+1) = \mathbf{P}_{T_j}(k+1|k)\mathbf{H}'_{T_j}(k+1)\mathbf{S}^{-1}_{T_j}(k+1) \tag{4.22}$$

4.5 Multi-Sensor Distributed Multi-Target Tracking (MS-DMTT)

In this section, a MS-DMTT scheme is proposed based on the assumption that a given sensor node can only track and serve a single target at a time. The MS-DMTT scheme deals with each target separately. This means that the MS-DMTT scheme is a series of STT problems. Therefore, each target has separate tracking time steps that are independent of others. However, the problem of conflict nodes, which arises only in case of MTT, is considered and formalized at the beginning of this section. Next the framework and assumptions associated with the MS-DMTT scheme are explained. The sampling interval selection, sensor nodes selection, sensor node election and recovery mechanisms and algorithms are presented. Finally, the Distributed Multi-target Selection (DMS) algorithm that solves the conflict nodes problem is introduced.

4.5.1 Problem Formalization

Figure 31 shows a MTT scenario for WSNs. Two targets are assumed to be tracked. At a particular time the target locations are shown in Figure 31. Each target is cooperatively tracked and served using a group of sensor nodes. Unlike the MTT schemes proposed in Chapter 2, this thesis considers the problem of conflict node selection, which is shown in Figure 31. R_s is the sensing range. All sensor nodes in the conflict area can detect and serve both Target 1 and Target 2. However, each sensor node is assumed to be able to track and serve only one target at a time [140][141]. Therefore, each conflict node has to locally decide its preferred target that it will track.

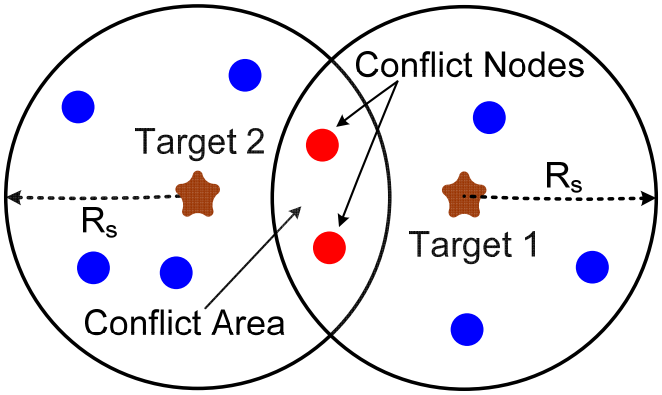


Figure 31 Conflict Nodes in Multi-Target Tracking

A sensor node can detect a new target whilst serving another target. If a sensor node prefers the new target, it will leave the group of the old target. Therefore, the group of the old target has to reconfigure itself to collect one more node to replace the departing node.

4.5.2 MS-DMTT Framework and Assumptions

Figure 32 shows the framework of the proposed MS-DMTT scheme. The same framework as used in the MS-ASTT scheme (presented in Section 3.6) is employed. For simplification, measurement origin uncertainty [103] and the false alarms from the sensor field are not considered further here.

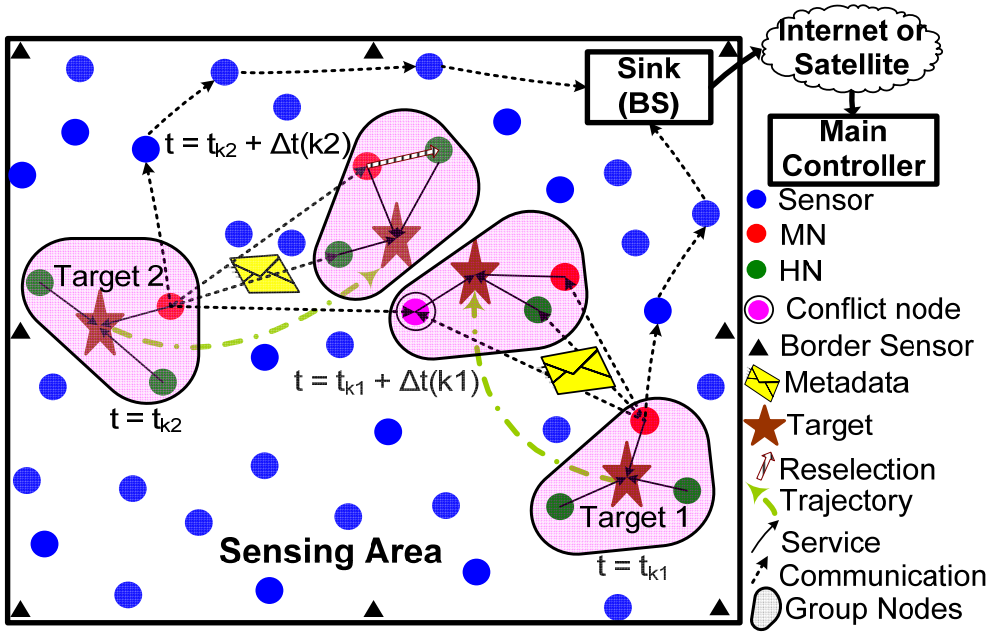


Figure 32 MS-DMTT Framework in WSNs

At each time step k, one member of the group that is formed to track the target T_j is elected to be the Main Node, $MN_{T_j}(k)$ and the other members are called Helper Nodes $HNs_{T_j}(k)$. The main goals of the proposed MS-DMTT scheme are to proactively select next groups to track the targets, elect one sensor node from each group to be the MN_{T_j} of that group, calculate the next sampling intervals for the targets, reform the group in case of the conflict node problem and perform recovery in the case of target loss so that tracking continuity is maintained. This is to be done such that the network lifetime, energy efficiency and tracking accuracy are improved. The tracking initialization is started when mobile targets enter the sensing area. The border sensors sense the targets, localize them using triangulation as presented in Section 2.6.4, and set the initial error covariance. Thus, the border sensors predict the targets' next states using EKF, select the next groups, perform the election of the next MN_{T_j}, initialise the sampling intervals to their minimum value and trigger the next groups to wakeup. If any node s_i receives more than one request to track targets, it performs the Distributed Multi-Target

Selection (DMS) algorithm to decide its preferred target. For example, in Figure 32, the conflict node receives the strongest target influence from Target 1. Therefore it decides to serve Target 1, which is in more important (i.e., higher priority).

4.5.3 Sampling Interval Selection, Sensors Selection, Sensors Election and Recovery Mechanism

The sampling interval selection, sensor nodes selection, sensor node election and recovery mechanism techniques and algorithms used for STT in Chapter 3 are adopted with the MS-DMTT scheme. However, these techniques and algorithms are performed for each target separately based on the target trajectory, movement patterns, importance and sensor nodes locations and resources.

4.5.4 Distributed Multi-Target Selection (DMS) Algorithm

In this thesis, the target importance or priority of the target, is considered. Sensor measurement noise is less when the sensor is closer to the target [95]. Therefore, the target chemical diffusion strength, G (i.e., influence strength on the sensor node) is inversely proportional with distance, D from the node and is higher for the targets of a higher importance. The chemical diffusion strengths for the targets versus the distance from the sensor node are plotted in Figure 33. As shown in Figure 33, Target 1 has a higher priority than Target 2. Thus, the chemical diffusion strength of Target 1 is higher than the case of Target 2.

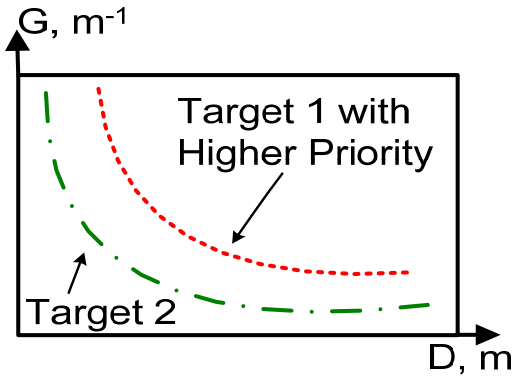


Figure 33 Chemical Diffusion Strength

The conflict node (s_c) that is located in the sensing areas of the targets set (S_T) at the same time decides locally their preferred target by running the DMS algorithm given in Figure 34. In the algorithm, $D(T_j, s_c)$ in line 1 is the Mahalanobis distance between the T_j predicted location PDF and conflict node (s_c). Both target importance and the distance from the target to the sensor node are considered in the calculation of target

influence. Therefore, if two targets share the same importance, the conflict node will select the closer one. On the other hand, conflict node will select the target with more importance if the two targets are the same distance from it. Otherwise, the ratio between the target importance and distance is computed for target selection. The algorithm shown in Figure 34 should be self-explanatory and easy to follow.

1. For all $T_j \in S_T$, calculate $G(T_j, s_c) = \frac{Z_{T_j}(k+1)}{D(T_j, s_c)}$;

2. Select the target (T_s) with maximum $G(T_j, s_c)$;

3. If more targets have the same vale of maximum $G(T_j, s_c)$:

4. Select the target (T_s) with maximum $Z_{T_j}(k+1)$;

5. If s_c is already a member of group $S_{g(T_c)}$ for a target (T_c) and $T_c \neq T_s$:

6.if (s_c is a MN of $S_{g(T_c)}$) do:

7. Reform the $S_{g(T_c)}$ by selecting one more node and electing a new MN;

8. Handover the MN responsibilities to the new MN;

9. Abort $S_{g(T_c)}$ group;

10. else do: // s_c is HN

11. Abort $S_{g(T_c)}$ group and inform the MN to reform the group $S_{g(T_c)}$;

12. end if of line 5;

13. Join the group $S_{g(T_s)}$ of the selected target T_s ;

Figure 34 The DMS Algorithm

4.6 Multi-Sensor Adaptive Multi-Target Tracking (MS-AMTT)

In this section, a MS-AMTT scheme is proposed based on the assumption that the sensor node can detect and serve more than one target at the same time. MS-AMTT scheme deals with all targets at the same tracking time steps. Therefore, all targets have the same tracking time steps. Additionally, sampling interval for all target are the same and fixed. At the beginning of this section, the problem of sensor nodes selection is formalized as a combinatorial optimization problem. Then, The framework and assumptions of MS-AMTT scheme is explained. The main functionalities for the group members that track the target are introduced. After that, the target importance calculation technique is present. The local search strategy to get near-optimal solution for the sensor nodes selection is presented. The computational complexity of the

proposed MS-AMTT scheme is then proposed. Finally, the election technique to select the MN for each tracking group and the leader node for all groups is introduced.

4.6.1 Problem Formalization

At each time step k, the main aim of MS-AMTT is to select the next sensor groups that will track the targets at the next time step $k+1$. Assume $S_{d(T_j)}(k+1|k)$ is defined as the predicted detecting nodes of target T_j with size $n_{d(T_j)}(k+1|k)$ at time step $k+1$. It is calculated at time step k and contains all the predicted sensors that may detect the targets at time step $k+1$. It considers the sensors inside the circle:

$$[\mathbf{X}-\hat{\mathbf{L}}_{T_j}(k+1|k)][\mathbf{X}-\hat{\mathbf{L}}_{T_j}(k+1|k)]' = R_s^2 \tag{4.23}$$

where R_s is the sensing range of the sensor nodes, $\hat{\mathbf{L}}_{T_j}(k+1|k)$ is defined in Equation (4.17) and $\mathbf{X} = [x \quad y]'$. Assume, $S_D(k+1|k)$ with size of $n_D(k+1|k)$ is a set of all the predicted detecting nodes of targets and is calculated as:

$$n_D(k+1|k) = \sum_{\ell=1}^{M} n_{d(T_\ell)}(k+1|k) \tag{4.24}$$

$$S_D(k+1|k) = S_{d(T_1)}(k+1|k) \cup S_{d(T_2)}(k+1|k) \ldots\ldots \cup S_{d(T_M)}(k+1|k) \tag{4.25}$$

where M is the number of the targets. Define $S_{g(T_j)}(k+1|k)$ with size of $n_{g(T_j)}(k+1|k)$ as the predicted sensors group of target T_j. It is calculated at time step k and contains the predicted group nodes to track the target T_j at time step $k+1$. Assume $S_G(k+1|k) \subset S_D(k+1|k)$ with size of $n_G(k+1|k)$ is a set of all targets sensor groups and is calculated as:

$$n_G(k+1|k) = \sum_{\ell=1}^{M} n_{g(T_\ell)}(k+1|k) \tag{4.26}$$

$$S_G(k+1|k) = S_{g(T_1)}(k+1|k) \cup S_{g(T_2)}(k+1|k) \ldots\ldots \cup S_{g(T_M)}(k+1|k) \tag{4.27}$$

The closest sensor nodes selection is not always the best solution for target tracking because of triangulation and co-linearity in [95]. Therefore, the objective function for sensor groups' selection is to minimize the overall updated errors of all targets. The

updated error of a target is defined as the "trace" of its updated covariance matrix. At time step k, the objective function is defined as:

$$F_{obj}(k+1|k) = \sum_{\ell=1}^{M} \{ \; trace\left(\mathbf{P}_{T_\ell}(k+1|k+1)\right) + Z_{T_\ell}(k) \max\left(0, trace\left(\mathbf{P}_{T_\ell}(k+1|k+1)\right) - E_{th}\right) \; \} \tag{4.28}$$

where, $Z_{T_\ell}(k)$ is the importance of target T_ℓ at time step k and E_{th} is a predefined tracking error threshold. Using Equation (4.21), $\mathbf{P}_{T_\ell}(k+1|k+1)$ can be calculated at time step k for a sensor group without knowledge of the sensor measurements at time step $k+1$. The optimal sensors groups $S_G^*(k+1|k)$ that will track the targets at time $k+1$ is selected as:

$$S_G^*(k+1|k) = \arg \min_{S_G(k+1|k)} F_{obj}(k+1|k) \tag{4.29}$$

subject to:

$$n_G(k+1|k) = 3M \tag{4.30}$$

$$n_{g(T_j)}(k+1|k) \geq 1 \qquad \forall 1 \leq j \leq M \tag{4.31}$$

However, this is NP-hard combinatorial optimization problem [142][143]. The number of combinations that need to be calculated to find the optimal solution for this problem is:

$$\binom{n_D(k+1|k)}{n_G(k+1|k)} = C_{n_G(k+1|k)}^{n_D(k+1|k)} = \frac{n_D(k+1|k)!}{n_G(k+1|k)!(n_D(k+1|k)-n_G(k+1|k))!} \tag{4.32}$$

Therefore, it is infeasible to find a solution to this problem in real-time especially for bigger vales of $n_D(k+1|k)$ and $n_G(k+1|k)$. In this chapter, a local search algorithm is thus used to find near-optimal solution in real-time. More details are provided in Section 4.6.5.

4.6.2 MS-AMTT Framework and Assumptions

Figure 35 shows the framework of the proposed MS-AMTT scheme. The same framework of MS-DMTT scheme, which is presented in Section 4.5.2, is used in MS-AMTT scheme.

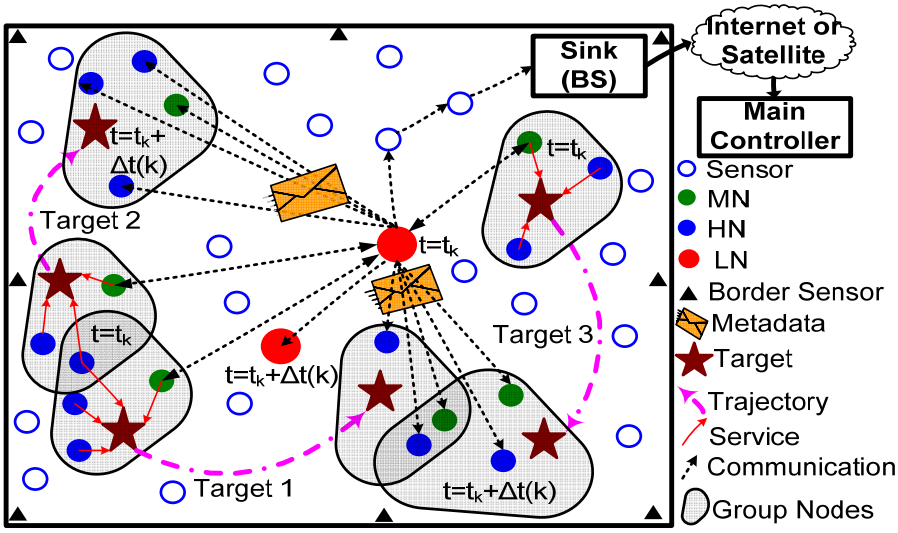


Figure 35 MS-AMTT WSN Framework

The target metadata strategy proposed in Chapter 3 is used in MS-AMTT scheme. The MS-AMTT scheme operates in four steps. Step (1) computes the target importance and number of local search iterations based on the location metadata pertaining to the target's past locations, by which the movement pattern is computed, Step (2) selects the next groups of sensors to track the targets in order to optimize or nearly optimize the tracking performance and continuity. Step (3) elects one sensor from each target group to act as the "Main Node" and select others to be "Helper Nodes". Finally, in Step (4) one node is elected to be the "Leader Node" of all the target groups. MS-AMTT scheme aims to improve the tracking continuity, energy-efficiency and prediction success. At each time step k, the Main Node and Helper Nodes for target T_j are denoted by $MN_{T_j}(k)$ and $HNs_{T_j}(k)$ respectively. Additionally, the leader node is denoted by $LN(k)$. The tracking initialization is started when each target enters the sensing area. The border sensors sense the target, localize it using, for example, triangulation [27], and set the initial covariance error. Thus, the border sensors predict the next target state using EKF, select the next groups of the targets, perform the election of the next MN_{T_j} and LN, and trigger the next groups and LN to wakeup.

4.6.3 The Proposed Algorithms for MS-AMTT scheme

At each time step k, helper nodes, $HN_{T_j}(k)$ measure the target ranges and send the data to the $MN_{T_j}(k)$. The $MN_{T_j}(k)$ calculates the current and predicted target states using EKF, and sends them to the $LN(k)$. The $LN(k)$ performs the management and other

computational duties according to the algorithm shown in Figure 36. Detailed description of this algorithm is provided in Sections 4.6.4, 4.6.5 and 4.6.6.

1. Send the current target states to the sink
2. Compute the targets' location metadata $M_{T_j}(k,K_m)$
3. Calculate the targets' importance $Z_{T_j}(k)$ using the location metadata $M_{T_j}(k,K_m)$
4. Update the target metadata $TMD_{T_j}(k)$
5. Select the next target groups for time step $k+1$ according the Equation (4.29)
6. Perform the election of the next $MN_{T_j}(k+1)$
7. Perform the election of the next $LN(k+1)$
8. Trigger the next groups and leader node for time step $k+1$ to wakeup
9. Send $TMD_{T_j}(k)$ to the next groups and leader node

Figure 36 Algorithm Running in the Leader Node

4.6.4 Adaptive Target Importance

Unlike the MTT schemes proposed in [103] and [104], the target importance is adaptively calculated according to the historical movement pattern of each target in order to obtain seamless and accurate tracking. Figure 37 illustrates the trajectory of a mobile target, T_j. As shown in Figure 37 (a), if the target T_j is in manoeuvring with sharp-bends or random movement, its importance $Z_{T_j}(k)$ will be large. From Equation (4.29), the aim is to find a solution where the objective function is minimized in order to determine which nodes will be assigned to track the various targets. However, if some targets move erratically there is a risk that the overall node mapping solution would assign insufficient nodes to track these targets, whilst more sensor nodes are used to track predictable targets than are needed. To take this into account the second term of the summation in Equation (4.29) is invoked if its tracking error of a particular target exceeds a threshold value. This causes the tracking cost of this target to be increased and so inflate the overall fitness cost, making this solution less desirable. Conversely, if tracking errors remain below this threshold then the additional cost term is not introduced when looking for the least cost solution. However, target importance can be determined offline based on actual priorities of the targets or if the target movement pattern is known in advance. The target movement pattern is modelled using the

location metadata. Location metadata for the target T_j, $M_{T_j}(k, K_m)$ proposed in Chapter 3 is calculated for each target T_j.

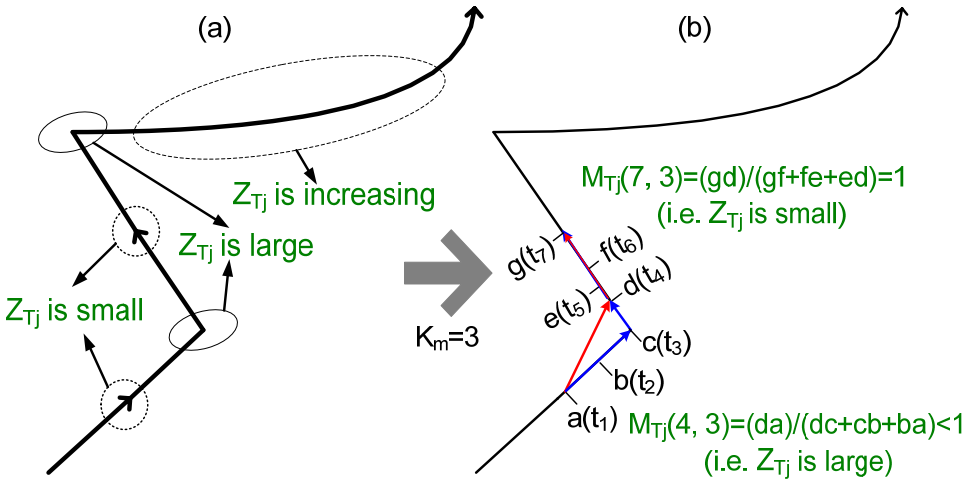


Figure 37 Adaptive Target Importance

In Figure 37 (b), the location metadata at time t_4 and t_7 are calculated using three previous tracking snapshots. If the location metadata at time t_7, $M_{T_j}(7,3)$ is equal to 1 in turn the target importance is set to a small value because the target is moving in a uniform manner. On the other hand, if the location metadata at time t_4, $M_{T_j}(4,3)$ is less than 1 the target importance is set to a large value because the target is moving in a unpredictable manner. As shown in Figure 38, the target importance $Z_{T_j}(k)$ is permitted to adaptively change in the interval $0 \le Z_{T_j}(k) \le Z_m$. $Z_{T_j}(k)$ and is assumed to be a linear function of the location metadata, $Z_{T_j}(k) = f(M_{T_j}(k, K_m))$, according to the following equation:

$$Z_j(k) = Z_m(\ 1 - M_j(k, K_m)\ ) \tag{4.33}$$

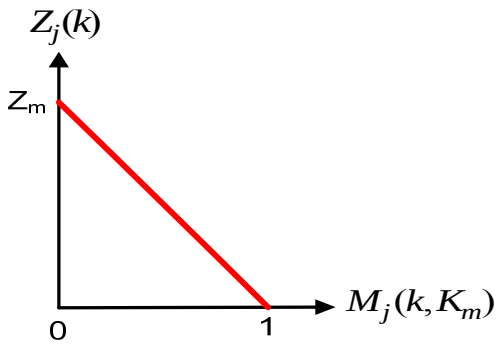


Figure 38 Target Importance as a Function of Location Metadata

4.6.5 Local Search Algorithm

A local search algorithm is used to find near-optimal solution in the real-time for the combinatorial optimization problem [142][144]. It is possible in polynomial time to find a local optimal solution that is the best in the sense that there is nothing better in its local neighbourhood. Given a feasible solution, the set of solutions that are close in some sense to it is called local neighbourhood [142]. The current solution of the local search algorithm starts with an initial solution, which should be chosen carefully as described in Section 4.6.5.1. It then evaluates the objective function for the local neighbourhood around the current solution. Two possible implementations of local search are first improvement (first-fit) and best improvement (best-fit) [142]. In first-fit, the current solution is set to the first neighbour that has better objective function. In best-fit, the objective function has to be computed for the overall neighbourhood and the current solution is set to the neighbour which has the best objective function. The local search is terminated if there is no solution in the neighbourhood that has a better objective function than the current solution.

In fact, the choice of the initial solution and local neighbourhood structure is crucial because it influences the efficiency and computational time of the local search algorithm [145]. In the following sections, a detailed description concerning the initial solution, neighbourhood structure and local search heuristic are presented.

4.6.5.1 Initial Solution Selection

At each time step k, assume the initial solution for the groups tracking the targets at time step $k+1$ is $I_s(k+1|k)$ with size of $n_G(k+1|k)$ nodes. $I_s(k+1|k)$ is divided into equal M subsets (one for each target) such that:

$$I_s(k+1|k) = \left\{ I_{s(T_1)}(k+1|k), I_{s(T_2)}(k+1|k), ..., I_{s(T_M)}(k+1|k) \right\} \tag{4.34}$$

Each subset of $I_s(k+1|k)$ consists of three sensor nodes such that:

$$I_{s(T_j)}(k+1|k) = \left\{ i_{s(T_j)}(1), i_{s(T_j)}(2), i_{s(T_j)}(3) \right\} \qquad \forall j = 1,2,..,M \tag{4.35}$$

Each subset defined in Equation (4.35) is calculated as follows. The biologically inspired target model proposed in Chapter 3 is used. For each target T_j, the Mahalanobis distance $D_{T_j}(k+1|k, s_i)$ [136] which considers the predicted target location covariance $\mathbf{P}_{T_j}(k+1|k)$ in its calculations is obtained between the target's

predicted location PDF and each of predicted detecting nodes $s_i \in S_{d(j)}(k+1|k)$ as follows:

$$D_{T_j}(k+1|k,s_i)=\sqrt{[\mathbf{L}_{s_i}-\hat{\mathbf{L}}_{T_j}(k+1|k)]'\boldsymbol{\Sigma}_{T_j}^{-1}(k+1|k)[\mathbf{L}_{s_i}-\hat{\mathbf{L}}_{T_j}(k+1|k)]} \tag{4.36}$$

$\boldsymbol{\Sigma}_{T_j}(k+1|k)$ is the predicted target location covariance matrix and is calculate by the same method proposed in Chapter 3. The target chemical diffusion strength $G_{T_j}(k+1|k,s_i)$ is defined as:

$$G_{T_j}(k+1|k,s_i)=\frac{1}{D_{T_j}(k+1|k,\mathrm{s_i})} \tag{4.37}$$

Therefore, the selection fitness function by which the sensor nodes will be selected is computed as follows:

$$f_{S(T_j)}[k+1|k,s_i]=\frac{G_{T_j}(k+1|k,s_i)}{\sum_{\forall \ell \in S_{d(T_j)}(k+1|k)} G_{T_j}(k+1|k,s_\ell)} \tag{4.38}$$

The ith sensor node, $i_{s(T_j)}(i) \in S_{d(T_j)}(k+1|k) \quad \forall 1 \le i \le 3$ is selected so that:

$$i_{s(T_j)}(i)=\arg\max_{s_i}\{\ f_{S(T_j)}[k+1|k,s_i]:\ s_i \in S_{d(T_j)}(k+1|k)\setminus \overset{i-1}{\underset{q=1}{U}} i_{s(T_j)}(q)\ \&\ \overset{M}{\underset{\substack{\ell=1\\ \ell\neq j}}{U}} I_{s(T_\ell)}(k+1|k)\ \} \tag{4.39}$$

where $S_{d(T_j)}(k+1|k)\setminus \overset{i-1}{\underset{q=1}{U}} i_{s(T_j)}(q)\ \&\ \overset{M}{\underset{\substack{\ell=1\\ \ell\neq j}}{U}} I_{s(T_\ell)}$ is the set of $S_{d(T_j)}(k+1|k)$ members excluding the ones from $i_{s(T_j)}(1)$ to $i_{s(T_j)}(i-1)$ and the ones that already selected for other targets.

4.6.5.2 Neighbourhood Structure

At each time step k, assume the current solution for the groups tracking the targets at time step $k+1$ is $C_s(k+1|k)$ with size of $n_G(k+1|k)$ sensor nodes. The remaining set $S_r(k+1|k)$ with size $n_r(k+1|k)$ contains the sensor nodes in $S_D(k+1|k)$ excluding the ones in $C_s(k+1|k)$. Therefore, $S_r(k+1|k)$ and $n_r(k+1|k)$ are calculated as:

$$S_r(k+1|k)=S_D(k+1|k)\setminus C_s(k+1|k) \tag{4.40}$$

$$n_r(k+1|k)=n_D(k+1|k)-n_G(k+1|k) \tag{4.41}$$

Each sensor in $S_r(k+1|k)$ replaces one by one the $n_G(k+1|k)$ sensors in $C_s(k+1|k)$ to form $n_G(k+1|k)$ neighbours. Therefore, the neighbour sets, $N_s(k+1|k)$ of the current solution, $C_s(k+1|k)$ and its size, $n_s(k+1|k)$ are calculated as:

$$n_s(k+1|k) = n_G(k+1|k)\{n_D(k+1|k) - n_G(k+1|k)\} \tag{4.42}$$

$$N_s(k+1|k) = \{N_{s(1)}(k+1|k), N_{s(2)}(k+1|k), \ldots\ldots N_{s(n_s(k+1|k))}(k+1|k)\} \tag{4.43}$$

The neighbours are generated according to the following algorithm shown in Figure 39. Therefore, we have a maximum of $n_s(k+1|k)$ neighbours for the current solution $C_s(k+1|k)$. However, the neighbours will be generated one by one during the search to reduce the computational complexity.

1. Compute the objective function for each node $s_i \in S_r(k+1|k)$ when it is used alone to detect all targets.
2. Sort the $S_r(k+1|k)$ nodes in non-increasing order based on the number of targets detected by sensors in $S_r(k+1|k)$.
3. Sort the $S_r(k+1|k)$ resulted from Step 2 in non-decreasing order based objective function for sensors in $S_r(k+1|k)$.
4. Set the neighbour number ℓ to 1.
5. **for** each sensor $s_i \in S_r(k+1|k)$ **do:**
6. **for** each sensor $s_j \in C_s(k+1|k)$ **do:**
7. $N_{s(\ell)}(k+1|k) = C_s(k+1|k)$;
8. In $N_{s(\ell)}(k+1|k)$, replace s_j by s_i;
9. **if** $N_{s(\ell)}(k+1|k)$ detect all targets by at least one node **do:**
10. Store $N_{s(\ell)}$ in neighbours structure ($N_s(k+1|k)$);
11. Increment ℓ;
12. **end inner for;**
13. **end outer for;**

Figure 39 neighbourhood Structure

4.6.5.3 Complete Local Search Heuristic Algorithm

A First-fit local search approach is used in this thesis. The complete search algorithm is summarized as follows:

Step (1) Compute the initial solution $I_s(k+1|k)$, set the current solution $C_s(k+1|k) = I_s(k+1|k)$ and set $iter = 0$.

Step (2) Calculate the objective function $F_{obj}(C_s) = F_{obj}(k+1|k)$ for $C_s(k+1|k)$ and set the neighbour number $\ell = 1$.

Step (3) Compute the neighbour $N_{s(\ell)}(k+1|k)$.

Step (4) If $iter$ > maximum allowable iterations (I_A), go to Step 7.

Step (5) Calculate the objective function $F_{obj}(N_{s(\ell)}) = F_{obj}(k+1|k)$ of $N_{s(\ell)}(k+1|k)$.

Step (6) If $F_{obj}(N_{s(\ell)}) < F_{obj}(C_s)$, $iter = iter + 1$, set $C_s = N_{s(\ell)}$ and go to Step (2).

Else

- $\ell = \ell + 1$.
- if $\ell > n_s(k+1|k)$ go to Step 7.
- else go to Step 3.

Step (7) Return $C_s(k+1|k)$ as the result of the local search and finish.

4.6.5.4 Computational Complexity

In the neighbourhood structure discussed in Section 4.6.5.2, one sensor node is changed from the current solution to get the new neighbours. Therefore, the complexity for the neighbourhood structure is bounded by $O\{n_G(k+1|k)n_D(k+1|k)\}$. In the local search algorithm described is Section 4.6.5.3, the worst case is to search the entire neighbourhood structure. Therefore, the complexity is bounded by $O\{n_G(k+1|k)n_D(k+1|k)\}$. However, the maximum allowable iterations (I_A) can dramatically control the computational time of the local search algorithm. Unlike the MTT scheme proposed in [103] and [104], the MS-AMTT scheme controls the number of iterations to reduce the computational time and energy consumption whilst normally maintaining seamless tracking. The computational time increases with increasing I_A. However, using a small I_A can degrade the tracking performance and lose targets especially if the targets move in a random fashion. In this thesis, the maximum allowable iterations (I_A) is adaptively calculated according to the historical location metadata of the targets. Generally speaking I_A is set to a small value if the targets move in uniform manner because the targets can be successfully predicted using EKF and vice versa. The maximum allowable iterations (I_A) is permitted to adaptively change in

the interval $I_{\min} \le I_A \le I_{\max}$. As shown in Figure 40, at each time step k, the maximum allowable iterations (I_A) is calculated based in the following equations:

$$M_{\min} = \min\left\{M_{T_1}(k, K_m), M_{T_2}(k, K_m), ..., M_{T_M}(k, K_m)\right\} \quad (4.44)$$

$$I_A(k) = I_{\max} + (I_{\min} - I_{\max}) M_{\min} \quad (4.45)$$

where $I_{\max}$ and $I_{\min}$ are the maximum and minimum of I_A respectively.

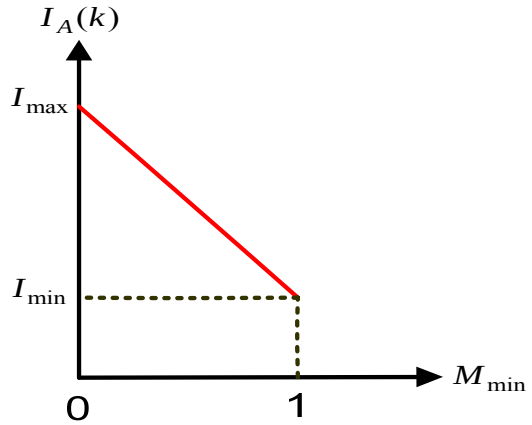


Figure 40 Maximum Allowable Iteration as a Function of Minimum Location Metadata

The computational complexity of the location metadata calculation is $O(MK_m)$. However, the target's importance and maximum allowable iterations are calculated once for the current solution and its neighbours at each time step.

4.6.6 Main and Leader Node Election

The group is classified into one MN and a number of HN(s). One more node is selected to be the LN of the groups. The MN and LN typically perform more processing and communication activities. Hence, choosing the MN and LN is a crucial issue to improve the energy saving and network lifetime. The election mechanism proposed in Chapter 3 is used to elect the MN and LN. At time step k, the group main node, $MN_{T_j}(k+1|k)$ at the next time step $k+1$ of the target T_j is elected from the selected group of nodes for the target, $S_{g(T_j)}(k+1|k)$. The leader node, $LN(k+1|k)$ of the target groups is elected from the sensor nodes inside the area that is surrounded by group MNs. The election algorithm details are provided in Chapter 3.

4.7 Biologically Inspired and Self-Organized Aspects

Like the MS-ASTT scheme, the principle of biological differentiation is applied in the MS-DMTT and MS-AMTT schemes; the sensor nodes start equally and then exhibit

some kind of specialisation in order to perform target tracking. The sensor nodes before the selection and election algorithms are all equal. The selection algorithm differentiates the jobs of the sensors nodes so that some of them will be selected to sense the target and others will remain in their sleeping mode. The election algorithm classifies the selected group of nodes into one MN, HN(s) and an LN. Furthermore, this is the first research to treat the target as a virtual chemical emitter.

The proposed MS-DMTT and MS-AMTT schemes are self-configured and self-organizing without the need for external administration. Therefore, their complexity is hidden from the users. Furthermore, a recovery mechanism is designed to solve the problem of tracking failures. Prediction is provided to anticipate the target future location and prepare the tasking nodes before the target arrives in their vicinity. Load balancing is adopted in the election of the leader of the group of tasking nodes to track the target. The sensor nodes are all the time aware of their remaining resources.

4.8 Chapter Summary

In this paper, MS-DMTT and MS-AMTT schemes are presented for multi target tracking for WSNs. The MS-DMTT scheme is developed based on the assumption that the sensor node can only detect and serve one target at the same time. On the other hand, the MS-AMTT scheme is developed based on the assumption that the sensor node can detect and serve more than one target at the same time.

In MS-DMTT scheme, at each tracking step, the sampling interval is computed such that the prediction is likely to succeed and the tracking is continuous. The next tracking groups for the targets are then proactively selected and one of the group members is elected as a group MN such that the energy efficiency of the communication and network lifetime are improved. Finally, conflict nodes locally decide the preferred target based on target importance and their distance from the target.

In the MS-AMTT scheme, at each tracking step, a target's importance and maximum allowable iterations are computed. Then, the next tracking groups are proactively selected such that the tracking continuity and accuracy are improved. After this, one of the group members is elected as the MN and another node from the area surrounding the MNs is elected to be the LN so that the communication energy efficiency and network lifetime are improved.

In the Chapter 5 task mapping and scheduling in WSNs is proposed to improve the network lifetime and the execution time of applications that may be running across a group of sensor nodes.

Chapter 5 Task Mapping and Scheduling in WSNs

5.1 Chapter Introduction

Chapter 3 and 4 introduce single and multi target tracking in WSNs, respectively. In this chapter, Task Mapping and Scheduling (TMS) in WSNs are presented. Firstly, a Biological Task Mapping and Scheduling (BTMS) algorithm is proposed. In the BTMS algorithm, the application is assumed to be decomposed into dependent tasks with different computation weights. Secondly, the Biological Independent Task Allocation (BITA) algorithm is introduced. In BITA, the application is assumed to be decomposed into equal-weighted independent tasks. Finally, a chapter summary is provided.

5.2 Biological Task Mapping and Scheduling (BTMS) Algorithm

BTMS is a TMS algorithm in which an application is executed by a group of sensor nodes in parallel. Tracking algorithms are one of the attractive applications that can employ BTMS especially as these algorithms are computational intensive and require real time execution [6]. At the beginning of this section, a model for a general high-level application is considered. After that the problem of TMS in WSNs is formulated. Then, the BTMS algorithm is presented. To increase the network lifetime, decision-making rules are then introduced.

5.2.1 Application Model

The application is assumed to be decomposed into dependent tasks with different computation weights. A DAG is adopted to provide a general model for the application [120][124][146]. The DAG $A=(V,E)$ consists of a set of vertices V representing the (n) tasks, $V=\{v_i : i=1,2,...,n\}$, and a set of edges E representing the (e) communication dependencies, $E=\{\xi_k : i=1,2,...,e\}$. The edge $\xi_k \in E$ between $v_i \& v_j \in V$ is denoted as e_{ij}, where v_j is called the immediate successor of v_i and v_i is called the immediate predecessor of v_j. Therefore, the task cannot be executed until it receives all the results from its immediate predecessors. This dependency between tasks execution is called the communication dependencies constraint. As shown in Figure 41, a task without immediate predecessors is an entry-task or a source-task while a task without immediate successors is an exit-task or a sink-task. In WSNs, the entry-tasks are used for sensing or gathering the raw data to detect physical phenomena. Therefore, task placement

constraints can be defined as only one source task can be assigned to the sensor node. In Figure 41, v_1 and v_2 are source-tasks, v_8 is the sink-task, v_3 and v_4 are the immediate predecessors of v_6, and v_7 is the immediate successor of v_5 and v_4. The task v_3 cannot be executed until it receives the communication edges (i.e., dependencies) e_{13} and e_{23} from tasks v_1 and v_2 respectively. The latency constraint of the application means that the application should be executed before the application deadline, P.

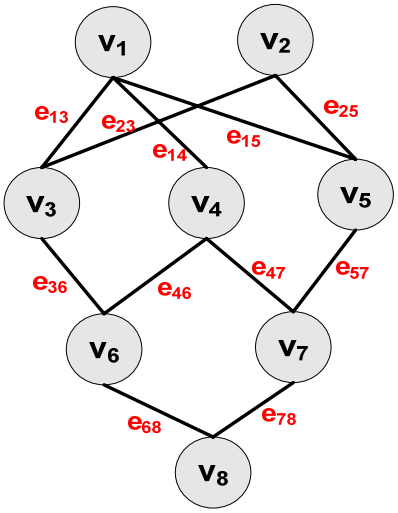


Figure 41 An Example DAG

5.2.2 Problem Formulation

Assume, $S_{net} = \{s_i, i = 1,2,...,m\}$ are the set of sensor nodes for a homogenous WSN so that $p_{s_1} = p_{s_2} = ... = p_{s_m} = f$ where p_{s_i} is the processor speed of sensor node s_i and f is the value of the processor speed for all sensor nodes. During the network operation, assume a sensor node s_T makes a request to its neighbouring sensor nodes to ask them to share in the execution of an application. Assume the set of neighbours that decide to participate the sensor node s_T in the execution of the application are $S_n = \{s_i, i = 1,2,....,n_n\}$. Therefore, the overall set of sensor nodes that can share to execute the application in parallel are $S_m = \{s_i : i = 1,2,..., n_m\} = S_n \cup \{s_T\}$ where $n_m = n_n + 1$. Define $S_g = \{S_g^Z : Z = 1,2,...\}$ as a set of the subsets, $S_g^Z \subset S_m$ where $S_g^Z = \{s_i : i = 1,2,...,n_g^Z\}$ and $n_g^Z \le n_m$. Unlike the CoRAl algorithm [121] which does not consider the energy consumption, the main goal of the BTMS algorithm is to find the set of the sensor nodes, S_g^{opt} so that the energy efficiency of executing an application is optimized or nearly optimized without violating the latency constraint of the application. Therefore, unlike the EcoMapS algorithm [126], the BTMS algorithm can guarantee the execution

of the application before the application's deadline. The main objective of BTMS is to find $S_g^{opt} \subset S_g$ where $S_g^{opt} = \{s_i : i = 1,2,...,n_g^{opt}\}$ and $n_g^{opt} \le n_m$ so that the total energy consumption using S_g^{opt} , $energy\,(S_g^{opt})$ is minimized subject to meet the application's deadline. Mathematically, this optimization problem can be formulated as:

$$S_g^{opt} = \arg\min_{S_g^Z} energy\,(S_g^Z) \tag{5.1}$$

$$energy\,(S_g^Z) = \sum_{S_g^Z} E_{comm} + \sum_{S_g^Z} E_{comp} \tag{5.2}$$

Subject to:

$$CET \le P \tag{5.3}$$

where CET is the Collaborative Execution Time of the application, and E_{comm} and E_{comp} are the total energy consumption of the data communication and computation required to execute the application using S_g^{opt} sensor nodes, respectively . In the WSN target-tracking schemes proposed in Chapter 3 and 4, s_T can be the MN and the set S_n consists of the HN(s). In fact, the task mapping and scheduling problem is an NP-complete problem [7]. Therefore, the BTMS algorithm, which is greedy heuristic algorithm, is proposed to obtain near-optimal solution.

5.2.3 BTMS Algorithm

In the DAG $A=(V,E)$, each task $v_i \in V$ can be modelled as a tuple of the form: $\{ N_{v_i}$, t_{v_i} , $E_{v_i} \}$. N_{v_i} is the number of its computational cycles and t_{v_i} is its execution time where:

$$t_{v_i} = N_{v_i} / f \tag{5.4}$$

E_{v_i} is the computational energy consumption to execute it. Each edge $\xi_k \in E$ (i.e., denoted as e_{ij}) between tasks v_i and v_j can be modelled as a tuple of the form: $\{ b_{e_{ij}}$, $t_{e_{ij}}$, $E_{e_{ij}} \}$. $b_{e_{ij}}$ is the data size of the dependency generated from task v_i and needed to execute task v_j. $t_{e_{ij}}$ is the time required to transmit the $b_{e_{ij}}$ bits from the sensor node on which v_i is mapped to the sensor node on which v_j is mapped. $E_{e_{ij}}$ is the communication energy consumption required to transmit and receive $b_{e_{ij}}$ bits. If v_i and

v_j are mapped in the same sensor node, $t_{e_{ij}}$ and $E_{e_{ij}}$ are set to zero. Otherwise, $t_{e_{ij}}$ is the transmission and propagation times needed to transmit and receive the $b_{e_{ij}}$ bits such that:

$$t_{e_{ij}} = b_{e_{ij}} / B + d_{e_{ij}} / c \tag{5.5}$$

where c is the speed of light, B is transmission speed and $d_{e_{ij}}$ is the distance between sensor nodes where the v_i and v_j are mapped. E_{v_i} and $E_{e_{ij}}$ are calculated using the energy consumption models proposed in Chapter 3. Therefore, unlike the TMS techniques proposed in [122] and [124], BTMS considers the energy consumption for every communication and processing activity. Moreover, the BTMS algorithm differentiates between energy costs at the sender and receiver according to the energy consumption models proposed in Chapter 3. Each sensor node $s_i \in S_{net}$ is a tuple of the form: $\{ NID_{s_i}, E_{s_i}, x_{s_i}, y_{s_i}, E_{th} \}$ where NID_{s_i} is the sensor identification, E_{s_i} is the remaining energy of the sensor node, (x_{s_i}, y_{s_i}) is the 2D location of the sensor node and E_{th} is the threshold energy after which the sensor node cannot participate any processing activity. Figure 42 shows the BTMS algorithm.

In line (1), the level-based DAG is built so that the lowest level contains the entry-tasks and the highest level contains the exit-tasks. The immediate predecessors of the tasks in each level are only located at the lower levels [122]. Figure 43 shows the DAG before and after converts it to level-based DAG.

Unlike, the RT-MapS [127] and MTMS [128] algorithms, in line 2 in each level, the tasks are arranged in non-increasing order so that the large tasks in each level are mapped first. This arrangement leads to less CET because the large tasks will be executed in parallel with small tasks rather than executing small tasks and then waiting until large tasks finish execution.

Figure 44 explains the motivation for this arrangement using a simple scenario. Assume a particular level in the DAG contains four tasks 1, 2, 3 and 4. Assume three sensor nodes share execution of the application. In Figure 44 the height of the rectangle that contains the task number indicates the task size. In Figure 44 (a), the tasks are mapped in order while in Figure 44 (b), the tasks are arranges in non-increasing order before mapping. In Figure 44 (b), task 1, 2 and 3 are executing during execution of task 4. Therefore, the CET with tasks arranged in non-increasing order leads to a smaller CET than without any such arrangement.

1. Convert the DAG into level-based DAG;
2. Sort the task in each level in decreasing order;
3. Select $\lambda \in [0 \quad 1]$;
4. **for** each task $v_i \in V$ from the lowest level **do:**
5. **for** each node $s_j \in S_m$ **do:**
6. Calculate $E_T(s_j, v_i)$;
7. Calculate $t_f(s_j, v_i)$;
8. Calculate $f(s_j, v_i)$;
9. End of inner for loop;
10. Map and schedule the task v_i to the node s_j that has minimum $f(s_j, v_i)$;
11. Update the nodes energy remaining;
12. Do not assign any more tasks to the nodes that has energy less than E_{th} ;
13. **end of outer for loop;**
14. Calculate the execution time (CET);
15. **if** ($CET > P$) **do:**
16. Ignore the current task allocation,
17. Choose different λ ;
18. Go to step 4;
19. **end if;**
20. Finish;

Figure 42 BTMS Algorithm

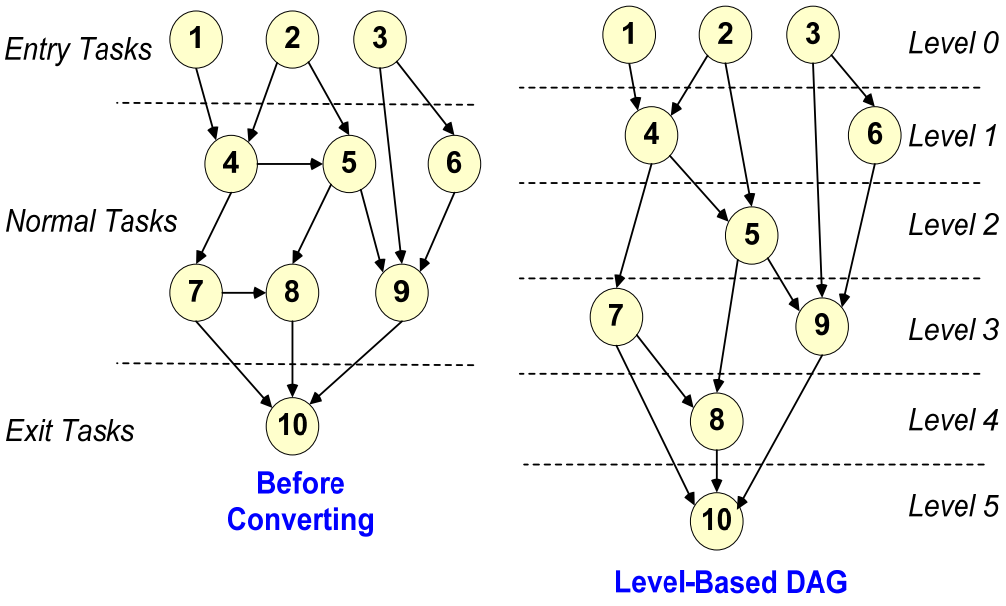


Figure 43 Level-Based DAG

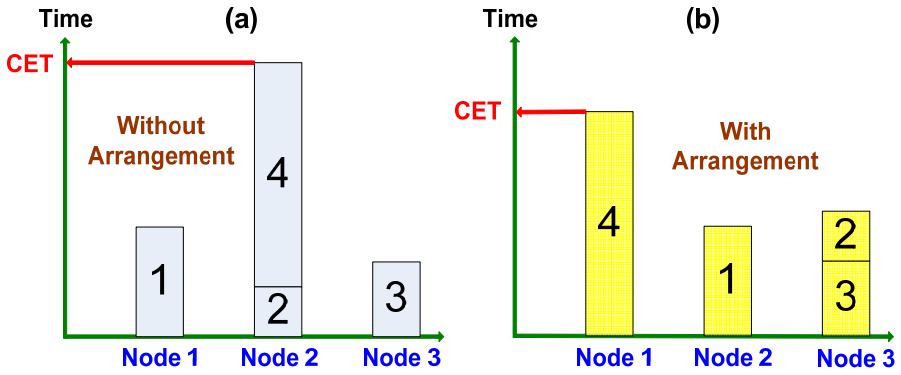


Figure 44 Arrangement the Tasks in Non-increasing Order

In line (6), $E_T(s_j, v_i)$ is the total energy required to execute the task v_i on the sensor node s_j and is calculated as:

$$E_T(s_j, v_i) = E_{v_i} + \sum_{\forall k \in pred\ (v_i)} \left(E_{e_{ki}} + \varepsilon_s\right) \tag{5.6}$$

where, ε_s is the average energy consumption required to manage the communication (i.e., including the collision cost and the channel access management) and $pred\ (v_i)$ is the set of all immediate predecessors indices of v_i. In line (7), $t_f(s_j, v_i)$ is the time required to execute the task v_i on the sensor node s_j and is calculated as:

$$t_f(s_j, v_i) = t_s(s_j, v_i) + t_{v_i} \tag{5.7}$$

where, $t_s(s_j, v_i)$ is the start execution time of the task v_i on the sensor node s_j. It is calculated as follow:

$$t_s(s_j, v_i) = \max\left\{ava(s_j), \max_{\forall k \in pred\ (v_i)} \left(t_{f_{ki}} + t_{e_{ki}} + \tau_s\right)\right\} \tag{5.8}$$

where, $ava(s_j)$ is the availability of the sensor node s_j (i.e., the time at which the sensor node can execute the next task), τ_s is the average time required to manage the communication (i.e., including the collision cost and the channel access management) and $t_{f_{ki}}$ is the completion time of the immediate predecessor (k). In line (8), the fitness function $f(s_j, v_i)$ of executing the task v_i on the sensor node s_j is calculated as:

$$f(s_j, v_i) = \left[\lambda * \frac{t_f(s_j, v_i)}{P}\right] + \left[(1-\lambda) * \frac{E_T(s_j, v_i)}{E_{\max}}\right] \tag{5.9}$$

where, $E_{\max}$ is the maximum energy level of the sensor nodes and $\lambda \in [0\ \ 1]$ is a design parameter which controls the weight of minimizing the total energy consumption and the application execution time, CET.

5.2.4 Decision-Making Algorithm

Unlike the related work of TMS in WSNs proposed in Chapter 2, decision-making rules are defined in each sensor node to increase the network lifetime and connectivity. The decision-making rules shown in Figure 45 allow the sensor node to decide whether it can participate in the processing activities or not. Dead sensor nodes give rise to holes in the network. Therefore, the sensor node can participate in the processing activities if the energy level of a sensor node is above a threshold value and the sensor node has enough neighbours to relay the data in the network (i.e., it is identified as NotOnlyRelayNode). On the other hand, the sensor node will prefer to remain a relay node to forward the data to other sensor nodes if it is located in scarce area or if its remaining energy is under a threshold value. Therefore, the network connectivity and lifetime is improved.

1. **if** (($E_{s_i} > E_{th}$) && (NotOnlyRelayNode)) **do:**
2. Participate the processing activities;
3. **end if;**
4. **else do:**
5. Do not participate the activity;
6. **end else;**

Figure 45 The Decision Making Rules

5.2.5 Computational Complexity Analysis

Recalling Section 5.2.1 and 5.2.2, assume there are n_m sensor nodes and the DAG has n computational tasks. In BTMS algorithm presented in Figure 42, the loop in line 4 is executed in $O(n)$ time and the loop in line 5 is executed in $O(n_m)$. Thus, the computational complexity of BTMS algorithm is $O(nn_m)$. Min-Min technique is the core of the MTMS algorithm [128]. As discussed in Chapter 2, Min-Min technique involves all tasks with all nodes to map a particular task. Thus, its computational complexity [128] is $O(n^2 n_m)$. Therefore, BTMS is less complex than MTMS because BTMS does not involve all tasks to map a particular task.

5.2.6 Biological Inspired Aspects in BTMS Algorithm

Unlike the related work on TMS in WSNs proposed in Chapter 2, BTMS is biological inspired algorithm. The same biological principles presented in Chapter 3 are applied in the proposed BTMS. The network nodes start equally in a default state and then exhibit some kind of differentiation to execute an application. As shown in Section 5.2.3 and 5.3.4, the application tasks are mapped to the sensor nodes according to their resource availability and locations where according to Equation (3.10), the energy consumption for the communication is related to the distance between sensor nodes. Therefore, each node will be specialized to execute different tasks.

5.3 A Biological Independent Task Allocation (BITA) Algorithm

In this section, the BITA algorithm is presented. In BITA, the application is assumed to be decomposed into equal-weighted independent tasks. The equal-weighted of the tasks means that all tasks require the same number of CPU clocks to be executed. A collection of independent tasks is called a meta-task [111][112]. BITA can be used to allocate these equal-weighted independent tasks among a group of sensor nodes according to their available resources and locations. The BITA algorithm is the first task allocation technique used in WSNs to map independent tasks of equal-weighted among a group of sensor nodes.

Assume an application can be decomposed into (N) independent equal-weighted tasks. Like BTMS, during the network operation, assume a sensor node s_T makes a request to its neighbouring sensor nodes to ask them to share in the execution of an application. Assume the set of the neighbours that decide to participate the sensor node s_T in the execution of the application are $S_n = \{s_i, i = 1,2,....,n_n\}$. Therefore, the overall set of sensor nodes that can share to execute the application in parallel are $S_m = \{s_i : i = 1,2,..., n_m\} = S_n \cup \{s_T\}$ where $n_m = n_n + 1$. The tasks are assumed to be submitted for execution from s_T to the $S_n = \{s_i, i = 1,2,....,n_n\}$ nodes. Similarly, the results of task execution are submitted back to s_T. Obviously, the tasks mapped to s_T do not required submission.

Unlike the CoRAl algorithm [121], BITA explicitly considers the sensor node energy resource. Based on energy models proposed in Chapter 3, the data transmission energy consumption is proportional to the square of the distance between the source and the destination. Therefore, the nearer the sensor node ($s_i \in S_m$) to the sensor node (s_T),

the less communication power is required to submit the tasks to s_i and get the results from it in return, the more tasks can be located to s_i. Furthermore, in order to prolong the network lifetime, the node resource availability is considered. The more resource that is available at sensor node $s_i \in S_m$, the more tasks that can be allocated to it.

In this thesis, the sensor node is treated as a virtual chemical emitter that influences other sensor nodes with a varying strength, which is determined according to the sensor node proximity to other sensor nodes. Therefore, the sensor's influence on other sensors nodes is characterised by chemical diffusion strength (G). The chemical diffusion strength of a sensor node decreases with distance. Mathematically, the chemical diffusion strength (G) of sensor node s_i on another sensor node s_j is calculated as follows:

$$G(s_i, s_j) = \begin{cases} \dfrac{1}{\sqrt{[\mathbf{X}_{s_i} - \mathbf{X}_{s_j}][\mathbf{X}_{s_i} - \mathbf{X}_{s_j}]'}} & i \neq j \\ Z \qquad i = j \end{cases} \tag{5.10}$$

where, $\mathbf{X}_{s_i} = [x_{s_i} \quad y_{s_i}]'$ and $\mathbf{X}_{s_j} = [x_{s_j} \quad y_{s_j}]'$ are the 2D locations of sensor nodes s_i and s_j respectively, $\sqrt{[\mathbf{X}_{s_i} - \mathbf{X}_{s_j}][\mathbf{X}_{s_i} - \mathbf{X}_{s_j}]'} = d_{ij}$ is the distance between sensor nodes s_i and s_j, and Z is a real number . The decomposed fitness functions $f_D(s_i, \delta, S_g)$ that indicates the ability of sensor nodes to execute the tasks is calculated for each sensor node $s_i \in S_m$ as follows:

$$f_D(s_i, \delta, S_m) = \left[\beta * \frac{G(s_i, s_T)}{\sum_{\forall k \in S_m} G(s_k, s_T)} \right] + \left[(1 - \beta) * \frac{E_{s_i}}{\sum_{\forall k \in S_m} E_{s_k}} \right] \tag{5.11}$$

where $0 \leq \beta \leq 1$. To improve the load balancing and energy efficiency, each sensor node $s_i \in S_m$ is assigned a number of tasks, $n(s_i, N)$ so that:

$$\frac{n(s_1, N)}{f_D(s_1, \delta, S_m)} = \frac{n(s_2, N)}{f_D(s_2, \delta, S_m)} = \text{.....} = \frac{n(s_{n_m}, N)}{f_D(s_{n_m}, \delta, S_m)} \tag{5.12}$$

$$\sum_{\forall k \in S_m} n(s_k, N) = N \tag{5.13}$$

Therefore,

$$n(s_i, N) = N * \frac{f_D(s_i, \delta, S_m)}{\sum_{\forall k \in S_m} f_D(s_k, \delta, S_m)} \tag{5.14}$$

Therefore, from Equations (5.10) and (5.14), the value of z determines how much the sensor node s_T can participate in the task execution. Larger values of z give larger values of $f_D(s_T, \delta, S_g)$ and $n(s_T, N)$. The chemical diffusion strength of sensor node s_T itself is assumed to be the same as the chemical diffusion strength of s_T on any sensor far away from s_T by $R_r/2$ where R_r is the radio range. Therefore, z in Equation (5.10) is set to $2/R_r$.

The target tracking proposed in Chapter 3 and 4 is one that is assumed to be partially decomposed into independent equal-weighted tasks because it has intensive matrix calculations. Furthermore, a stream equal independent network jobs can be allocated in WSNs using BITA. The sensor node s_T can be the MN and the set S_n consists of the HN(s).

The same biological inspired aspects adopted in the BTMS algorithm are applied in the operation of the BITA algorithm. Each sensor node specializes to execute particular tasks according to its available resources and its location.

5.4 Chapter Summary

In this chapter, TMS in WSNs is presented. If the application can be decomposed into independent tasks, BTMS is used as TMS algorithm. The main aim of BTMS is to reduce the energy consumption and the execution time such that the application deadline is met. Conversely, BITA is used as the TMS algorithm in the case when the application can be divided into independent equal-weighted tasks. The main goal of the BITA algorithm is to reduce the execution time of the application by parallelizing its execution and to increase the network lifetime by adopting load balancing.

In Chapter 6 the simulation environment, data structures and simulation events are introduced in relation to the simulator used to evaluate the proposed target tracking, and task mapping and scheduling techniques proposed in Chapter 3, 4 and 5.

Chapter 6 Simulation Environment

6.1 Chapter Introduction

Chapters 3 and 4 present STT and MTT schemes for WSNs. In Chapter 5, a task mapping and scheduling (TMS) scheme is proposed. In this chapter, the simulation models used to evaluate the proposed target-tracking and TMS are explained. In the beginning of the chapter, event driven simulation is introduced. The main simulator flow chart and the random number generator used are then introduced. The data structures and different event types are discussed. A chapter summary is provided at the end.

6.2 Event Driven Simulation

Most real systems are complex. Hence, analytical solutions become very difficult to obtain. Therefore, simulations are used to evaluate the system numerically. Mainly, two types of simulation exist which are discrete and continuous simulations. System states that describe the system at a particular time are discrete in discrete simulation and continuous in continuous simulation. A discrete event simulation is a simulation in which the system is modelled as it evolves over a time (i.e., dynamic simulation). Therefore, the system state variables change instantaneously at separate points in time (i.e., discrete simulation). As a result, the system can only change at a countable number of points in times, at which the events can occur. Therefore, the event is an instantaneous occurrence that may change the state of the system [147][148]. The proposed target tracking and TMS in this thesis are considered as complex discrete systems. Therefore, a discrete event simulation has been developed to evaluate them.

6.3 Simulation Framework

Figure 46 shows the overall structure of the implemented event-driven simulator. The chart summarizes the steps involved to run one experiment (i.e., trial) of the simulator. The simulator starts by reading the input parameters and constants. Then, the output files are created and initialized to store the results. After that, the cumulative statistical counters are initialized. The simulation initialization routine is called to initialize the simulation clock, build data structures and create the first event. After that, the timing function is called to obtain the next event and advance the simulation clock. Depending

on the next event type, the appropriate event subroutine is called. Extra events are added to the event list based on the current event. The simulation stop condition is checked. If the condition is met, the simulator generates the statistical report and terminates. If the condition is not met, the timing function is called again to obtain the next event and advance the simulation clock. The simulation stop condition varies depending on the simulated scenario.

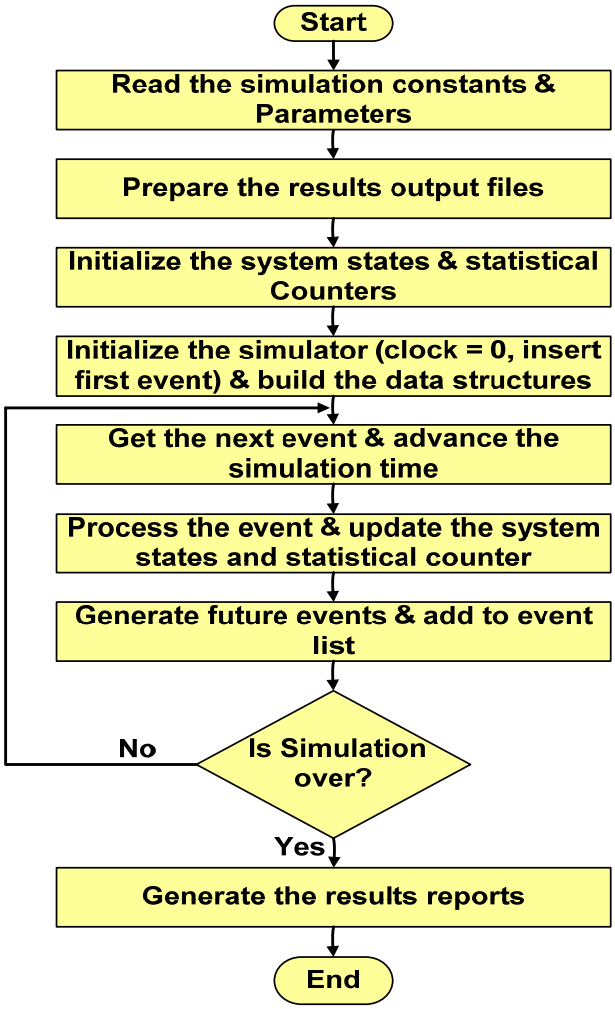


Figure 46 Overall Simulator Structure

6.4 Object Oriented Programming

Although existing simulators could have been used to evaluate the proposed scheme, such as NS2 and SensorSim, the authors chose to implement their own tool to evaluate the proposed schemes in a way that better matched the particular characteristics of the scenarios. This includes the mobility models, MAC layer behaviour, energy consumption model and tracking protocol operation. The concept of adaptive sampling and consideration of the target classes are not implemented in any existing simulators.

Additionally, many physical and MAC layers aspects of these simulators are not required. Therefore, the same programming effort or maybe more would be spent adapting one of the available simulators rather than developing a bespoke solution. C++ [149][150], which is an object-oriented language, has been used to build the simulation models. The class principle and use of libraries in C++ helped to implement large models in a modular fashion.

The Mersenne Twister [151] random number generator has been implemented in a separate class with attributes to set different seeds and functions that implement different property distributions. Objects are declared from the random number generator class as needed. The node is modelled in a class and node objects from this class are used to create the network node data structures. The target attributes are encapsulated in a class and objects of that class are initiated as needed. The messages are also implemented in one class and objects of this class are created when the node needs to send a message.

As shown in Figure 47, the world where the network is deployed is modelled as a 2-dimensional array using a random placement of the sensor nodes. The distance between any two objects $O_i(x_i, y_i)$, $O_j(x_j, y_j)$ in the world can be calculated using Euclidean distance:

$$d_{ij} = \left\| O_i - O_j \right\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{6.1}$$

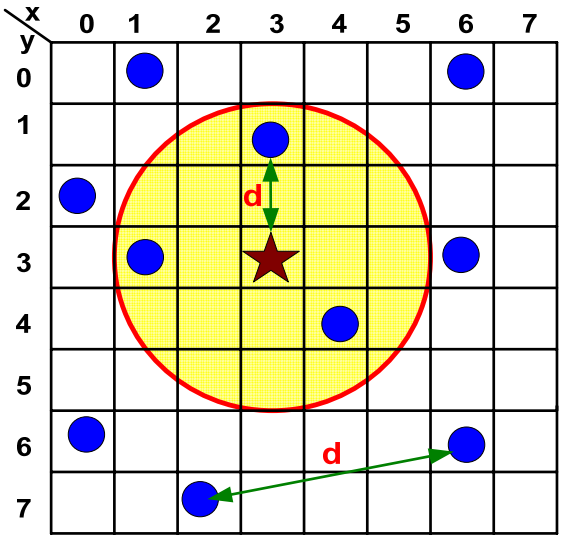


Figure 47 The Simulation World Model

Simulation parameters can be read from an input file or they can be declared as constants. Vectors, lists and arrays are primarily used to store the simulation data

structures. Detailed description about the developed simulators is introduced in details in Appendix A.

6.5 CSMA/CA Event List

In the simulator, the packet exchange is designed based on the CSMA/CA MAC protocol. When the target arrives at the sensing area, the main events that manage the message exchange between nodes are target arrival, ready to send, wait Distributed Inter-Frame Space (DIFS), back off, transmission, reception, wait acknowledgment (i.e., collision), and tick events. Figure 48 shows these in an events graph. Defer and Process shown in the Figure 48 are not events. They are included just to clarify certain event operations. As shown, target arrivals allow the simulation to commence. After that, the nodes that detect the target start to transmit control packets to identify the group and serve the target. The tick event is triggered periodically in the simulator to record some of the simulation results. The simulation is stopped based on predefined conditions. The pseudo code for each event is explained briefly in Appendix A.

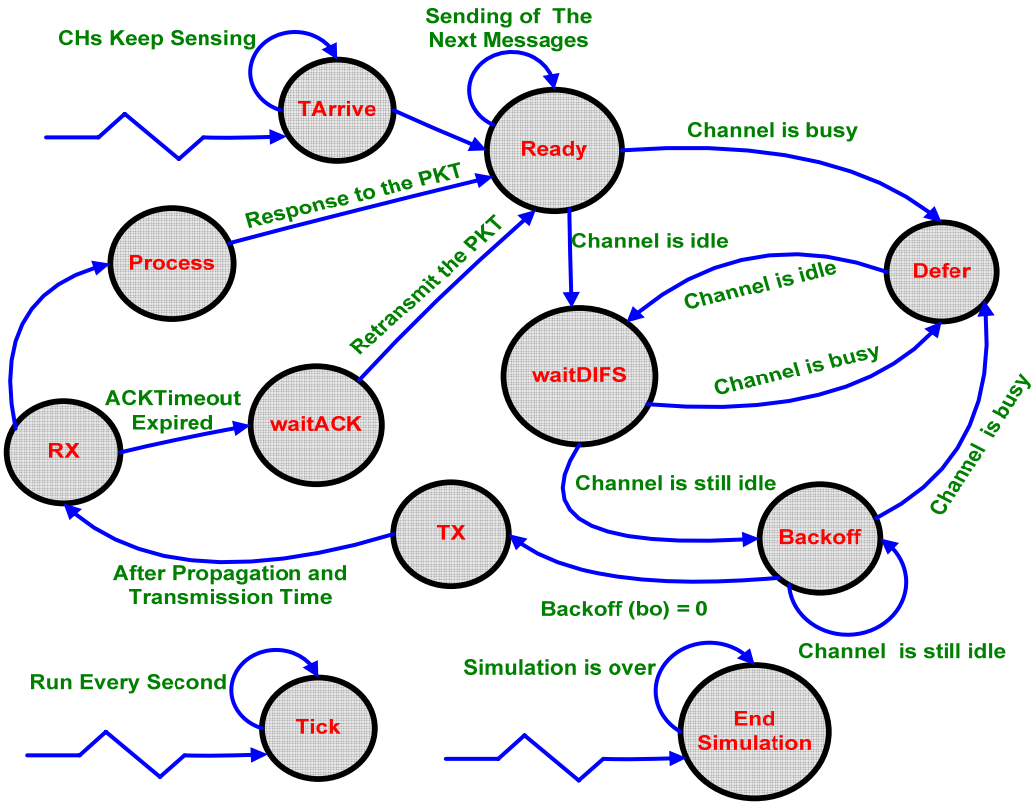


Figure 48 CSMA/CA Event Graph

6.6 Chapter Summary

In this chapter, the simulation environment is explained. The simulator employs Mersenne Twister random number generator. The event graph is presented. Detailed description about the developed simulators is presented in Appendix A. For further information, the pseudo code of the main events is given in Appendix A. The next chapter discusses the simulation results. A critical assessment of the results is also provided.

Chapter 7 Simulation Results

7.1 Chapter Introduction

In this chapter a performance evaluation of the MS-ASTT, MS-DMTT, MS-AMTT, BTMS and BITA schemes that are proposed in Chapter 3, 4 and 5 is given. A critical assessment and discussion of the simulation results is also provided. Additionally, the proposed schemes are compared against well-known approaches. Finally, a summary of the chapter is provided.

7.2 Simulation Assumptions

In this section the assumptions common to all simulations presented in this chapter are introduced. The performance of the proposed schemes presented in Chapter 3, 4 and 5 are evaluated using a C++ simulation environment and 1.73 GHz Pentium IV processor. Unless specifically stated, to improve the statistical significance of the simulation results [137], the results are averaged over 20 runs using different random sensor placements with a fixed density. Line of sight (LOS) communication is assumed between the nodes within the same coverage area. Two nodes are in the same coverage area if the distance between them is equal to or less than the radio range, which is set to 100m. The radio range is set to be twice the sensing range [69]. Therefore, the sensing range is set to 50m. The Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol proposed in Section 2.4 is used as the MAC layer protocol. According to the Bianchi model [152], the parameters of CSMA/CA using Frequency-Hopping Spread Spectrum (FHSS) as a physical layer protocol are listed in Table 1. As in [133], the energy model parameters for Equation (3.10) to Equation (3.12) are set as follows: $\varepsilon_{amp} = \varepsilon_{FS} = 10pJ/b/m^2$, $V_T = 26mV$, $c = 0.5$, $C = 0.67nF$, $I_o = 1.96mA$, $n = 21.26$, $K = 239.28MHz/V$, $E_{elec} = 50nJ/b$, and $f = 100MHz$. If the sensor node detects the target, the sensing energy cost is assumed to be 8×10^{-9} J [5]. In the MS-ASTT scheme, all the sensor nodes are assumed to have identical measurement noise variances, $\sigma_s^2 = 0.001$ [5]. For the MS-DMTT and MS-AMTT schemes, γ in Equation (4.10) is set to 0.001 [5]. The amount of randomness in the process noise is $q_{T_j} = 50$ for all targets [5]. $S(k+1)$ in Equation (3.6) and $S_{T_j}(k+1)$ in Equation (4.6) are set to 40 for all k and all targets [110]. n in Equation (3.6) and Equation (4.6) is set to

2 for all k and all targets [110]. The adaptive sampling interval, α in Equation (3.32) is set to 0.5 so that the weight of measured sampling interval is the same as the previous sampling interval. The adaptive sampling and target importance, K_m in Equation (3.27) is chosen to cover snapshots over the last 2 seconds of the target path. DSDV discussed in Chapter 2, has been implemented in the model as the multi-hop network layer routing protocol.

Parameter	Value
Packet Payload	8184 bits
PHY Header	128 bits
MAC Header	272 bits
ACK	112 bits + PHY Header
RTS	160 bits + PHY Header
CTS	112 bits + PHY Header
Channel bit rate	1Mbps
Propagation delay	1μs
Slot Time	50μs
SIFS	28μs
DIFS	128μs
ACK_Timeout	300μs
CTS_Timeout	300μs
CWmin	255
CWmax	1024

Table 1 CSMA/CA FHSS Parameters

7.3 MS-ASTT Scheme Evaluation

In this section, the MS-ASTT scheme proposed in Chapter 3 is evaluated and compared with other well-known STT schemes. The results are critically assessed at the end of this section.

7.3.1 Simulation Setup

As described in Section 3.12, a uniformly distributed random deployment of 500 wireless sensor nodes across an area of 300m × 300m can guarantee the coverage of the target at any location by at least 3 sensors. For simplicity [5][110], the data processing required to localize the target, predict the target next state, update the target state and recover the target location in case of loss are assumed to be $N = 1$, 2, 2 and 1 MCC (Mega Clock Cycles), respectively. The size of every message used for the operation of the MS-ASTT scheme is assumed to be $l = 288$ bits. The energy consumption to trigger the nodes to wakeup using the low energy communication channel is neglected [134]. The "*Timer_recovery*" and "*Timer_levels*" timers used for recovery are set to 0.05

seconds. Unless specifically stated, an adaptive sampling interval is used according to Equation (3.31) and Equation (3.32) with $T_{\min}=0.1$ sec and $T_{\max}=0.5$ sec [5][110]. The maximum energy level of each sensor node $E_{s_i}^{\max}$ is set to 100J. The group size to track the target $n_g(k)=3$ for all k. The recovery mechanism presented in Chapter 3 is used to recapture lost targets.

7.3.2 Recovery Mechanism Evaluation

In this scenario, a single target travels in straight line for 10 min starting from the position (10, 10). When the target reaches the edge of the sensing area, it randomly changes its direction to keep travelling inside the sensing area. In Equation (3.31), $T_{\max}$ is changed during the simulation over the interval [0.1 0.5] while $T_{\min}$ is kept fixed to a value of 0.1 seconds. Figure 49 shows the real and estimated trajectories of the target after 5 min. The estimated trajectory is close to the true trajectory.

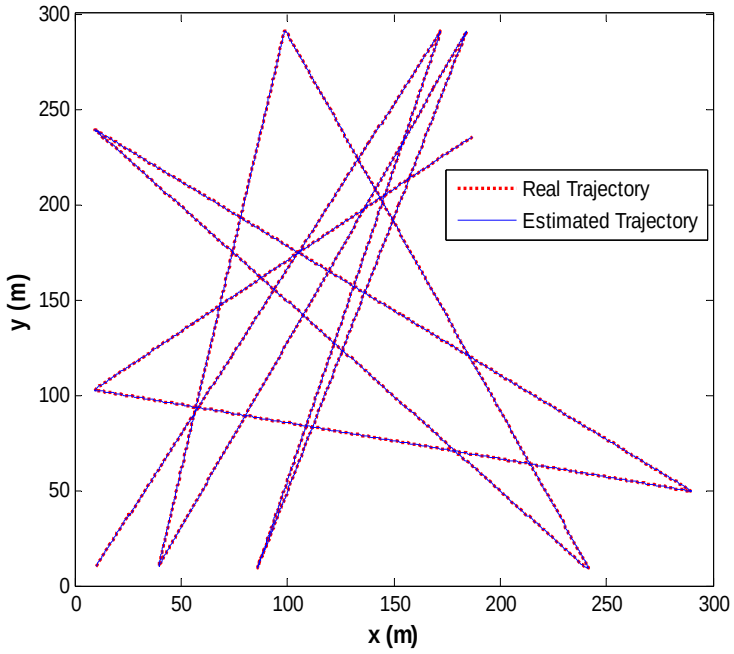


Figure 49 Target Trajectory using $T_{\max}=0.1$ min and Velocity=10m/s

In Figure 50, 51 and 52, the number of recovery events, the total energy consumption and the total recovery time are plotted versus $T_{\max}$ for different velocity values. In Figure 50, the number of recovery events increases with increasing $T_{\max}$ and velocity.

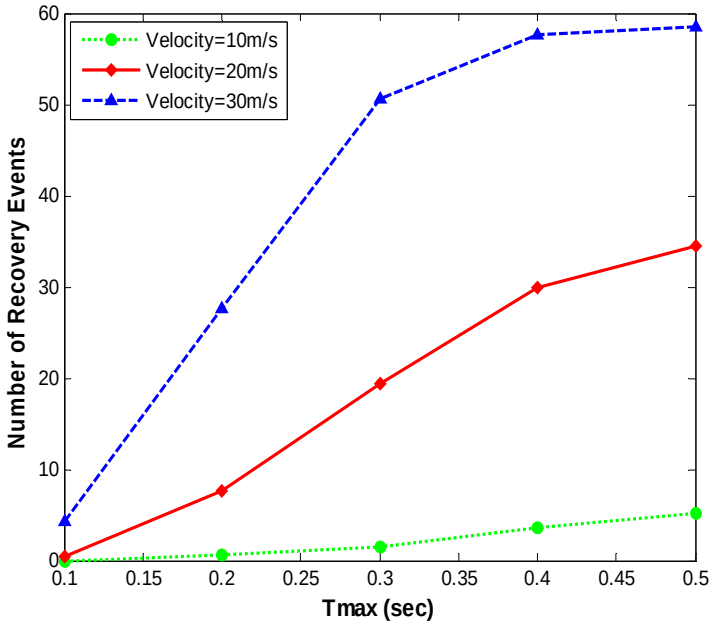


Figure 50 Number of Recovery Events Variations with T_{max}

In Figure 51, the energy consumption decreases dramatically with increasing T_{max} because less tracking snapshots are processed. On the other hand, with increasing velocity, the increase in the energy consumption is small, although the number of recovery events is larger for higher velocities. As shown in Figure 52, the time required for recovery increases with increasing T_{max} and velocity. Finally, first and second level recoveries are found to be sufficient to successfully recover the tracking. Moreover, the percentage of level-1 recoveries during this scenario is 99.38%.

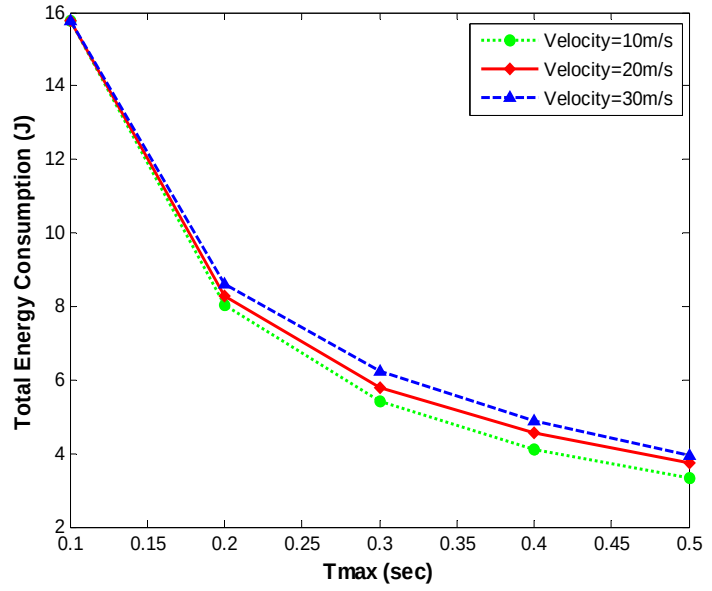


Figure 51 Energy Consumption Variations with T_{max}

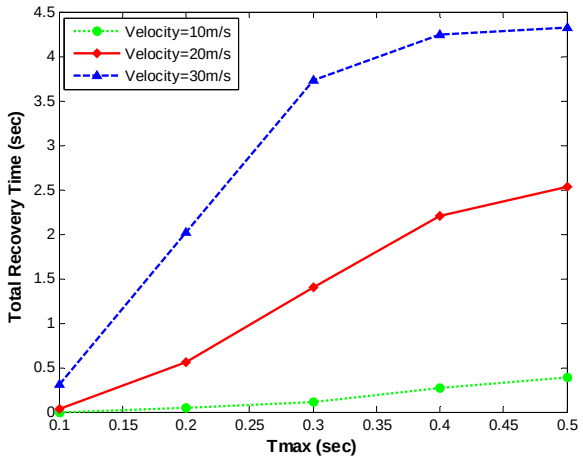


Figure 52 Total Recovery Time Variations with $T_{\max}$

7.3.3 Impact of Adaptive Node Election

The effect of the weighting parameter δ, defined in Equation (3.41) is evaluated in this section. The same scenario as used in Section 7.3.2 is employed with $T_{\min} = 0.1$ sec, $T_{\max} = 0.5$ sec and velocity 10m/s. The simulation is stopped after 600 minutes. The network lifetime is defined as the time at which the first sensor node death occurs. The performance metric that indicates the load balancing performance is defined as follows:

$$P_m = \frac{\sum_{j=1}^{m} E_{s_j} / E_{s_j}^{\max}}{m} \tag{7.1}$$

where m is the number of the sensor nodes in the network (i.e., 500), E_{s_j} is the remaining energy of the sensor node s_j and $E_{s_j}^{\max}$ is the initial energy of the sensor node s_j which is chosen randomly over the interval, $0 < E_{s_i}^{\max} < 1$ Joules. P_m is recorded at the end of each simulation run. Table 2 shows the lifetime, P_m and energy consumption for different δ values. Setting δ to zero improves the load balancing (i.e., P_m) and in turn the network lifetime. On the other hand, choosing a value of unity for δ reduces the energy consumption of the network. Adaptive calculation of δ reduces the energy consumption and improves load balancing.

Approach	Lifetime(min)	P_m	Energy (J)
$\delta = 0$	14.23	0.628	198.964
$\delta = 1$	2.55	0.479	198.848
Adaptive δ	14.04	0.636	198.893

Table 2 Lifetime, Load Balancing Performance and Energy Consumption for Different δ Values

7.3.4 Impact of Group Size

The effect of the group size, $n_g(k)$ that is formed to track the target is evaluated in this section. The same scenario as used in Section 7.3.3 is used. The group size is set to fixed values of 1, 2, 3, 4 and 5. In Figure 53 and 54, the total number of recovery events and time required during the simulation decreases with increasing the group size because the tracking accuracy is improved with increasing the sensor node measurements.

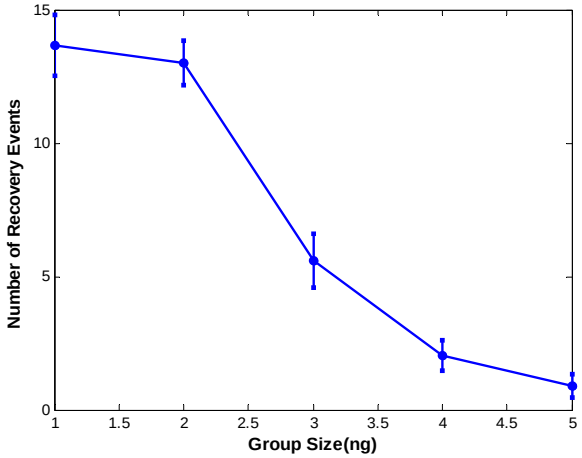


Figure 53 Number of Recovery Events versus Group Size with a 95% Confidence Interval

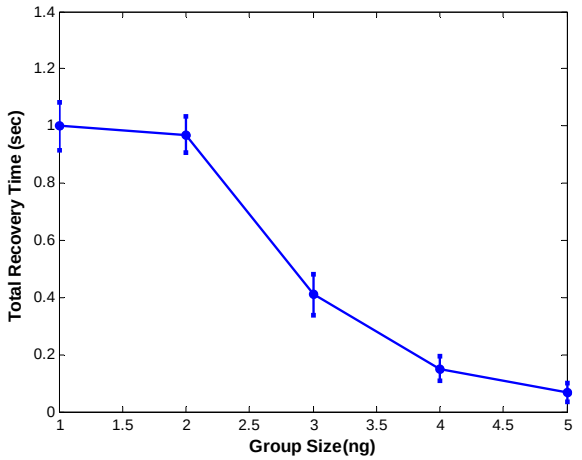


Figure 54 Total Recovery Time versus Group Size with a 95% Confidence Interval

In Figure 55, different overhead messages are plotted. The number of "*TRan*" messages increase with increasing group size. "*GTrig*", "*TLos*" and "*TRec*" messages decrease with increasing the group size because as shown in Figure 53 the number of recovery events increases with decreasing the group size. The number of "*TRec*" messages equals to the number of "*TLos*" messages if only level-1 recovery is needed. In Figure 56, the number of overall overhead increases with group size.

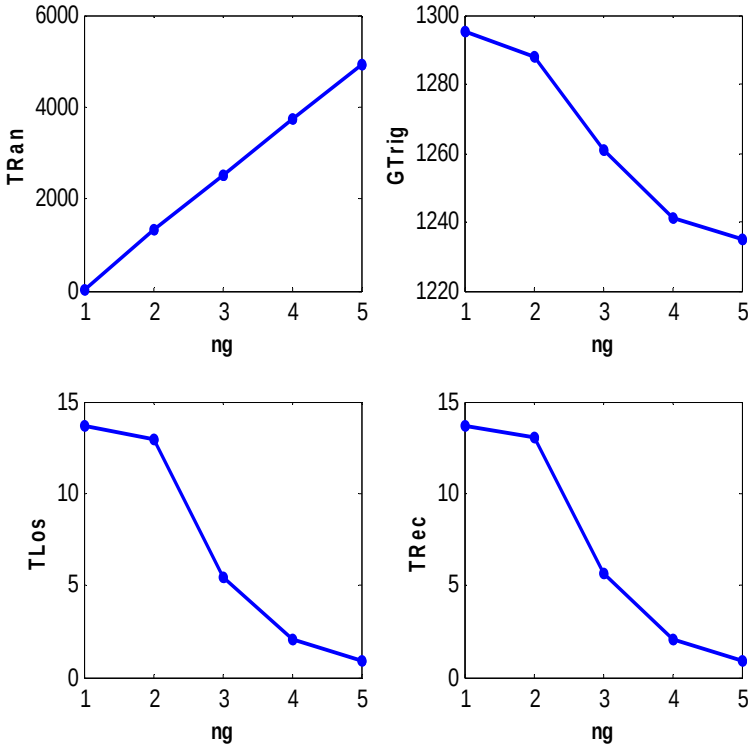


Figure 55 Overhead Message Characteristics

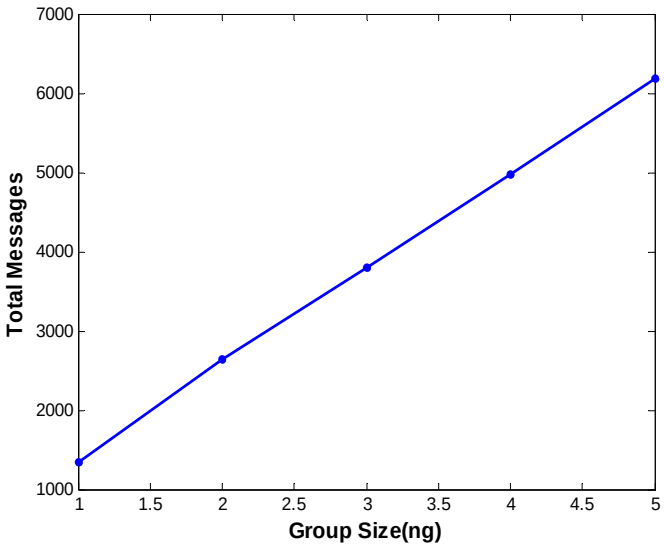


Figure 56 Total Messages versus Group Size

In Figure 57, the overhead time is defined as the time required at each tracking time step to send the "*TRan*" and "*GTrig*" messages. The overhead time increases with the group size. However, the overhead time is always less than the minimum sampling interval (i.e., $T_{\min} = 0.1$ sec).

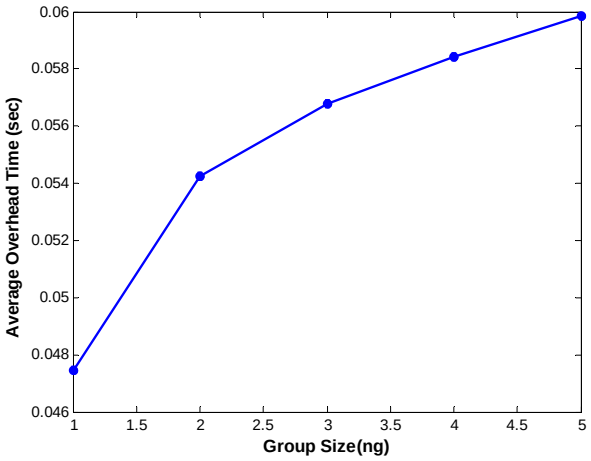


Figure 57 Average Overhead Time versus Group Size

In Figure 58, the energy consumed during the simulation period is plotted. In Figure 53, the number of recovery events with a group size of 2 is less than with a group size of 1 by only about 0.3. Therefore, the energy consumption rises at group size of 2 because as shown in Figure 56 more messages are required for the larger group size. However, it then reduces at a group size of 3 because as shown in Figure 53, the number of recovery events at group size of 3 is less than the group size of 2 by about 8. After that, it starts to rise again at group size of 4 and 5 as more messages are required for larger group sizes, whilst there is little impact on the number of recovery events.

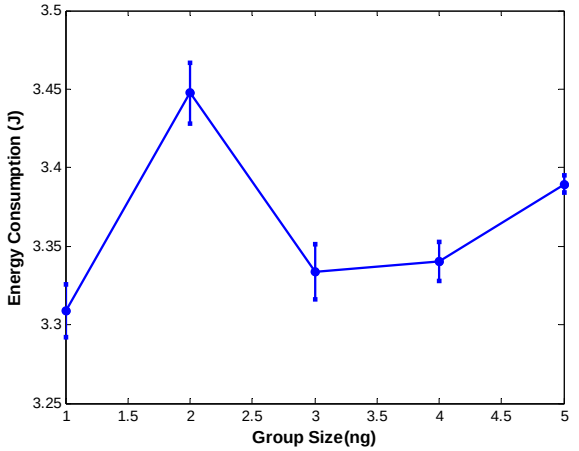


Figure 58 Energy Consumption versus Group Size with a 95% Confidence Interval

In Figure 59, the total number of retransmissions at the simulation end increases with increasing the group size because as shown in Figure 55 more ”*TRan*“ messages are sent with increasing group size. This increases the possibility that sensor nodes will access the channel at the same time and in turn increases the possibility of the collisions.

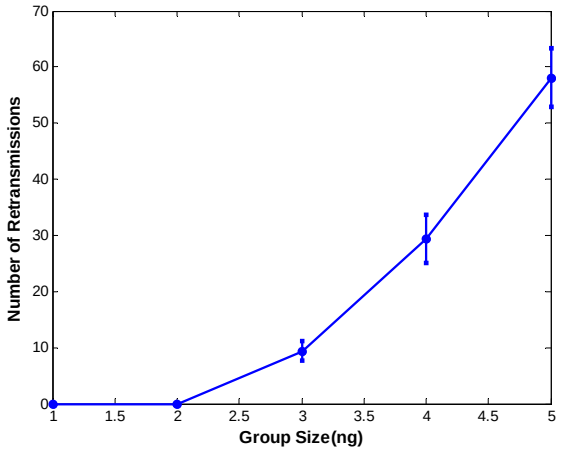


Figure 59 Number of Retransmissions versus Group Size with a 95% Confidence Interval

7.3.5 Comparison with other STT Schemes Using Fixed Trajectory

In this section, a single target travels for 2 minutes with constant speed of 10m/s and importance, $Z_T = 1$, along the path shown in Figure 60. The border sensors initiate the tracking process with initial target state $[0 \;\; 5 \;\; 100 \;\; 5]'$ and initial covariance matrix $10\mathbf{I}$ [110], where $\mathbf{I}$ is the identity matrix. In this section, the tracking update error is defined as the "trace" of the updated state covariance matrix, $\mathbf{P}(k \mid k)$ while the tracking prediction error is defined as the "trace" of the predicted target location covariance matrix ($\mathbf{\Sigma}_T(k+1 \mid k)$). In Figure 60, the true and estimated target trajectories for the proposed MS-ASTT scheme are plotted. The estimated trajectory is close to the true trajectory.

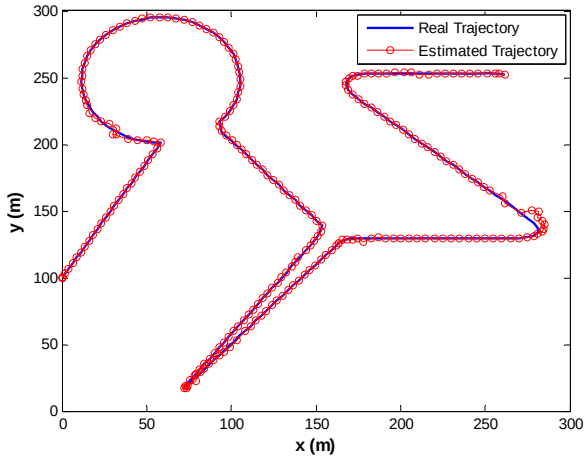


Figure 60 Real and Estimated Target Trajectories

The proposed MS-ASTT scheme is compared against the schemes of Xiao [110], Lin [5] and one with a uniform sampling interval of 0.1sec. The sampling interval variation over time for different schemes is plotted in Figure 61.

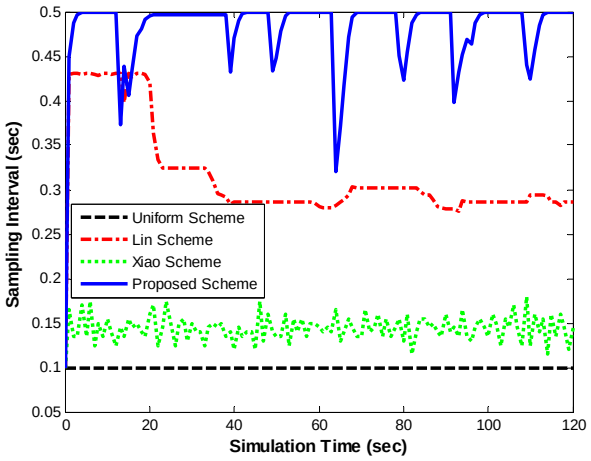


Figure 61 Sampling Interval for Different Schemes

In Xiao's scheme the next sampling interval is computed so that the tracking update error is satisfactory. The threshold of the updated tracking error for Xiao's approach is set to 18. In Lin's approach the next sampling interval is calculated so that the tracking prediction error is satisfactory. The predicted tracking error threshold for Lin's approach is set to 5. In the proposed MS-ASTT scheme, the measured sampling interval is proportional to the location metadata shown in Figure 62. Therefore, the sampling interval is a weighted sum of the measured sampling interval and previous sampling interval. The sampling interval is large during times when the target travels along a uniform path. On the other hand, the sampling interval reduces when the target manoeuvres sharply. The sampling interval progressively increases to its maximum value once the target's path returns to a steady trajectory. In Figure 61 the average sampling interval plotted for the proposed MS-ASTT scheme is 0.48 second.

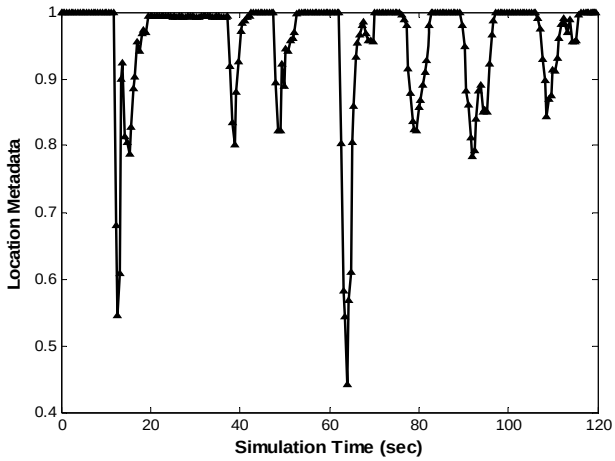


Figure 62 Location Metadata Variations

The tracking update and prediction errors are plotted in Figure 63 and 64, respectively, for different schemes. The uniform scheme obviously produces the smallest tracking error because the tracking snapshot is performed every 0.1 seconds. Unless there is no recovery, the tracking update error for the Xiao scheme is guaranteed to be less than or equal the threshold value (i.e., 18). Similarly, unless there is no recovery, the tracking prediction error for Lin's scheme is guaranteed to be less than or equal the threshold value (i.e., 5). Although the tracking error in the proposed MS-ASTT scheme is the largest, it still successfully tracks the target over the complete observation period.

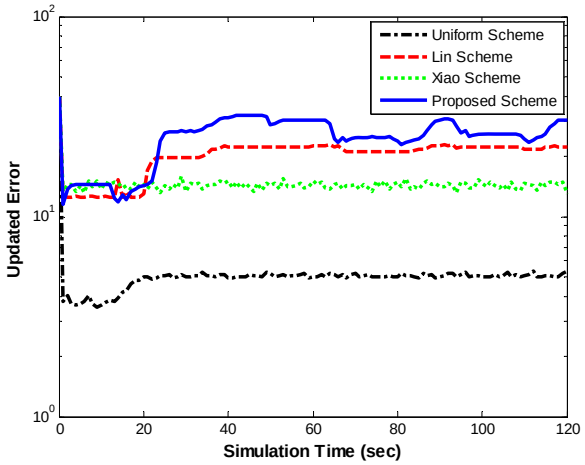


Figure 63 Tracking Update Error for Different Schemes

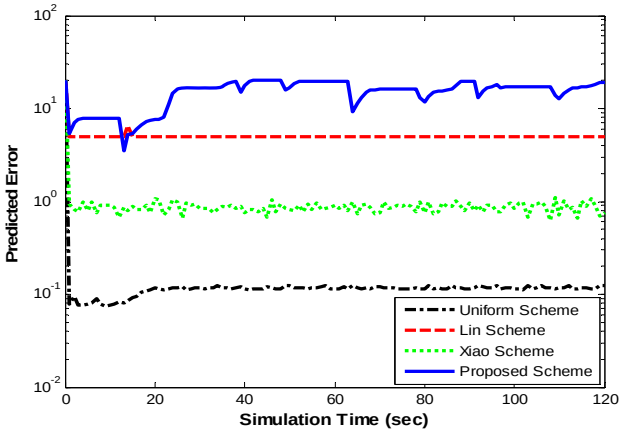


Figure 64 Tracking Prediction Error for Different Schemes

The total energy consumption for different schemes is plotted in Figure 65. At the end of the simulation, the total energy consumption of the uniform, Lin, Xiao and proposed MS-ASTT schemes are 3.2J, 1.1J, 2.4J and 0.68J, respectively, averaged over 20 simulations. Therefore, the proposed MS-ASTT approach can save 79%, 38% and 72% of the energy used by uniform, Lin and Xiao schemes, respectively, representing a significant improvement.

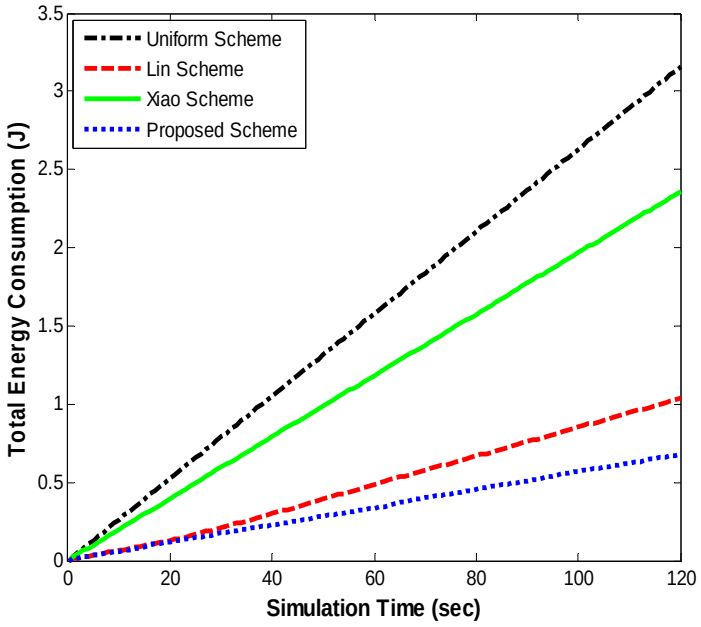


Figure 65 Energy Consumption for Different Schemes

In Table 3, the number of recovery events and total recovery time during the simulation are tabulated. The uniform scheme using a sampling interval of 0.1 second does not have any recovery events. The proposed MS-ASTT scheme has the largest number of recovery events.

The STT Approach	Number of Recoveries	Recovery Time (sec)
Proposed MS-ASTT	1.25	0.094893
Lin Scheme	1.10	0.079778
Uniform Scheme	0	0
Xiao Scheme	0.1	0.007326

Table 3 Recovery Results for Different Schemes

The overhead and recovery times are plotted in Figure 66. The overhead time is the sum of the times required for communication and processing. The communication time is due to "*TRan*" and "*GTrig*" transmissions. The processing time is due to the EKF update or localization processing, sampling interval calculations, EKF prediction processing, the group selection algorithm and the MN election processes. The recovery time is the sum of "*Timer_recovery*" and all "*Timer_levels*" timers, and "*TLos*" and "*TRec*" transmission times. As shown in Figure 66, both times are less than 0.1 seconds which is the minimum sampling interval.

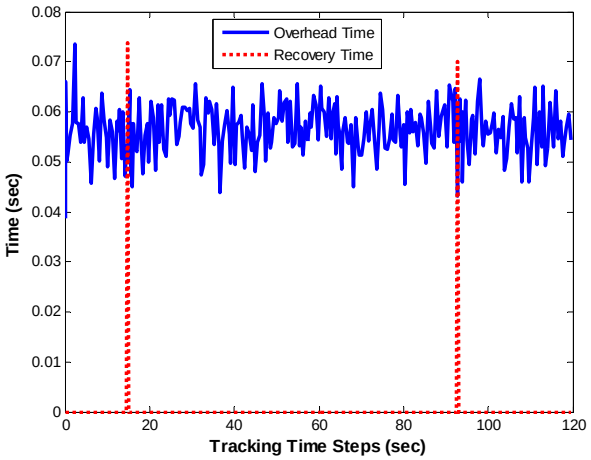


Figure 66 Overhead and Recovery Times

7.3.6 Impact of Adaptive Group Size

In this section, the adaptive group size algorithm proposed in Section 3.9.3 is evaluated. The tracking error threshold (ψ_0) shown in Figure 25 is set to 0.001. A single target travels for 2 minutes with constant speed of 10m/s and importance, $z_T = 1$, along the path shown in Figure 67..

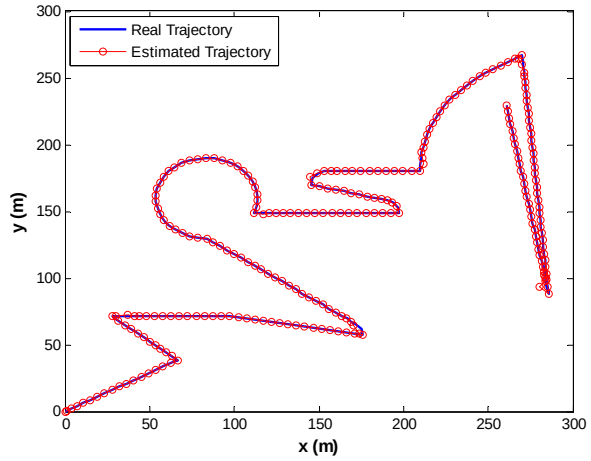


Figure 67 Real and Estimated Target Trajectories

The border sensors initiate the tracking process with initial target state $[0\ \ 5\ \ 0\ \ 5]'$ and initial covariance matrix $10\mathbf{I}\,[110]$. The adaptive group size algorithm uses $n_g^{\min} = 1$ and $n_g^{\max} = 10$. In Figure 67, the true and estimated target trajectories are plotted. The estimated trajectory is close to the true trajectory

The number of sensors for each tracking step is plotted in Figure 68. The results are approximated to integer numbers. Five sensors are the most used group size. However, more or less sensors could be used to try to guarantee the updated tracking accuracy.

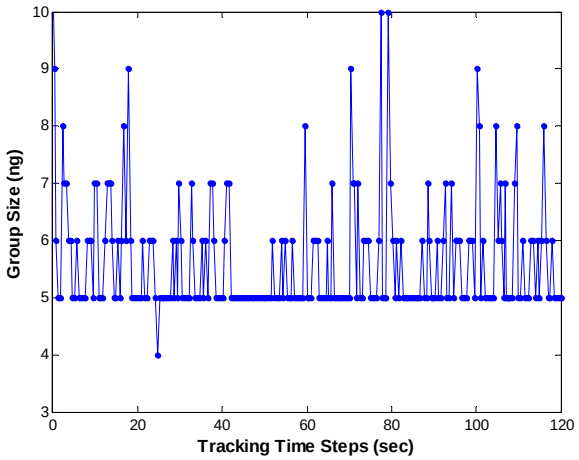


Figure 68 Group Size Variation

The tracking error defined in Equation (3.39) is plotted in Figure 69. The updated tracking error is initially large (i.e., about 20 at time 0). Note that the initial tracking error is not shown in Figure 69 to permit showing other values of the update tracking error more clearly. The tracking error stabilizes as more measurements are obtained. Most of the time, the updated tracking error is below the threshold because more sensors are involved for tracking when updated tracking error is high.

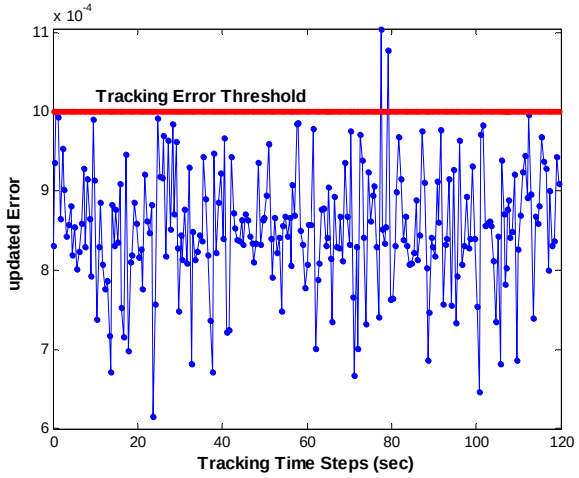


Figure 69 Updated Tracking Error using Adaptive Group Size

7.3.7 Results Discussion

In Figure 50 the number of recovery events increases with increasing the $T_{\max}$ and the velocity because the prediction of the next state is more likely to fail when $T_{\max}$ or the velocity is large, especially during sharp manoeuvrings of the target that happen at the edge of the sensing area. Figure 50 to Figure 52 show that increasing of the number of recovery events slightly increases the energy consumption and dramatically increases the time required for recovery. Therefore, the proposed recovery mechanism is energy

efficient and the main cost of the recovery is the time required to perform it which can lead to target loss during the recovery process.

In Table 2, if δ is set to 1, the load balancing in terms of energy remaining will not be considered in the election algorithm. Therefore, if the battery level of the MN that performs most of the processing and management tasks is small, the MN will die quickly leaving a "hole" in the network coverage and in turn the network lifetime is reduced. Adaptive selection of δ improves the network lifetime and energy saving.

As shown in Figure 53 and 58, group sizes of 1 and 2 have high number of recovery events compared to other group sizes. Therefore, energy consumption increased due to the high number of recovery events. Therefore, selecting group size greater than or equal to 3 can improve the number of recovery times and energy consumption for this particular scenario.

In Figure 61, the sampling interval for the proposed MS-ASTT scheme is large during times when the target travels along a uniform path, and hence the energy efficiency is improved. The sampling interval reduces during the target manoeuvrings to improve the tracking accuracy and continuity.

As shown in Figure 63, 64 and Table 3, the proposed MS-ASTT scheme has the highest tracking error and number of recovery events compared with other schemes. However, the number of recovery events of the MS-ASTT scheme is less than 2 recoveries. Additionally, the proposed MS-ASTT scheme provides the minimum energy consumption as shown in Figure 65. In fact, there is trade-off between the tracking error and the energy consumption. As shown in Figure 58 for group sizes of more than or equal to 3 sensor nodes, the energy consumption increases with increasing group size. However, the group size can be used to maintain the tacking error that is below a predefined threshold. Therefore, the trade-off between the tracking error and energy consumption can be decided according to the target and the application.

In Figure 68 and 69, the updated tracking accuracy is satisfied. The group size is small for small update tracking errors. It starts to rise when the update tracking error is exceeds the threshold. However, a maximum value of group size is selected to reduce the likelihood of channel access contention and the increment of energy consumption due to the use of large numbers of group nodes.

7.4 MS-DMTT Scheme Evaluation

In this section, the MS-DMTT scheme proposed in Chapter 4 is evaluated. The simulation assumptions are presented firstly. Then, different results are presented. The results are critically discussed at the end of this section.

7.4.1 Simulation Setup

As described in Section 3.12, a uniformly distributed random deployment of 500 wireless sensor nodes across an area of 300m × 300m can guarantee the coverage of one target at any location by at least 3 sensors. In this simulation, three targets are assumed to be tracked and any sensor node is capable of tracking and serving only one target at a time. Therefore, 1500 wireless sensor nodes are required to be randomly deployed across the sensing area. For all targets, an adaptive sampling interval is used with $T_{\min}=0.1$ sec and $T_{\max}=0.5$ sec [5][110].

7.4.2 Sensor Nodes Selection

Three targets, Target 1, Target 2 and Target 3 with importance of 30, 20 and 10 respectively are assumed to be tracked. In Figure 70 and 71, particular locations and the selected sensor nodes for the three targets are shown with and without consideration of the target importance respectively.

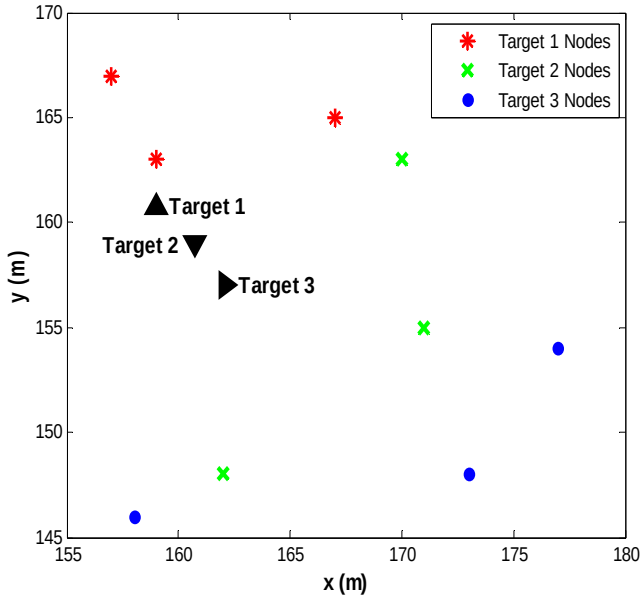


Figure 70 Selected Sensors considering Target Importance

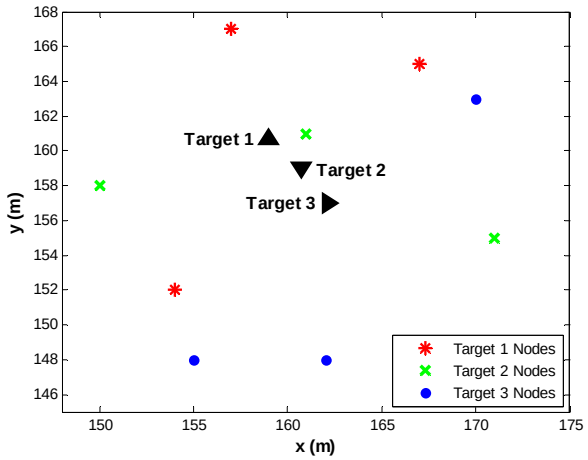
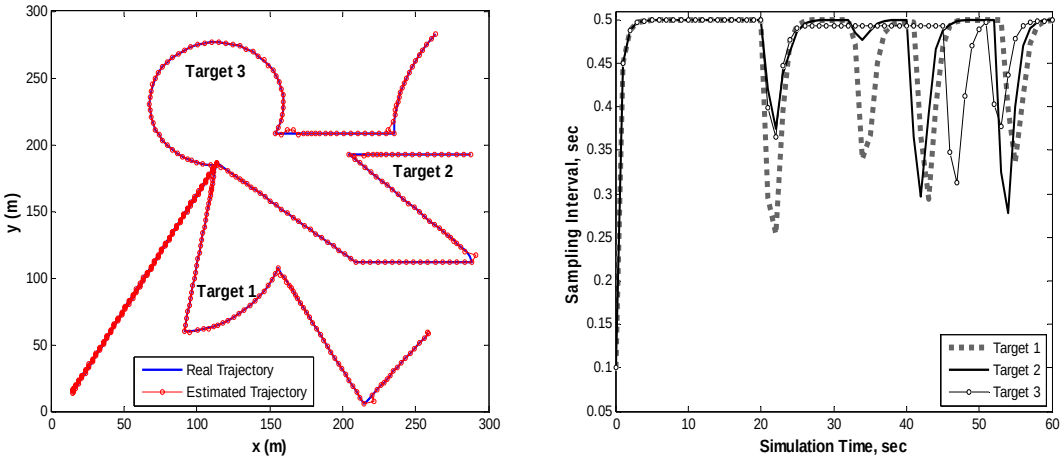


Figure 71 Selected Sensors without considering Target Importance

In Figure 70, since Target 1 is the most important one, its selected group of nodes are the nearest to it compared with other targets. In Figure 71, the sensor selected for a particular target cannot then be reselected for another, even if it has higher importance.

7.4.3 Tracking Error and Sampling Interval

In this section, Target 1, 2 and 3 travel for 60 seconds with constant speed of 10m/s and importance of 30, 20 and 10, respectively, along the paths shown in Figure 72 (a). The border sensors initiate the tracking process with initial Target 1, 2 and 3 states of $[15\ \ 5\ \ 15\ \ 5]'$, $[15\ \ 5\ \ 16\ \ 5]'$ and $[15\ \ 5\ \ 14\ \ 5]'$, respectively. The initial covariance matrix for all targets is $10\mathbf{I}$ [5]. In Figure 72 (a), the true and estimated target trajectories for the proposed scheme are plotted. The estimated trajectories are close to the true trajectories. The sampling interval variations for different targets are plotted in Figure 72 (b).



(a) **(b)**

Figure 72 (a) Real and Estimated Trajectories and (b) Sampling Interval for Different Targets

The tracking update error of T_j is defined as the "trace" of the updated state covariance matrix $\mathbf{P}_{T_j}(k \mid k)$. The tracking update errors for each target with and without consideration of target importance are plotted in Figure 73 and 74, respectively. As shown in Figure 72, targets travel close to each other during the first 20 seconds. Therefore, conflict nodes arise during this period. In Figure 73, during the first 20 seconds, the tracking update error is smaller for high importance targets. On the other hand, in Figure 74, tracking update error takes a similar value for all targets.

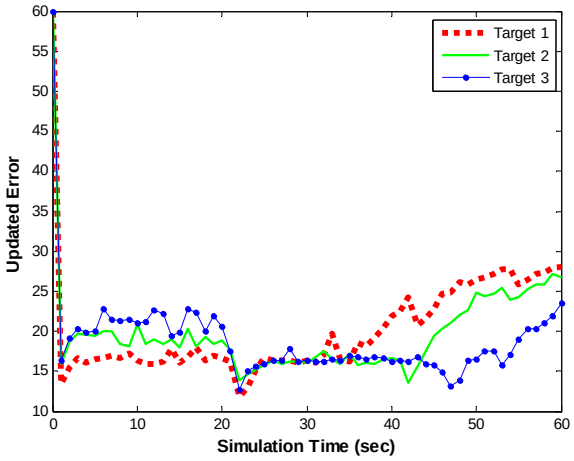


Figure 73 Tracking Update Error for Different Targets with Target Importance

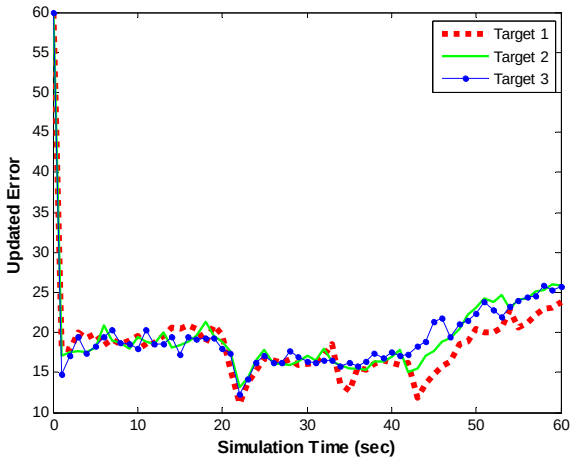


Figure 74 Tracking Update Error for Different Targets without Target Importance

7.4.4 Results Discussion

As shown in Figure 72 and 73, the proposed MS-DMTT scheme can successfully track targets that move along paths that include random abrupt manoeuvrings. The sampling interval for each target is adaptively calculated according to the target location metadata. The calculation of the sampling interval for each target is independent of other targets. Therefore, the MS-DMTT scheme is a series of MS-ASTT problems. In

Figure 72 (b), the sampling interval for each target is based on its past movement pattern (i.e., location metadata). However, the conflict nodes problem can arise in the case of the MS-DMTT scheme because it supports multiple target tracking. The DMS algorithm introduced in Section 4.5.4 is invented to tackle the conflict node problem. The DMS algorithm is a new technique in MTT literature. However, in Section 7.3.5 the proposed MS-ASTT scheme is compared against well-known STT algorithms and the results show that the proposed MS-ASTT scheme performs better than other approaches in terms of energy-efficiency. Therefore, since MS-DMTT is a series of MS-ASTT problems, it will perform better than other tracking schemes that adopt an adaptive sampling interval. Furthermore, the proposed DMS algorithm is evaluated and the results are plotted in Figure 73 and 74. The tracking error is smaller for targets with higher importance because conflict nodes prefer to serve and track the high importance targets.

7.5 MS-AMTT Scheme Evaluation

In this section, the MS-AMTT scheme proposed in Chapter 4 is evaluated and compared with other well-known MTT schemes. The simulation assumptions are presented first. Then, different results are plotted. The results are critically discussed at the end of this section.

7.5.1 Simulation Setup

In this simulation, three targets are assumed to be tracked and the sensor node is assumed to detect and serve more than one target at the same time. As described in Section 3.12, a uniformly distributed random deployment of 500 wireless sensor nodes across an area of 300m × 300m can guarantee the coverage of a target at any location by at least 3 sensors. The sampling interval $\Delta t(k)$ is set to 1 sec for all k and all targets. Z_m, $I_{\max}$ and $I_{\min}$ for Equation (4.33) and (4.45) are set to 100, 30 and 10, respectively.

7.5.2 Sensor Node Selection

Assume three targets, Target 1, 2 and 3 are to be tracked. Target 1 and 3 move in a uniform manner and Target 2 follows a manoeuvring pattern. Figure 75, 76 and 77 show the particular target locations and the selected sensor nodes for the three targets using the selection strategies of closest-sensor, MS-AMTT considering the target importance and MS-AMTT without considering the target importance (i.e., Z_{T_j} is zero for all targets), respectively.

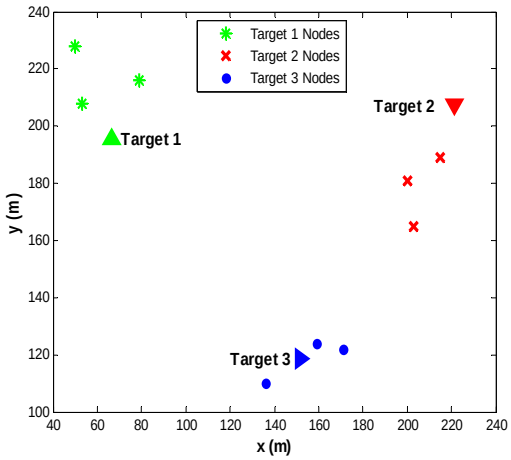


Figure 75 Selected Sensors using Closest-Sensor Selection

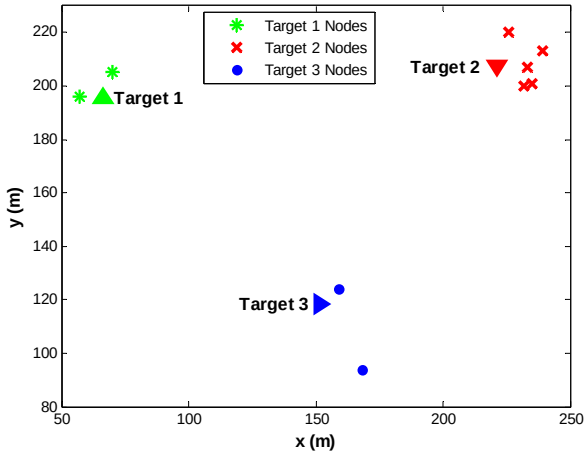


Figure 76 Selected Sensors using MS-AMTT considering Target Importance

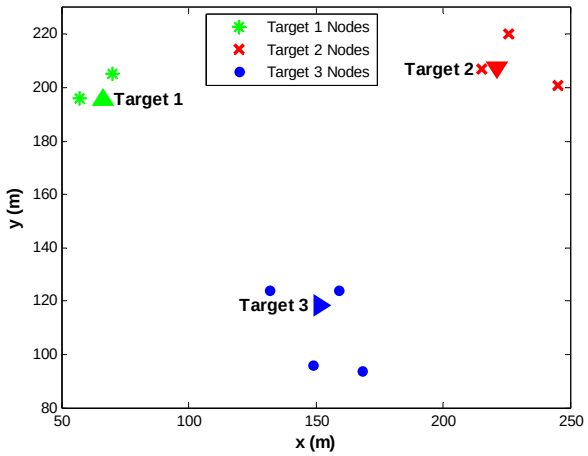


Figure 77 Selected Sensors using MS-AMTT without considering Target Importance

In the blind closest-sensor selection technique shown in Figure 75, the nearest three sensor nodes to the predicted target location are selected for each target. In Figure 76, Target 2 has more sensor nodes to track it because its importance is the highest (i.e.,

Target 2 moves randomly). This compensates for the potential failure in the next state predictions for Target 2 despite to its random movement and allows for seamless tracking of all targets. In Figure 77, the tasking sensor nodes are selected based on the overall update tracking error of the targets defined in Equation (4.28) without considering the target's importance. As shown in Figure 77, Target 3 is tracked by the highest number of nodes despite it moving in a uniform manner.

7.5.3 Targets' Importance and Group Size

In Figure 78, Target 1, 2 and 3 travel for 60 seconds with constant speed of 10m/s. The border sensors initiate the tracking process with initial Target 1, 2 and 3 states of $[0\ \ 5\ \ 300\ \ 5]'$, $[0\ \ 5\ \ 0\ \ 5]'$ and $[300\ \ 5\ \ 0\ \ 5]'$ respectively. The initial covariance matrix for all targets is $10\mathbf{I}$. In Figure 78, the true and estimated target trajectories for the MS-AMTT scheme considering the target importance are plotted. The estimated trajectory is close to the true trajectory.

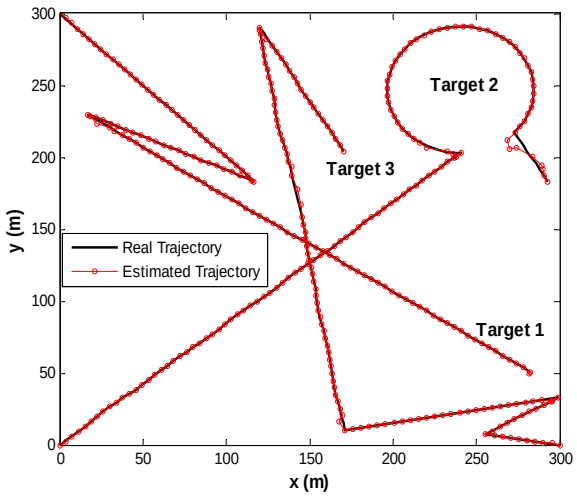


Figure 78 Real and Estimated Trajectories for Different Targets

The target importance and location metadata variation over time for each target are plotted in Figure 79 and 80 respectively. The target importance is inversely proportional to the location metadata of the targets according to the Equation (4.33). As shown in Figure 78 and 79, the target importance is small during times when the targets travel along a uniform path. On the other hand, the target importance increases when the target manoeuvres sharply.

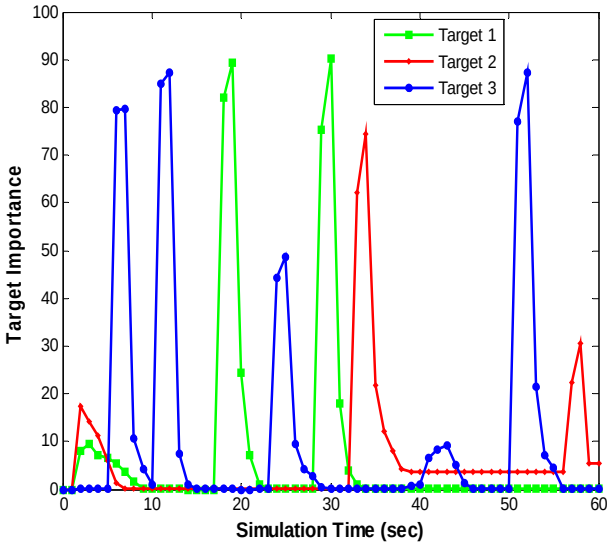


Figure 79 Target Importance for Different Targets

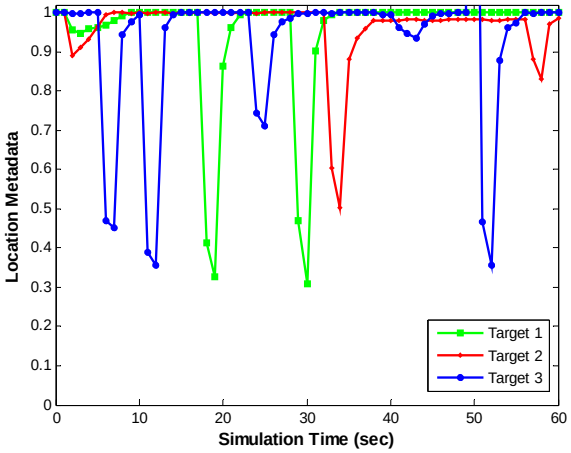


Figure 80 Target Metadata for Different Targets

In Figure 81 and 82 the group size for each target using MS-AMTT, with and without consideration of the target importance, are plotted.

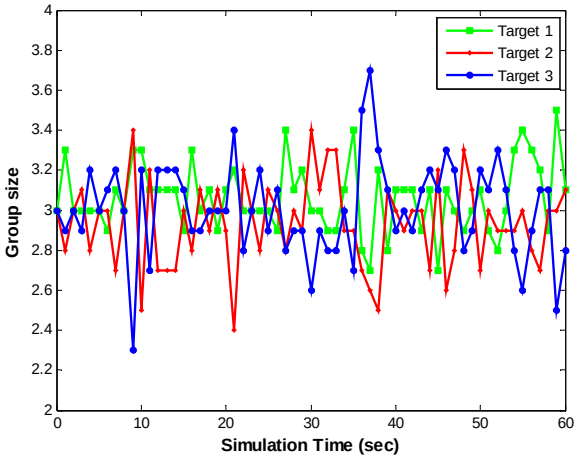


Figure 81 Group Size using MS-AMTT without considering Target Importance

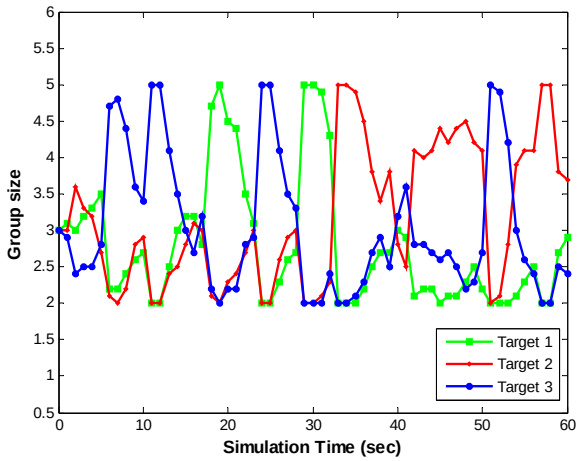


Figure 82 Group Size using MS-AMTT considering Target Importance

In Figure 82, during the target manoeuvrings, its importance is high and hence the objective function is biased by this. Therefore, the final solution involves more sensor nodes for targets that move with abrupt changes of speed and direction. On the other hand, sensor group sizes for targets are chosen in Figure 81 without consideration of the target importance.

7.5.4 Tracking Update Error

In Figure 83, the location and velocity errors, which are defined as the difference between the estimated and real values, are plotted for MS-AMTT with and without consideration of the target importance. Generally, the location and velocity errors are high during the target manoeuvrings due to the prediction error and they are low when the targets travel in predictable and uniform fashion. However, as shown in Figure 83 the location and velocity errors when considering the target importance are small compared with MS-AMTT without considering the target importance. The Root Mean Square Error (RMSE) of location and velocity over 20 simulations runs for MS-AMTT scheme and the closest-sensor selection strategy during its successful tracking time are tabulated in Table 4. The percentage of tracking loss time for the closest-sensor selection strategy is shown in Table 5. Obviously, closest-sensor selection strategy has the largest errors because this strategy assigns the same number of sensor nodes for each target without considering the movement patterns for the targets (i.e., targets' importance).

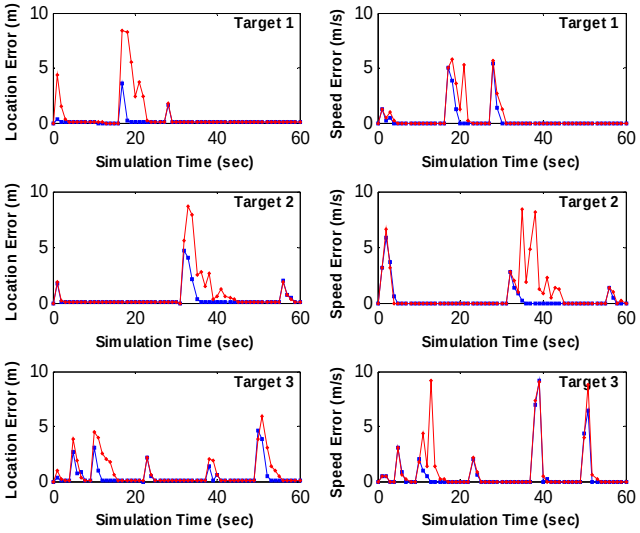


Figure 83 Location and Velocity Errors using MS-AMTT without (red curves) and with (blue curves) considering of the Target Importance

Sensor Selection Strategy	Location RMSE (m)			Velocity RMSE (m/s)		
	T1	T2	T3	T1	T2	T3
MS-AMTT with Importance	0.2	0.2	0.4	0.3	0.3	0.8
MS-AMTT without Importance	0.6	0.6	0.7	0.5	0.8	0.9
Closest-sensor Selection	4.9	2.0	2.7	2.9	2.1	2.0

Table 4 Average Location and Velocity Errors

7.5.5 Local Search Iteration

The average number of iterations, computational time and the percentage of tracking loss events for MS-AMTT with and without considering the target importance, and closest-sensor schemes are shown in Table 5. The tracking loss is the percentage of the time during which the target is lost. The computational time of the MS-AMTT scheme with considering the target importance is greater than the case of MS-AMTT without consideration of target importance. The closest-sensor selection scheme obviously does not require any local search to select the tasking sensor nodes. However, due to target sharp manoeuvring, the closest-sensor selection approach tracks the target only 20% of the simulation time.

Sensor Selection Strategy	Number of Iterations	Computational Time (sec)	Percentage Tracking Loss
MS-AMTT with importance	10.038	1.585	0
MS-AMTT without importance	9.421	1.215	0
Closest-sensor Selection	0	0	80

Table 5 Average Iterations, Computational Time and Tracking Loss

The performance results for adaptive and fixed allowable iterations are tabulated in Table 6 and 7. With increasing allowable iterations, the average number of iterations and computational time increase while the location and velocity errors decrease. However, using adaptive allowable iterations yields similar errors to the fixed schemes using high allowable iterations whilst the computation time is near that of the fixed scheme using small allowable iterations.

Allowable Iterations	Number of Iterations	Computational Time (sec)
30	13.682	2.350
20	12.500	1.914
10	9.019	1.014
Adaptive	10.038	1.585

Table 6 Average Iterations and Computational Time

Allowable Iterations	Location RMSE (m)			Velocity RMSE (m/s)		
	T1	T2	T3	T1	T2	T3
30	0.2	0.2	0.3	0.2	0.3	0.7
20	0.4	0.3	0.4	0.4	0.3	0.8
10	0.6	1.1	0.6	0.8	1.4	0.9
Adaptive	0.2	0.2	0.4	0.3	0.3	0.8

Table 7 Average Location and Velocity Errors

7.5.6 Comparison with Other Well-Known MTT Schemes

In Table 8 and 9, the performance of the proposed MS-AMTT scheme considering target importance is compared with "Tharmarasa" and "naive" approaches [103] using 500 and 1000 sensor nodes. The maximum allowable iterations is set to infinity. This means that the local search runs until the final solution is found. In Tharmarasa's method, the objective function for each sensor node is computed using that sensor node alone. Then, the sensor nodes are ranked according to their objective functions. The best n_G sensor nodes are chosen as the initial solution. For each current solution, the remaining sensor nodes are ranked based on their individual objective functions (i.e., when using each sensor node alone). Then, each sensor node of those remaining are ranked, replacing one-by-one the n_G sensors in C_s to form n_G neighbours. The naive method chooses the sensor nodes that have the maximum index (i.e., sensor identification) as the initial solution. The remaining sensor nodes are ordered by index

to generate the neighbourhood. As shown in Table 8 and 9, the proposed MS-AMTT method has the smallest value of computational time and the number of iterations.

Local Search Strategy	Computational Time (sec)	Number of Iterations
MS-AMTT Method	3.528	14.523
Tharmarasa Method	6.944	30.497
Naive Method	8.042	32.698

Table 8 Average Iterations and Computational Time using 500 Sensors

Local Search Strategy	Computational Time (sec)	Number of Iterations
MS-AMTT Method	20.352	15.264
Tharmarasa Method	79.969	37.193
Naive Method	84.428	38.920

Table 9 Average Iterations and Computational Time using1000 Sensors

7.5.7 Results Discussion

The closest-sensor selection strategy shown in Figure 75 is not always a good solution due to the target location triangulation calculation [103], the sensor nodes' co-linearity [95] and the prediction failure of the target next state that likely happens during random movements of the target. Moreover, the sensor nodes selected for Target 2 shown in Figure 75 are far from the target's true location because the sensor node selection is based on the distance from the predicted target location which is not accurate for Target 2 due to its manoeuvring. Therefore as shown in Table 4, the average location and velocity errors of the closest-sensor selection strategy is high compared with the MS-AMTT scheme. Figure 76 and 77 show that considering the target importance in the objective function of sensor selection, defined in Equation (4.28), can force the selection of more sensor nodes for manoeuvring targets and, in turn, the tracking continuity is improved for all targets. On the other hand, the goal of the objective function without consideration of the target importance is simply to minimize the overall tracking error of the targets.

As shown in Table 5, considering the target importance in the objective function requires more iterations to obtain a solution. Therefore as shown in Figure 78 and 79, the target importance is small during times when the targets travel along a uniform path, and hence the local search efficiency (i.e., the number of iterations to get a suitable solution) is improved because the solution is obtained without the objective function being biased by the target importance. On the other hand, the target importance increases when the target manoeuvres sharply, and hence the tracking error shown in

Figure 83 is reduced and seamless tracking is achieved because more sensor nodes are involved in the tracking for more important targets as shown in Figure 82. For example, at time 18 seconds in Figure 82, Target 1 is manoeuvring. Therefore, Target 1 has more selected sensor nodes to improve its tracking accuracy and maintain continuity.

Unlike the proposed MS-AMTT scheme, the number of sensor nodes used in Tharmarasa's method is not ranked by the number of targets they detect. Additionally, the solutions used in Tharmarasa's method do not check whether all targets are detected by at least one sensor node. Therefore, as shown in Table 8 and 9, the proposed MS-AMTT scheme performs better than Tharmarasa's method.

As shown in Table 8 and 9, the main drawback of the using optimization techniques for sensor selection in MTT is the risk of experiencing a computational time that exceeds the sampling interval. However there are different approaches to overcome this drawback. One approach is to use more powerful processing but this may not be appropriate for tiny sensor nodes. Another approach is to increase the sampling interval [103] but this will degrade the tracking accuracy and potentially lose targets for long periods. A third approach is to adopt BTMS and BITA algorithms presented in Chapter 5 so that the execution of the algorithm is parallelized among a group of nodes. As shown in Section 7.6 and 7.7, this approach is efficient both in terms of execution time and energy saving.

7.6 BTMS Algorithm Evaluation

In this section, the performance of the BTMS algorithm, presented in Chapter 5, is evaluated. Recall from Chapter 5, BTMS is used to parallelise the execution of an application that can be decomposed into a number of dependent tasks. The performance of BTMS algorithm is compared against the MTMS [128] algorithm.

7.6.1 Simulation Setup

This simulation setup is performed before the target tracking simulations take place. The sensor nodes are randomly (i.e., using uniform distribution) deployed across an area of 1km × 1km. ε_s, τ_s and λ in Equation (5.6), Equation (5.8) and Equation (5.9) are set to 0.5uJ, 0.6528ms and 0.5, respectively. At the beginning of the simulation, the sensor node s_T, that makes a request to its neighbouring sensor nodes to ask them to share in its execution of an application, is selected randomly from the all sensor nodes. The subtasks of the application DAG are mapped to the neighbours of the sensor node

s_T according to the BTMS scheme proposed in Figure 42. This means that it is not necessary to use all the neighbours of sensor node s_T to execute the application DAG. Mapping and scheduling are performed in the neighbours of sensor node s_T until all subtasks of the DAG are completed.

7.6.2 Network Node Density

The sensor node density is first calculated. The simulation finishes for each trial when the number of target arrivals reaches fifty. Some of these targets will find nodes that can detect and serve them and other targets will not find any nodes around them. Therefore, the percentage of targets served is calculated after the end of each trail based on the formula:

$$\text{Target Detection Percentage}(\%) = (\text{Number of served targets}/50)*100 \quad (7.2)$$

The target detection percentage reaches 100% when the node density is about 350 nodes in the chosen evaluation area. It means that all the target arrivals will find at least one node to serve them. For this reason, the node density is chosen to be 350 nodes in the ensuing simulations.

7.6.3 Real Example of Distributed Visual Surveillance

In [128], a distributed visual surveillance application is presented. Figure 84 shows an example visual surveillance DAG.

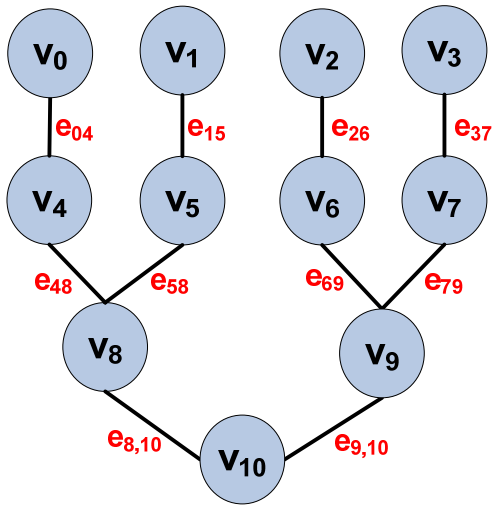


Figure 84 DAG of Visual Surveillance

Tasks V_o to V_3 represent background subtraction and bounding box abstraction. The moving object will be bordered by a rectangular bounding box from each camera and it is represented by the vertices of the bounding box. Tasks V_4 to V_7 represent the estimation error. To eliminate the estimation error, the location estimations from all

cameras are combined in task V_8 to V_{10}. The communication edges e_{04} to e_{37} represent the bounding box coordinates. e_{48} to $e_{9,10}$ represent the estimation object location and error estimation. Finally, the object location is calculated and sent to the sink node.

The Collaborative Execution Time (CET) and energy consumption is evaluated for the visual surveillance application DAG using the proposed BTMS and MTMS algorithms [103]. The simulation results are averaged over 20 runs with different visual image sizes (i.e., different number of computational cycles for the tasks and data size for communication edges). The average CET and energy consumption are tabulated in Table 10.

BTMS can perform better than MTMS in terms of CET and energy consumption. In BTMS, tasks in each level are arranged in decreasing order. Therefore, the large tasks are mapped first. Therefore, this feature decreases the CET because instead of executing large task after finishing the small ones, the large tasks will be executed concurrently with the execution of smaller ones

Algorithm	CET (ms)	Energy Consumption (mJ)
BTMS	25.496	2.304713
MTMS	44.2731	2.468315

Table 10 Results for Visual Surveillance DAG

7.6.4 CET and Energy Consumption using Random DAG

In this section, the simulation is repeated 250 times using 250 different DAGs. Each DAG is created using the following parameters: maximum immediate successors (maxSucc) of entry and normal tasks = 3, the minimum immediate successors (minSucc) of entry and normal tasks = 1, number of entry tasks = 5, number of normal tasks = 10 and number of exit tasks = 1. The entry tasks should not have any immediate predecessors and the exit tasks should not have any immediate successors. All other tasks (i.e., normal tasks) have at least one immediate predecessor. The number of immediate successors (numSucc) of entry to a normal task, (v_i) are uniformly distributed over [minSucc, min {maxSucc, number of tasks greater than (v_i) without including the entry tasks}]. After that the immediate successors of entry or normal tasks, (v_i) are chosen randomly from the tasks greater than (v_i) without including the entry tasks.

The deadlines are chosen so that they increase with increasing the computational load. For simplicity, deadlines are selected to be equal to the serial execution time,

which is the time needed to execute the application using one sensor node. Therefore, the deadlines increase with increasing computational load. In Figure 85, the CET increases with increasing deadlines for a fixed communication load (i.e., the data size for all communication edges). With increasing communication load, CET increases for a specified deadline. Therefore, CET increases with increasing communication and computational load.

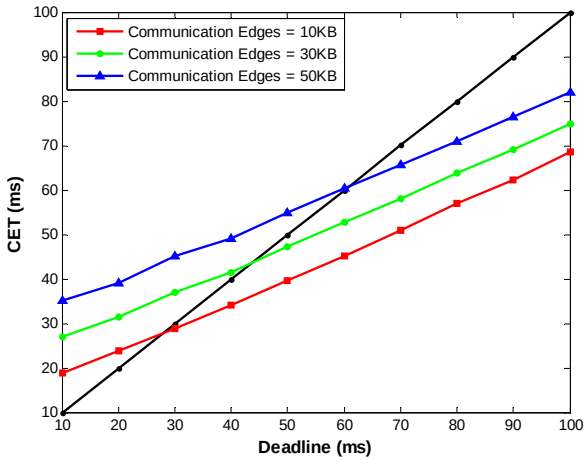


Figure 85 CET versus Deadline

In Figure 86, the energy consumption increases with increasing deadline for fixed communication load because as the computational loads increases, the deadline will increase and in turn, the computational energy consumption increases. With increasing communication load, energy consumption increases for a specified deadline because more energy will be dissipated when there is more communication. Therefore, energy consumption increases with increasing communication and computational load.

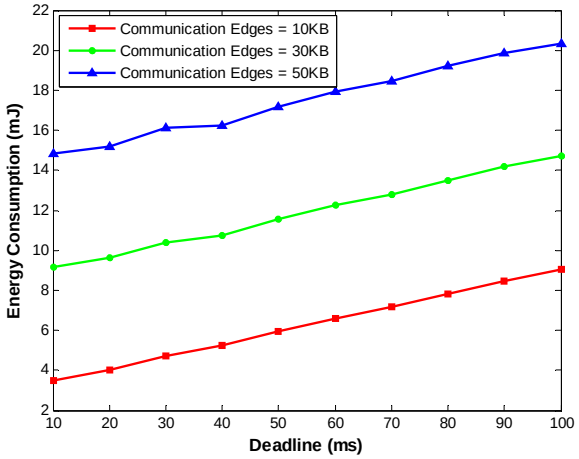


Figure 86 Consumed Energy versus Deadline

7.6.5 Network Lifetime Performance

In this section, the simulation is repeated twenty times for different network topologies. The average lifetime performance ratio (LT_{BTMS} / LT_{MTMS}) of the lifetimes of the proposed BTMS and the MTMS algorithms are plotted in Figure 87. The BTMS algorithm improves the lifetime relative to MTMS.

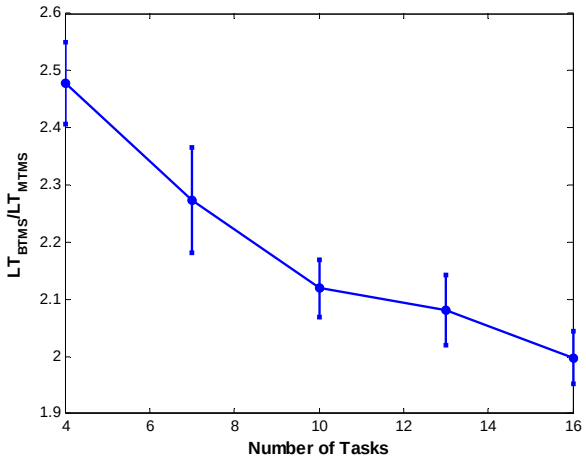


Figure 87 Lifetime Ratio versus Number of Tasks with a 95% confidence interval

7.6.5 Results Discussion

In Figure 85, the deadline has to be large enough so that CET can meet it. Therefore, the BTMS would be inefficient if the computational complexity was small relative to the communication overhead.

Figure 87 shows that BTMS improves the network lifetime compared with MTMS because BTMS adopts the decision rules presented in Chapter 5. The BTMS algorithm allows the mapping of tasks onto nodes on which the predecessors are mapped. Therefore, the energy consumption is reduced.

7.7 BITA Algorithm Evaluation

In this section, the performance of BITA is evaluated. Recall from Chapter 5, the BITA algorithm is used to parallelise the execution of applications that can be decomposed into independent and equal tasks. The performance of BITA is compared against Distributed Computing Architecture (DCA) [133] in which the cluster head performs high-level tasks. Therefore, in DCA, the number of sensor nodes or the group size (n_m) that can share to execute the application in parallel is 1.

7.7.1 Simulation Setup

In this simulator, 350 wireless sensor nodes are randomly (i.e., using uniform distribution) deployed across an area of 1km × 1km. The sensor node s_T that makes a request to its neighbour sensor nodes to ask them to share it in the execution of an application is selected randomly from the all sensor nodes. After finishing serving the current application at s_T, another one is randomly selected. Each sensor node is assumed to be able to execute 10 tasks per second. Z and β are set to 0.02 and 0.5 in Equation (5.10) and Equation (5.11) respectively. Each task is assumed to consume 1 unit of energy. Unless specifically stated otherwise, the total number of application independent tasks (N) is set to 100 and the maximum battery capacity of the sensor nodes ($E_{s_k}^{\max}$) is set to 500 units. The simulator is run twenty times with different node deployments and is stopped after 10 minutes.

7.7.2 Cooperative Execution Time (CET)

In Figure 88, for a fixed group size (n_m), the CET increases with increasing the number of tasks (N) because the computational load increases with more tasks (N). BITA ends with less CET than DCA because of the parallel computation used by BITA. CET decreases with increasing group size (n_m) because based on Equations (5.12) to (5.14), more load balancing can be achieved which increases the execution parallelism.

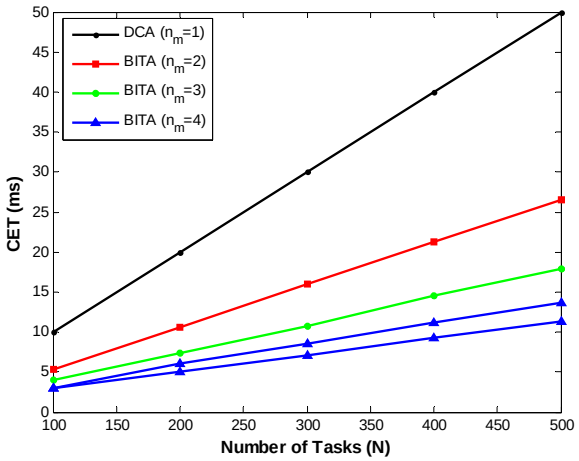


Figure 88 CET versus Number of Tasks (*N*)

7.7.3 The Performance Metric (Pm)

In this simulation, $E_{s_k}^{\max}$ is set randomly between 100 to 500 units using a discrete uniform distribution. The network performance metric (P_m) is defined in Equation (7.1).

In Figure 89, the performance metric (P_m) versus the group size (n_m) is plotted. Due to node cooperation, BITA has a lower (P_m) than DCA because the 100 energy units required for each application is distributed between the group of sensor nodes and consequently the node can live for longer. With increasing group size (n_m), a lower (P_m) is achieved because based on Equation (5.14), we can obtain more load balancing among the nodes.

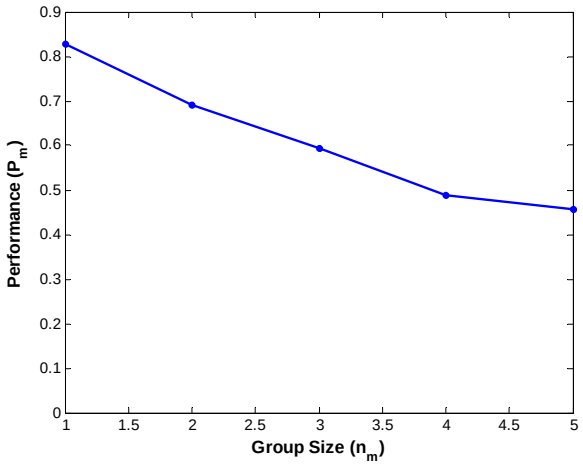


Figure 89 Performance Metric versus Node Group Membership Size (n_g)

7.8 Code Verification

In Appendix B, code verification for the proposed target tracking, and BITA and BTMS algorithms is presented. Analytical analysis is compared with simulation results to confirm the correctness of the simulation models.

7.9 Chapter Summary

This chapter presents different performance simulation results for the proposed target tracking and task mapping and scheduling algorithms. The simulation results show that the proposed schemes can successfully track targets that move along paths that include random abrupt manoeuvrings. The MS-ASTT scheme is compared against Xiao's and Lin's schemes and the results show that the proposed MS-ASTT approach reduces the energy consumption whilst maintaining seamless tracking. Simulation results show that the tracking accuracy of the proposed MS-DMTT scheme is better for targets of higher importance. Simulation results also show that the proposed MS-AMTT scheme reduces the computational time whilst maintaining seamless tracking compared with Tharmarasa's and "naive" approaches.

A critical assessment of these results is given. Code verification is given in Appendix B by analyzing typical scenarios analytically and comparing with the simulation results.

Chapter 8 Discussion and Conclusions

8.1 Chapter Introduction

The STT, MTT and TMS schemes in WSNs are presented in Chapter 3, 4 and 5, respectively. In Chapter 7, simulation results evaluating the proposed target tracking and TMS are introduced. This chapter provides a discussion of the simulation results, and the conclusion for this thesis.

8.2 Discussion

In this section, the results provided in Chapter 7 are analysed and discussed. The proposed target tracking schemes are firstly discussed. After that, the proposed task mapping and scheduling algorithms are analysed.

8.2.1 Target Tracking in WSNs

In Section 7.3.2, the recovery mechanism in target tracking is examined. The target recovery provides a reliability mechanism for the tracking operation such that the target can be recaptured if it is lost during the tracking process. Increasing the tracking sampling interval leads to an increasing number of recovery events because the prediction of the target state is more likely to fail, especially during sharp manoeuvrings of the target. Despite the recovery mechanism being reasonably energy-efficient the main drawback is the time required for recovery.

The results presented in Section 7.3.3 show an improvement in network lifetime and energy saving when adopting the adaptive calculation of the weighting parameter of Equation (3.41). The MN is elected to be the nearest node of a group of nodes in the case of having the same energy level for all group members. On the other hand, the MN is elected to be the node that has the maximum energy remaining if the group nodes have different energy levels. In Section 7.3.4, increasing the group size reduces the number of recovery events, which is costly in terms of time. In Section 7.3.5, the idea of using an adaptive sampling interval according to the target historical location improves the energy efficiency compared with other well-known schemes. Furthermore, the target can be successfully tracked during the abrupt manoeuvrings. Energy saving is a crucial goal in WSNs because the sensor nodes have limited energy. If one sensor node dies, it reduces the coverage in the network. Sensor nodes are typically very difficult to access after deployment. Therefore, if one sensor dies due to running out of energy, often it

cannot be recharged. The main drawback of the proposed STT is the lack of the tracking accuracy compared with other tracking schemes. However, the tracking accuracy can be improved by using multiple sensor nodes to track the target. Therefore, there is trade-off between improving the tracking accuracy and the energy consumption. In Section 7.3.6, the updated tracking accuracy is satisfied by using adaptive group size. The group size is small for small update tracking errors. It starts to rise when the update tracking error exceeds the threshold. Consequently, according to the results in Section 7.3.4 the number of recovery events is reduced.

In Section 7.4, the problem of conflicting nodes in MTT is handled. It is shown that it is possible to improve the tracking accuracy of high importance targets compared with others, by setting the conflict nodes preference for serving high importance targets. In Section 4.6, sensor selection in MTT is considered as a solution of an optimization problem whose main goal is to maximize the overall tracking accuracy of the targets. The simulation results in Section 7.5 show that the proposed MS-AMTT scheme reduces the average location and velocity errors compared with the closest-sensor selection strategy as the sensor nodes are selected based on the objective of improving the overall tracking accuracy. The adaptive calculation of target importance according to the target's previous locations improves the tracking continuity for all targets because the targets that travel in irregular patterns assigned more sensor nodes to track them (as they are classified as high importance targets). The proposed MS-AMTT scheme is compared against Tharmarasa's and naive methods and it is found to perform better than other schemes because the number of sensor nodes used in Tharmarasa's method is not ranked by the number of targets they detect. The computational complexity of the proposed MS-AMTT scheme is controlled by adaptively calculating the maximum number of allowable iterations according to the target's metadata. The main drawback of the using optimization techniques for sensor selection in MTT is the computational time that may exceed the sampling interval. However, this drawback can be overcome by adopting the BTMS and BITA algorithms proposed in Chapter 5.

8.2.2 Task Mapping and Scheduling in WSNs

In Section 7.6 and 7.7, the proposed BTMS and BITA algorithms are evaluated. The application execution is shared among more than one sensor node, which enables distributing the resources required to serve the target among the group nodes. Additionally, this concurrent processing decreases the prevalence of gaps in the network

caused from dying nodes and consequently increases the network lifetime. Moreover, decision maker is introduced in each node specifically to improve the lifetime of the network. The concurrent processing also facilitates the timely completion of real time applications by exploiting the speed-up resulting from the decomposition. As a result, the service time (i.e., CET) is improved, the rate of node deaths reduces and network can stay alive for longer. As shown in the simulation results of BTMS, there is a communication cost associated with exchanging the tasks dependencies among the nodes. Therefore, the parallelism of the application is useful if the computational costs of its dependent tasks are high enough compared to the communication costs of the dependencies. For example, with an application that has small computational cycles, it may be faster to execute it using one node rather than parallelizing it among several nodes because the time to send the dependencies among the nodes may well exceed the execution time of the whole application. This is a well-known result in parallelism. The coarse granularity parallelism which has large amounts of computational work compared with communication events is better that fine granularity parallelism which has small amounts of computational work compared with communication events [153].

8.3 Conclusions

In this thesis, the problem of energy-efficient, reliable, accurate and self-organized target tracking in WSNs is considered for sensor nodes with limited physical resources and abrupt manoeuvring targets. MS-ASST, MS-DMTT and MS-AMTT schemes are proposed for target tracking. The main motivations of this research are to improve the tracking accuracy, reliability and continuity, and the energy-efficiency of the target tracking system.

The MS-ASST scheme is used for STT and its operation can be summarized in four steps. Firstly, the sampling interval is computed using the location metadata pertaining to the target's past locations. Secondly, the next tracking group is proactively selected based on information associated with the predicted target location probability density function. The group size is adaptively changed such that the tracking accuracy is improved. Thirdly, one of the group nodes is elected as a MN so that energy efficiency and load balancing are improved. Finally, target recovery is supported to provide tracking reliability in case of target loss due to selection or prediction failures. Therefore, the tracking continuity and energy-efficiency are improved. In the MS-

DMTT scheme, a decision algorithm is proposed to allow the "conflict" nodes that are located in the sensing areas of more than one target at the same time to decide their preferred target according to the target importance and their distance to the target. The operational steps in MS-ASST scheme are as used for the MS-DMTT scheme but for multiple targets.

In MS-AMTT scheme, the target importance and allowable iterations are computed according to the location metadata derived from target's past locations, by which the movement pattern is computed. Then, the next tracking groups are proactively selected such that the tracking continuity and accuracy are improved. After that, one node from the group is elected to be the MN and the LN is elected from the sensor nodes inside the area that is surrounded by group MNs.

In this thesis, the problem of task mapping and scheduling in WSNs is also considered. BITA and BTMS algorithms are developed to execute an application using a group of sensor nodes. BTMS is used as the TMS algorithm to parallelize the execution of an application by decomposing it into dependent tasks. A DAG is adopted to model the application. BTMS is designed to map and schedule the application's tasks to sensor nodes so that the energy consumption is reduced and the application deadline is met. BITA is used as the TMS algorithm when it is possible to divide the application into independent equal-weighted tasks. The main goal of BITA is to reduce the execution time of the application by parallelizing its execution and to increase the network lifetime by exploiting load balancing.

The proposed tracking and TMS approaches are inspired from the biological principles. The inspiration is mainly from the biological behaviours of differentiation in zygote formation and the chemical emission. In target tracking schemes, the sensor nodes before the selection and election algorithms were all equal. The selection algorithm differentiates the functions of the sensors nodes so that some of them will be selected to sense the target and others will remain in a sleeping mode. The election algorithm classifies the selected group nodes into one MN and possibly one or more HN(s). In addition, the target is treated as a virtual chemical emitter. The nodes start from an initially uniform state and then exhibit some kind of differentiation to execute an application in a coordinated manner. The application tasks are mapped to the sensor nodes according to the node's available resources and location. Therefore, each node is specialized to execute particular tasks based on its suitability.

Simulation results show that compared with other well-known schemes, the proposed tracking, task mapping and scheduling schemes can provide a significant improvement in energy-efficiency, network lifetime and computational time, whilst maintaining acceptable accuracy and seamless tracking, even with abrupt manoeuvring targets.

Chapter 9 Future Work

The proposed STT tracking schemes are evaluated assuming targets travel with constant speed. Therefore, the ratio between T_{max} and T_{min} in Equation (3.32) is fixed. The tracking algorithms could be developed further by accommodating variable speed targets. In this case, the ratio between T_{max} and T_{min} would need to be calculated at various times according to the speed of target.

Determination of minimum sampling interval of the target so that tracking continuity and energy-efficiency are improved is a research topic receiving considerable attention. An analogy can be made with Nyquist theory where to avoid aliasing, the minimum sampling frequency should be at least two times the highest frequency contained within the signal [23].

In the target dynamic models proposed in Chapters 3 and 4, the acceleration of the target is modelled by a Gaussian distribution. The proposed tracking scheme can be evaluated in future work using more advanced dynamic models that are introduced in [73].

The sensing area, target states and sensor node coordination are assumed to be in a two-dimensional plane. Future work could focus on advancing the proposed algorithms for use with three-dimensional space.

In this thesis, EKF is adopted to calculate the predicted and updated target states and their covariance matrices. The Particle Filter (PF) proposed in Chapter 2 can be also used for the same purpose. Evaluating the proposed algorithms using PF and comparing them with the ones using EKF would be an interesting research topic with target tracking. Other filtering techniques, which are proposed in Chapter 2, could be used as well.

An acoustic signal is used in this thesis to permit a passive detection mechanism to measure the range of the target. Bearing sensor nodes can measure the angle of the target with respect to the sensor location [67][98]. Therefore, future work could focus on the sensor nodes that can measure the angle as well as the range of the targets using other passive or active detection techniques. Furthermore, sensor nodes are assumed to detect the target with 100% probability if the target is in their sensing range. The detection model could be advanced by assuming a predefined probabilities model for the sensor node detection [5].

In the proposed MTT schemes, false alarms in the network and the data association problem that determines which measurements were generated by which targets are not tacking into the account. The proposed MS-AMTT could be developed further by taking into the account false alarms within the network. It could then be enhanced by considering the problem of data association.

In the proposed task mapping and scheduling algorithms, the application is modelled using a DAG. Generating the DAG representations of the proposed STT and MTT tracking algorithms could be examined further in future work.

Appendix A Simulation Framework

A.1 Appendix Introduction

This appendix introduces the simulation framework. Detailed descriptions of the target tracking, task mapping and scheduling models are firstly presented. After that, the pseudo code for the various simulation events is explained. The C++ code is enclosed with this thesis (or available from the author).

A.2 Detailed Description of Simulation Framework

In this section, the developed models are introduced in detail. Firstly, the target tracking model is presented. After that, task mapping and scheduling models are explained.

A.2.1 Target Tracking Model

Figure 90 shows the pseudo code of the target-tracking model. In line 1, the needed C++ libraries are included such as math, input/output, list, vector and string libraries. The external libraries (header files) include the Mersenne Twister random variable generator, matrices manipulation and implemented data structure header files. In the data structure header file, the sensor node defines a class that encapsulates all the node attributes such as node ID, remaining energy, CSMA/CA attributes, neighbours information and routing table. As discussed in Chapter 6, the simulation is event driven. Therefore, the event is defined as a class that includes the event parameters such as event time (which indicate the time of event execution), event place (that shows on which node the event will be executed) and event name. In line 2, the simulation constants are given, including the sensing area dimensions, radio range, sensing range, CSMA/CA parameters, recovery timers, weighted parameters, energy-consumption parameters, tracking parameters, number of sensor nodes and event names. The global variables and statistical counters include writer variable that writes the results in text files, world array that models the sensing area, the number of dying nodes variable, matrices that encapsulate the target predicted and updated states, and their covariance matrices, group vector, that stores the current tasking group to track the target, the number of recovery events, the overhead time, a random variable that is generated for Mersenne Twister class, the events variable that is declared as list, the sensor nodes variable that are stored in vector and total energy consumption counter.

1. Include the required C++ and external libraries;
2. Declare the simulation constants, global variables and statistical counters;
3. Open the text files to store the simulation results;
4. Initialize the simulator **{** //start of initialization
5. Set the simulation time (sTime) to zero;
6. Create the sensor nodes;
7. Create the sensing area and deploy the sensor nodes randomly in it;
8. Schedule the TArrive event at sTime and insert it in the event list variable;
9. **}** //end of initialization
10. **while** (sTime < Stop Condition) **do**:
11. Get the next event from the event list and update the sTime (i.e., sTime = Event time);
12. **switch** (event) **do**:
13. **TArrive**: Execute the target arrive procedure;
14. **LOCALIZATION:** Execute the target localization procedure;
15. **PREDICTION:** Execute the target prediction procedure;
16. **NextSnapshot:** Execute the target next snapshot procedure;
17. **Ready:** Execute the Ready event procedure to transmit messages;
18. **UPDATE:** Execute the target update procedure;
19. **RECOVERY:** Execute the target recovery procedure;
20. **TICK:** Execute the TICK event procedure;
21. **waitDIFS:** Execute the waitDIFS event procedure;
22. **Backoff:** Execute the Backoff event procedure;
23. **TX:** Execute the TX event procedure;
24. **RX:** Execute the RX event procedure;
25. **waitACK:** Execute the waitACK event procedure;
26. **end switch**
27. **end while**
28. Write the results into the text files;
29. Initialize all the statistic counters and simulation parameters;
30. Change the node deployment;
31. **if** (number of runs is not over) **do**:
32. go to step 4;
33. finish;

Figure 90 Target Tracking Handling within the Simulator

In line 3, the text files that store the results are created using the writer variable. In line 6, the sensor nodes are created as objects from the class that encapsulates the node attributes. These objects are then stored in the sensor nodes variable that is declared as a vector. In line 7, sensor nodes are deployed randomly in the sensing area. The world

variable is a 2D array that stores the node ID. Therefore, the ID for each node is randomly stored in the world array. From line 13 to 25, the procedure for different events is executed. In Section A.3, the pseudo code for these events is introduced.

A.2.2 Task Mapping and Scheduling Models

TMS models are developed using the same principle used to simulate the target tracking schemes, which is presented in Section A.2.1. Therefore, the CSMA/CA events shown in Figure 90 are also used to evaluate the BTMS and BITA algorithms. More events are added to code the proposed TMS algorithms. The application DAG is created according to the strategy introduced in Section 7.6.3. The s_T node defined in Sections 5.2.2 and 5.3 is represented as the target and its neighbours represented as the group nodes.

Basically, each group node will have routing information (i.e. single-hop routes) about its neighbours. However, multi-hop routing paths between nodes that are not in the same radio range are determined. Destination Sequenced Distance Vector Routing (DSDV) has been implemented in this model as multi-hop network layer protocol. DSDV is a proactive routing protocol, in which each node maintains a routing table for the whole network topology. However, in this thesis, DSDV only runs inside the group of nodes that detect the target. The node uses the Route Discovery (*RDis*) unicast messages to exchange the routing table information. At the same time as sending an *RDis* packet, the node sets its *Routing_timer* which is used to indicate finishing of the *RDis* packet transmission and thus to start transmission the next control packets. The *Routing_timer* is chosen to be 200ms. The node only sends the routing table information to the neighbours that do not have this information. Each node stores information about the source of the routing packets. For example, for single hop-communication, *RDis* packets will not be sent by any nodes because all nodes have routing information about each other. Another example is shown in Figure 91. Both node 1 and 3 have information about node 2 from the *RDis* message sent by node 2. Node 2 has information about both node 1 and 3 from the *RDis* messages sent by them. Therefore, node 1 and 3 will not have any routing information to be sent to node 2. On the other hand, node 2 will send *RDis* to both node 1 and 3 to inform them about each other. Since the communication could be multi-hop ad hoc links, the *RDis* message is sent three times to ensure the group nodes have a full routing table about all the neighbours which detect the same target.

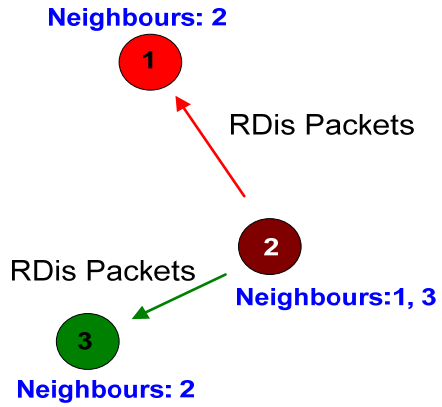


Figure 91 Sending *RDis* Packets

Therefore, the algorithm shown in Figure 92 is mainly used when the node sends the *RDis* packets to each neighbour.

```
1. for (node routing table) do:     //loop the node routing table
2.    if (the record is not in my neighbour routing table) do:
3.       send this record to my neighbour;
4.    end if
5. end for
```

Figure 92 Routing Algorithm 1

The *RDis* message mainly contains the routing table records of the available group nodes paths. When a node receives an *RDis* packet, it updates its routing table based on the shortest path (i.e. the minimum number of hops to the destination). For each routing record in the *RDis* message, the node runs the algorithm shown in Figure 93.

```
1. Route_Flag = true;
2. for (node routing table) do:      //loop the node routing table
3.    if (the received route record is in my routing table) do:
4.       Route_Flag = false;
5.       if (number of hops in the received record < number of  hops in the routing table) do:
6.          Delete the old route from the routing table;
7.          Add the new shorter route to my routing table;
8.          Add the nodes which have this record;
9.       end if
10.   end if
11.   if (Route_Flag) do:    //The node does not have this record
12.      Add the received routing record to the node routing  table;
13.   end if
14. end for
```

Figure 93 Routing Algorithm 2

A.3 Event Handling Pseudo Code

In this section, the pseudo code associated with events to perform the CSMA/CA as MAC protocol and the target tracking is introduced. The graph of CSMA/CA events is shown in Figure 48.

A.3.1 TArrive Event

In the target arrival (TArrive) event, the initial real state (i.e., location and velocity in 2D coordinates) of the target is declared. After that, the localization (LOCALIZATION) event is scheduled at time of simulation time plus the required time to perform the calculation of the target localization.

A.3.2 LOCALIZATION Event

The target localization (LOCALIZATION) event pseudo code is shown in Figure 94. In line 3, the sensor nodes that are in active mode cooperate to calculate the updated target state. In this case, the updated covariance matrix is initialized to a predefined value.

```
1. Calculate the energy consumption required to localize the target;
2. Update the total energy consumption counter;
3. Localize the target updated state and initiate the updated covariance matrix;
4. Set the sampling interval to its minimum value;
5. Schedule the PREDICTION  event at sTime plus the required time to execute EKF;
     prediction stage and insert it in the event list variable;
6. Schedule the NextSnapshot event at sTime plus current sampling interval;
```

Figure 94 LOCALIZATION Event

A.3.3 PREDICTION Event

In Figure 95, the pseudo code associated with the PREDICTION event is shown. In line 4, the current main node selects the next tracking group according to the algorithm described in Section 3.9.2. The group size is adjusted so that the updated tracking error is satisfied based on the adaptive group size algorithm presented in Section 3.9.3. In line 5, the main node is elected according the election algorithm described in Section 3.10. In line 6, the current main node schedules Ready event to wakeup the next selected group via a *GTrig* message.

```
1. Calculate the target predicted state and its covariance matrix using EKF;
2. Calculate the energy consumption required to predict the next target state;
3. Update the total energy consumption counter;
4. Select the next tasking group nodes so that the updated tracking accuracy is satisfied;
5. Elect one sensor node as MN;
6. Schedule Ready event immediately to send GTrig message;
```

Figure 95 PREDICTION Event

A.3.4 NextSnapshot Event

In Figure 96, the pseudo code associated with the NextSnapshot event is shown. The NextSnapshot is the event at which the target arrives to the next tracking step. In line 2, the real state of the target is computed. The target real state evolves according to the current target motion types, which could be uniform motion in straight line, circular motion in circles and curvature motion in curves. In line 3 to 7, each helper node initiates a *TRan* message transmission if it detects the target.

```
1. Schedule the RECOVERY event at sTime plus Timer_recovery;
2. Compute the current real state of the target;
3. for all helper nodes of the current group do:
4.    if (target is in the vicinity of the node) do:
5.       At time sTime plus the required time for sensing, schedule on the sensor node the
         Ready event to send TRan message to the main node;
6.    end if;
7. end for;
```

Figure 96 NextSnapshot Event

A.3.5 Ready Event

As shown in Figure 97, the Ready event is used to initiate messages transmissions. Each node is modelled so that it has output (qOut) and input (qIn) buffers. The node drops the message if its output buffer is full. If the output buffer is not empty, the node adds the message to queue of the output buffer to be served after finishing the leading message in the queue. Otherwise, the node defines a back off (bo) number which is a random number generated from uniform distributed over [0, CW], where CW is the predefined contention window. After that, the node senses the channel. If the channel is free, the node schedules a waitDIFS event after DIFS seconds from the simulation time (sTime), where DIFS is predefined Distributed Inter-Frame Space. Otherwise, the node defers the transmission and adds the message to output buffer. Finally, after the predefined timer expires, the node schedules the next Ready event to initiate the next message.

```
1. Prepare the message to be sent;
2. if (qOut is full) do:
3.   drop the message;
4. else do: //else of if in line 2
5.   if ( qOut is empty ) do:
6.     bo = uniform distributed over(0, CW);
7.     if (Channel is free) do:
8.       schedule waitDIFS at t = sTime + DIFS;
9.     else do:                          //channel is busy
10.      add the message to the qOut;
11.      defer = true;
12. else do:  //else of if in line 5
13.    add the message to the qOut;
14.    schedule the next Ready at t = sTime + predefined Timer;
```

Figure 97 Ready Event

A.3.6 UPDATE Event

If the main node detects the target and receives *TRan* messages from all helper nodes, it schedules the UPDATE event at current sTime plus the required time to execute EKF update stage and the sampling interval calculation. Figure 98 shows the pseudo code associated with the UPDATE event. In line 2, the sampling interval is calculated from the past target location using the algorithm introduced in Section 3.8.

```
1. Calculate the target updated state and its covariance matrix using EKF;
2. calculate the next sampling interval;
3. Calculate the energy consumption required to update the target state and calculate the
   sampling interval;
4. Update the total energy consumption counter;
5. Schedule the PREDICTION  event at sTime plus the required time to execute EKF;
   prediction stage and insert it in the event list variable;
6. Schedule the NextSnapshot event at sTime plus current sampling interval;
```

Figure 98 UPDATE Event

A.3.7 RECOVERY Event

If the one of the tasking group nodes does not detect the target, the main node sends the *TLos* message to the old main node. The old main performs the first level recovery. The pseudo code associated with the RECOVERY event is shown in Figure 99. In line 1 and 11, the first and second level recovery nodes are calculated according to the mechanism presented in Section 3.11.

```
1. Calculate the first level recovery nodes;
2. Calculate the energy consumption required for first level recovery;
3. Update the total energy consumption counter;
4. Schedule Ready event to send TRec message at sTime plus the required time to perform
   the first level recovery;
5. if (number of response > group size) do:
6.     Localize the target using the received target measurements;
7.     Set the sampling interval to its minimum value;
8.     Schedule the PREDICTION event at sTime plus the required time to execute EKF;
       prediction stage and insert it in the event list variable;
9.     Schedule the NextSnapshot event at sTime plus current sampling interval;
10. else do:
11.    perform level 2 recovery;
```

Figure 99 RECOVERY Event

A.3.8 Wait DIFS (waitDIFS) Event

In the waitDIFS event shown in Figure 100, if the channel was free during the last DIFS, the node schedules a BackOff event after one predefined time slot. Otherwise, if the channel is now free, the node finishes the deferring and schedules waitDIFS after DIFS time.

```
1. if (Channel was free during the last DIFS) do:
2.     schedule BackOff at t = sTime + Slot Time;
3. else do:
4.     if (Channel is free) do:
5.         defer = false;                    // stop the defer
6.         schedule waitDIFS at t = sTime + DIFS;
7.     else do:
8.         defer = true;
```

Figure 100 waitDIFS Event

A.3.9 Back off (Backoff) Event

In this event, which is explained in Figure 101, if the channel was free during the last time slot, the node checks the back off (bo) value. If bo is over and the channel is now free, the node schedules a TX event immediately. If bo is over and the channel is now busy, the node defers the transmission. The node decrements bo and schedules BackOff event after one predefined time slot, if bo is not over. If the channel was busy at any time during the last time slot and the channel is now busy, the node defers the transmission. If the channel was busy at any time during the last time slot and the channel is now free, the node schedules waitDIFS after DIFS time.

```
1. if (Channel was free during the last time slot) do:
2.    if (bo = 0) do:
3.       if (Channel is free) do:
4.          Schedule TX immediately;
5.       else do:  //else of if in line 3
6.          defer = true;
7.    else do:  //else of if in line 2
8.       Decrement bo;
9.       Schedule Backoff at t = sTime + time slot;
10. else do:  //channel became busy one time in the previous time slot
11.   if (Channel is free) do:          // currently the channel is free
12.      schedule waitDIFS at t = sTime + DIFS;
13.   else do:                  // currently the channel is busy
14.      defer = true;
```

Figure 101 Backoff Event

A.3.10 Transmit (TX) Event

With the TX event shown in Figure 102, the source node defers its transmission if there is acknowledgment (ACK) being sent from it. ACK transmission is initiated after receiving a unicast message. The ACK message will be ignored if the source node is sending other messages at the same time. The simulator determines the source node's neighbours. After that the channel status of the source node and its neighbours are set to the busy state. The message transmission times to the neighbours are calculated according to the following equation:

$$t_{RX} = t_s + t_t + t_p \tag{A.1}$$

where are t_s and t_p are the simulation time and propagation time, and t_t is the transmission time which can be calculated based on the following equation:

$$t_t = \text{Message Size}/\text{Channel Speed} \tag{A.2}$$

Then, the simulator schedules receive (RX) event. Finally, the source node schedules waitACK event, if the message is unicast.

```
1. if (node is sending ACK) do:
2.    defer = true;      //do not send this packet and defer
3. else do:
4.    if (message to be sent is ACK & the node is busy in sending other packets) do:
6.       do not send the ACK;
7.    else do:
8.       Determine the neighbour nodes;
9.       Set Channel flag to busy state in the node and the node's neighbours;
10.      Schedule RX at tRX;
11. if (message is unicast) do:
12.   Schedule waitACK at t = t(RX) + ACK Timeout;
```

Figure 102 TX Event

A.3.11 Collision (waitACK) Event

As shown in Figure 103 the source node initiates retransmissions of unicast messages that are not acknowledged. The maximum number of retransmissions is set in the node. The source node discards the message if the maximum permitted number of retransmissions is exceeded. For each retransmission, the CW is extended until it reaches to maximum value (CWmax). After that, the source node starts the transmission procedures by scheduling waitDIFS event.

```
1. if (Number Of Retransmissions is not over) do:
2.    Increment Number Of Retransmissions;
3.    CW = 2*CW;
4.    if (CW > CWmax) do:
5.       CW = CWmax;
6.       bo = uniform distributed over(0, CW);
7.       Schedule waitDIFS at t = sTime + DIFS;
8. else do:
9.    message dropped;
10.   send the next message in the qOut;
```

Figure 103 waitACK Event

A.3.12 Receive (RX) Event

At the RX event shown in Figure 104, the simulator sets the channel status to free for the source node and its neighbours. The source node and its neighbours end the deferring action, if any. Each source node's neighbour ignores the message if there are full or partial collisions. Otherwise, it processes the broadcast message and acknowledges the unicast message. The source node's neighbour either processes the unicast message if it is the final destination or redirects the message to the next hop.

```
1. Set channel status for free state in the node and its neighbours;
2. Cancel node and its neighbours defer if any;
3. for (all node neighbours) do:
4.    if (no full or partial collisions) do:
5.       if (message is unicast) do:
6.          Send ACK after SIFS time;
7.          if (The node is the final destination) do:
8.             Process the message;
9.          else do:
10.            Update message header and redirect it to the next hop;
11.      else do:    //the message is broadcast
12.         Process the message;
13.   else do:  // message is collided
14.      Ignore the message in the collided receivers;
```

Figure 104 RX Event

A.3.13 TICK Event

This is an interruptible event that runs periodically. The TICK event runs in the simulator every given number of seconds (e.g. once each second). Some simulation results are reported when the TICK event occurs. In the TICK event subroutine, the statistical counters are updated.

A.4 Appendix Summary

In this appendix, the simulation framework is described in detail. The target tracking, task mapping and scheduling implementations are first introduced. After that, the pseudo code for individual event handling is explained.

Appendix B Code Verification

B.1 Target Tracking Verification

In this section, target-tracking schemes proposed in Chapter 3 and 4 are verified. The target-tracking scenario presented in Section 7.3.5 is assumed. The snapshot at time of 9.6998 second is verified in this section. Figure 105 shows the target real location and the group nodes at time of 9.6998 second.

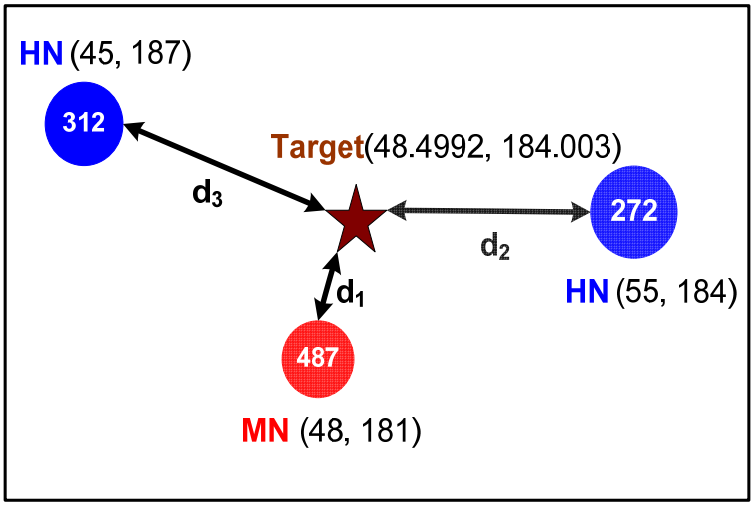


Figure 105 Target Tracking Snapshot at Time= 9.6998 seconds

B.1.1 Analytical Analysis

As shown in the Figure 105, nodes 487, 272 and 312 can detect the target because $d_1 = \sqrt{(0.5)^2 + (3)^2} = 3m$, $d_2 = \sqrt{(6.5)^2 + (0.003)^2} = 6.5m$ and $d_3 = \sqrt{(3.5)^2 + (2.9)^2} = 4.6m$ are less than the sensing range (i.e., 50m). When the MN (i.e., 487) detects the target, it measures its range and schedules the recovery process after a *Timer_recovery* of 0.05 second. Therefore, the recovery event is scheduled at time = 9.6998 + 0.05 = 9.7498 second. The HNs (i.e., 272 and 312) measure the target range and send it to the MN through *TRang* message. The HNs contend to transmit the *TRang* message as shown in Figure 106. All the nodes will wait DIFS = 128 ms and check the channel. If the channel is free, each node picks up a random number from the simulator RNG to set the back off times. The random back off values for nodes 272, and 312 are 40, 207 time slots (TS) respectively.

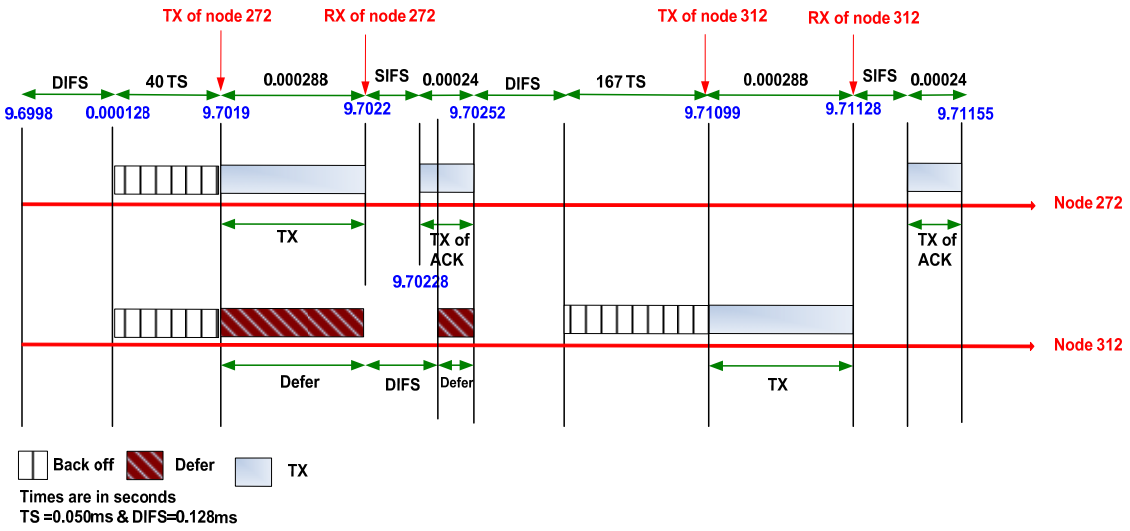


Figure 106 CSMA/CA Contention

The following equations are used to calculate the reception time after which the MN receives the *TRang* packet:

$$\text{Transition Time}(t_{TX}) = \text{Packet Size/Channel Bit Rate} \quad \text{(B.1)}$$

$$\text{Reception Time}(t_{RX}) = \text{Simulation Time} + \text{Transition Time} + \text{Propagation Delay} \quad \text{(B.2)}$$

Therefore, transmission time for *TRang* packet is $t_{TX} = 288/(1*10^6) = 0.288$ ms and for ACK is $t_{TX} = 240/(1*106) = 0.240$ ms. The propagation delay is neglected. Therefore, node 272 initiates its transmission at time = 9.6998 + DIFS + 40*TS = 9.70196 second and its reception at time = 9.70196 + 0.000288 = 9.70225 second. The wait for ACK event will be scheduled at time = 9.70196 (Time of transmission) + t_{TX} + 0.0003 (ACK timeout) = 9.70255 second. When the MN (i.e., 487) receives the *TRang* packet, it transmits ACK to the node 272 at time = 9.70225 (Time of reception) + 0.000028 (SIFS time) = 9.70228 second. The ACK will be received at node 272 at time = 9.70228 (Time of ACK transmission) + 0.000240 (Transition Time of ACK) = 9.70252 second, which is less than the ACK timeout.

Node 312 initiates its transmission at time = 9.70252 (Received Time of ACK at node 272 where the channel becomes free) + DIFS + (207-40)*TS = 9.71099 second and its reception at time = 9.71099 + 0.000288 = 9.71128 second. The wait for ACK event will be scheduled at time = 9.71099 (Time of transmission) + t_{TX} + 0.0003 (ACK timeout) = 9.71158 second. When the MN (i.e., 487) receives the *TRang* packet, it transmits ACK to the node 312 at time = 9.71128 (Time of reception) + 0.000028 (SIFS time) = 9.71131 second. The ACK will be received at node 312 at time = 9.71131

(Time of ACK transmission) + 0.000240 (Transition Time of ACK) = 9.71155 second, which is less than the ACK timeout.

After the MN receives the *TRang* packets from the HNs, it uses the EKF to calculate the target estimated state and the next sampling interval. These processes require 2MCC /100MHz=0.02 seconds. Therefore, the update stage finished at time = 9.71128 (time of second *TRang* reception) + 0.02 = 9.73128 second. The target estimated location at time 9.6998 second is (48.4992, 184.003). The target estimated location is used with the previous locations to calculate the next sampling interval. Table 11 shows the target location after inserting its current location. The data in Table 11 covers the target locations for more than 2 seconds history. Therefore, the oldest target location (i.e., index 1) will not be used in calculation.

Index	Time (sec)	x_T (m)	y_T (m)
1	7.72404	38.4992	166.683
2	8.22393	40.9992	171.013
3	8.72918	43.4992	175.343
4	9.22848	45.9992	179.673
5	9.73128	48.4992	184.003

Table 11 Target Estimated Locations Database

The net travel is the distance between the target location in index 2 and index 5 that are shown in Table 11 while the total travel is the sum of the distances between indices 2 and 3, 3 and 4, and 4 and 5. Therefore the net travel is 15 m, the total travel is 15 m and in turn the metadata is 15/15=1. The previous sampling interval was 0.5 seconds. Therefore, the measured sampling interval = (0.5 - 0.1) * 1 + 0.1 = 0.5 second and the sampling interval = 0.5 * 0.5 + 0.5 * 0.5 = 0.5 second. The next target snapshot will be scheduled at time = 9.6998 (the last time the target was detected) + 0.5 (sampling interval) = 10.1998 second.

After that prediction of the target next state and the group formation are performed. These processes require 2MCC /100MHz=0.02 seconds. Therefore, the prediction stage finished at time = 9.73128 (finish time of update stage) + 0.02 = 9.75128 second. The target estimated location at time 9.6998 second is (50.9992, 188.333). Table 12 summarizes the selected node's information.

Node	x_{S_i} (m)	y_{S_i} (m)	d_T (m)	$c = 1/d_T$ (m^{-1})	f_E
272	55	184	22.6058	0.0442364	0.367548
312	45	187	25.4403	0.0393077	0.326597
446	57	196	27.1655	0.0368114	0.305855

Table 12 Election Algorithm Information

In Table 12, d_T is the sum of the distances from other group nodes. $c=1/d_T$ is the node centrality. δ defined in Equation (3.41) is assumed to be 1. f_E is calculated based on Equation (3.41) as (c of the node)/summation of all c. Therefore, the MN for the next group is 272 because it has the maximum value of f_E.

The old MN (i.e., 487) sends the target information to the next group a long with the group election results by broadcasting a *GTrig* so that the new group has knowledge of the target before it arrives in their vicinity. The random back off value for nodes 487 is 121 time slots (TS) respectively. Therefore, node 272 initiates its transmission at time = 9.75128 (finish time of prediction stage) + DIFS + 121*TS = 9.75746 second and its reception at time = 9.75746 + 0.000288 = 9.75775 second.

B.1.2 Simulation Results

The simulation results of the scenario explained in Section B.1.1 are shown in this section through code output snapshots. Figure 107 shows the group nodes to track the target at time 9.6998 second. The target real location, the time of the recovery event and the HNs back off values are shown in Figure 107 as well.

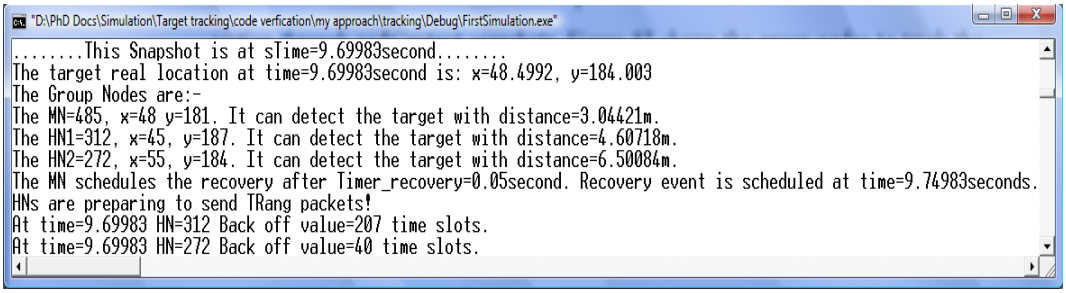


Figure 107 Target Tracking Snapshot at Time 9.6998 Seconds

Figure 108 and 109 shows the transmission and reception of *TRang* unicast (refer as message_type = 0) packets from the HNs 272 and 312 to the MN 485. The message serial numbers are used to ignore the received messages if they are sent twice due to the collision. The MN receives the *TRang* packet and other neighbours reject it. After that, the MN sends ACK message to inform the HNs about the correct reception of *TRang* messages.

Figure 110 shows the calculations of update stage, sampling interval, prediction stage and selection. The update stage that includes the sampling interval selection requires 0.02 seconds and the prediction stage that includes the formation of the next group needs 0.02 seconds as well.

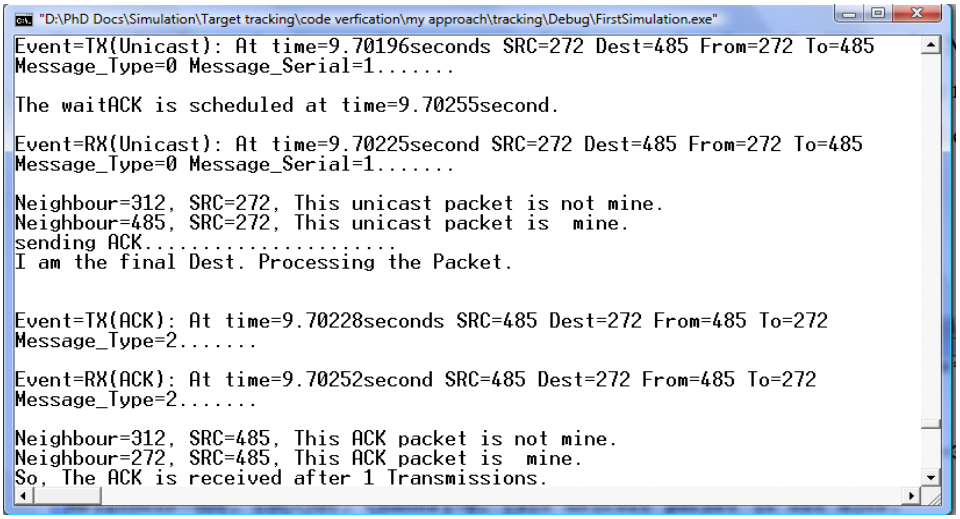


Figure 108 Simulation Snapshot: *TRang* Packet Transmission from HN1

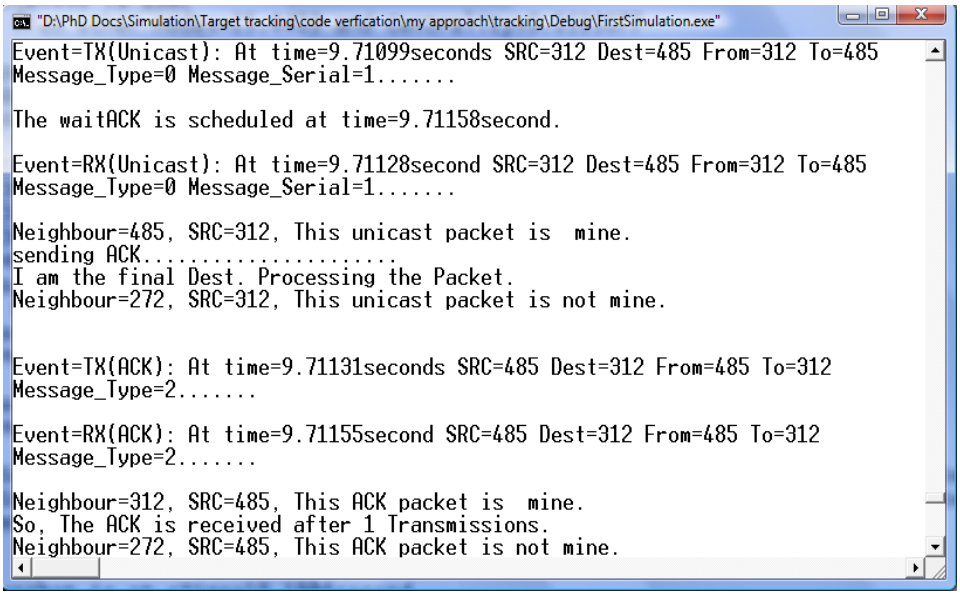


Figure 109 Simulation Snapshot: *TRang* Packet Transmission from HN2

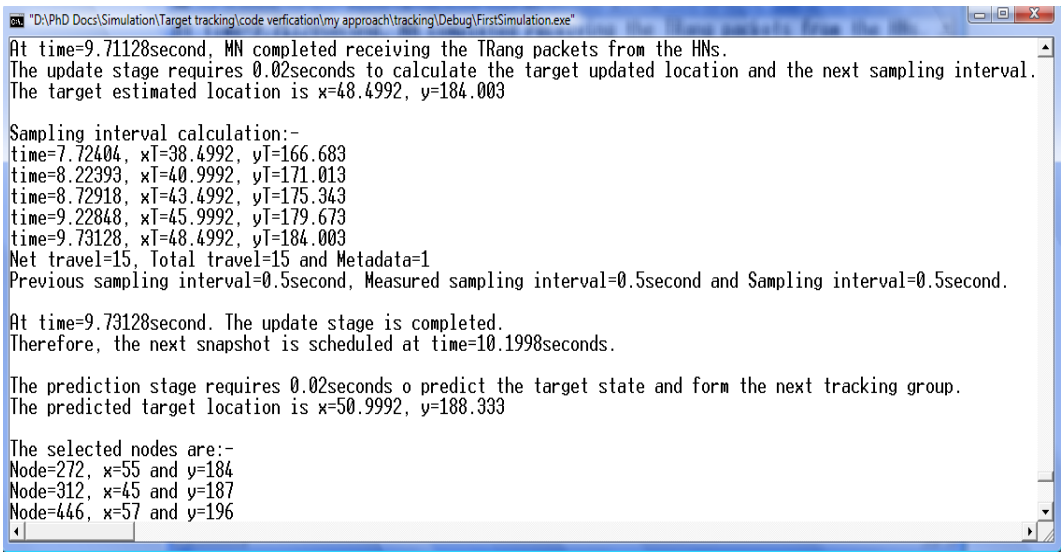


Figure 110 Update and Prediction Stages

In Figure 111, the election of the next MN is shown. Centrality for each node is calculated first. Then, the MN is selected so that it has the maximum election fitness function. In this case, node 272 is elected as the MN.

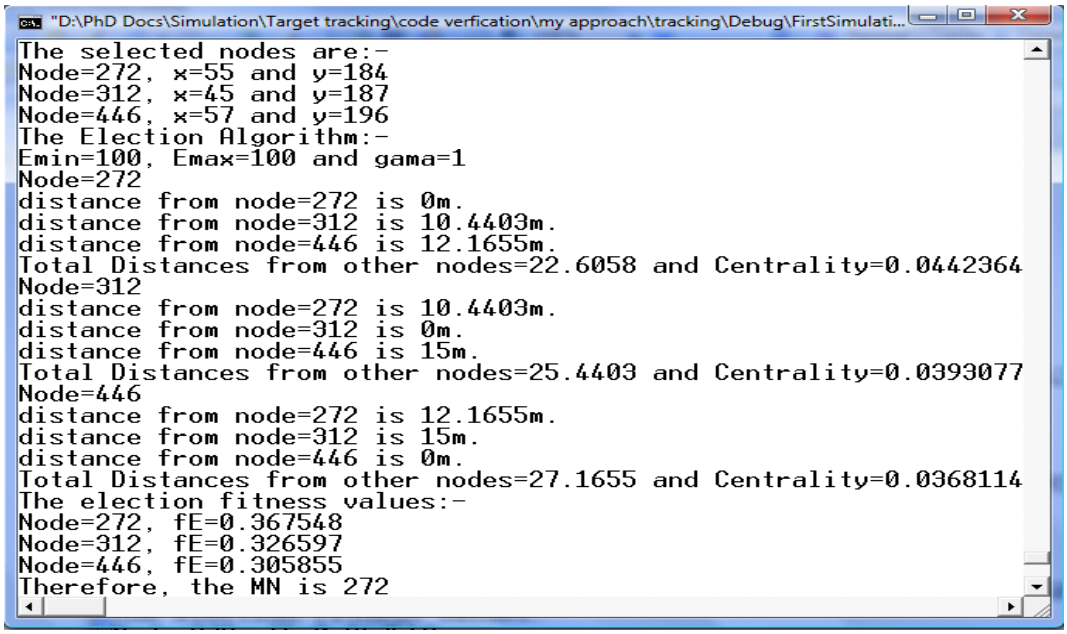


Figure 111 Election Algorithm

Finally, Figure 112 shows the time of transmitting and reception of the *GTrig* packet from the current MN to the next group. The destination is set to 0 to indicate that this message is broadcast packet.

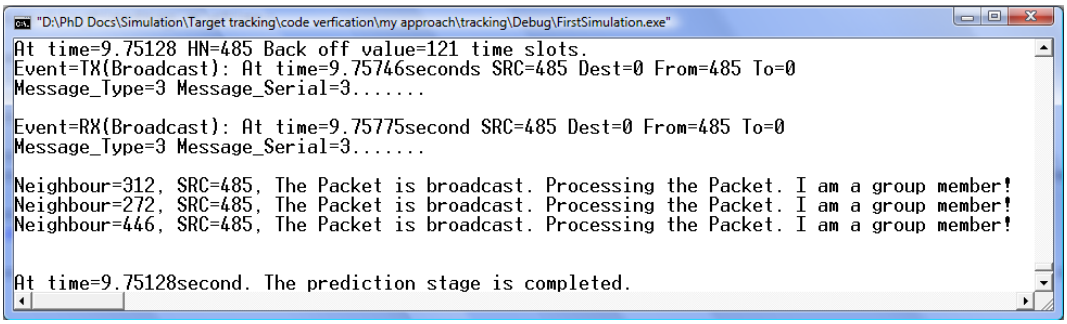


Figure 112 Transmission of *GTrig* Messages

The simulation snapshot shown in this section and the analysis presented in Section B.1.1 are both analytical matched. This indicates that the simulator is performed and run correctly.

B.1.3 Multi Target Tracking and the Optimal Solution

The simulation results in [103] show that the Tharmarasa's approach is close to the optimal solution. In Section 7.5.6, the simulation results show that the performance of the proposed MS-AMTT scheme is better than the Tharmarasa's approach in terms of computational time and number of iterations. Therefore, the proposed MS-AMTT

scheme is closer to the optimal solution compared with the Tharmarasa's approach given a limited number of iteration.

B.2 BITA Algorithm Verification

In this section, the BITA algorithm presented in Chapter 5 is verified. Assume the scenario shown in Figure 113 where the sensor node (s_T) is 57 and the S_n nodes that will share s_T in the execution of the application are 52 and 30. Assume the application can be divided into $N = 100$ independent equal-weighted tasks.

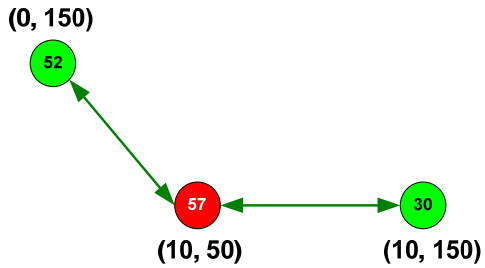


Figure 113 BITA Algorithm Verification

B.2.1 Analytical Analysis

The sensor node (s_T) performs the BITA algorithm. Assume the energy level for the three nodes is 100J. Based on Equation (5.10), for $Z = 0.02m^{-1}$, the influences of the nodes 52, 57 and 30 on the sensor node 57 are calculated as follows: $G(s_{52}, s_{57}) = 1/\sqrt{(50)^2 + (100)^2} = 0.00894427m^{-1}$, $G(s_{30}, s_{57}) = 1/50 = 0.02m^{-1}$ and $G(s_{57}, s_{57}) = Z = 0.02\,m^{-1}$. Based on Equation (5.11) and for $\beta = 0.5$, the decomposed fitness functions are calculated as follows:

$$f_D(s_{57}, 0.5, S_m) = 0.5 * (0.02/(0.02 + 0.02 + 0.00894427)) + 0.5 * (100/300) = 0.370981$$
$$f_D(s_{52}, 0.5, S_m) = 0.5 * (0.00894427/(0.02 + 0.00894427 + 0.02)) + 0.5 * (100/300) = 0.258039$$
$$f_D(s_{30}, 0.5, S_m) = 0.5 * (0.02/(0.02 + 0.02 + 0.00894427)) + 0.5 * (100/300) = 0.370981$$

Based on Equation (5.14), for $N = 100$ tasks, each group node will execute the following number of tasks:

$$n(s_{57}, 100) = 100 * 0.370981/(0.370981 + 0.258039 + 0.370981) = 37.0981$$
$$n(s_{52}, 100) = 100 * 0.258039/(0.370981 + 0.258039 + 0.370981) = 25.8039$$
$$n(s_{30}, 100) = 100 * 0.370981/(0.370981 + 0.258039 + 0.370981) = 37.0981$$

Assume, each node s_k can execute $f = 10$ tasks per second. Defines T_{s_k} as the execution finish time for the tasks allocated to the node s_k. Therefore, Cooperative Execution Time (CET) can be calculated according to the following equations:

$$CET = \max_{\forall k \in S_m}(T_{s_k}) \tag{B.3}$$

$$T_{s_k} = \frac{n(s_k, N)}{f} \tag{B.4}$$

Therefore, $\text{CET} = \max(37.0981/10,\ 25.8039/10, 37.0981/10) = 3.70981 \text{ sec}$. The same analytical analysis can be done for different N .

B.2.2 Simulation Results

Figure 114 shows the CET simulation results for the scenario in Section B.2.1 using different number of tasks N . Figure 115 shows more results detail for $N = 100$. Both analytical analysis and simulation results are matched.

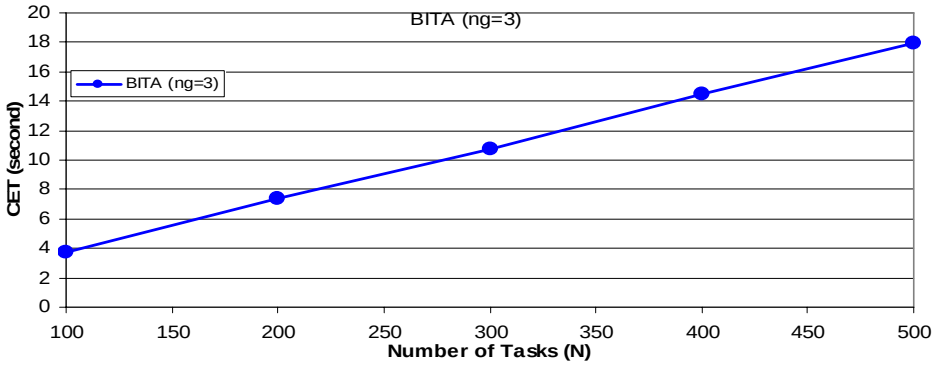


Figure 114 CET versus Number of Tasks Simulation Results

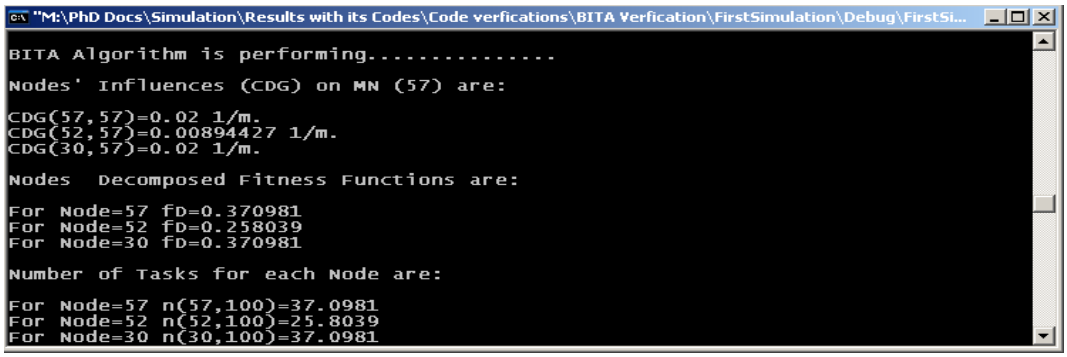


Figure 115 BITA: Simulation Results for N=100

B.3 BTMS Algorithm Verification

In this section, BTMS algorithm analytic analysis is compared with simulation results to verify the simulator. One simple scenario is verified in this section.

B.3.1 BTMS Algorithm and GA Algorithm

The simulation results in [122] shows that GA reduces the energy consumption among the sensor nodes by only 7% compared with Min-Min approach. The simulation results in Section 7.6.3 show that the proposed BTMS algorithm reduces the energy

consumption by 8% compared with BTMS algorithm that is based on Min-Min approach. Therefore, the performance of the proposed BTMS algorithm is close to the GA.

B.3.2 Analytical Analysis

The DAG application is generated as described in Section 7.6 with 3 entry tasks, 4 normal tasks and one exit task. Figure 116 shows one of the level-based DAGs which be used for the code verification. The numbers beside each task denote the number of clock cycles in Mega Clock Cycle (MCC). The numbers between the tasks edges represent the edge data size in Kilo bit (Kb) to be transmitted between tasks.

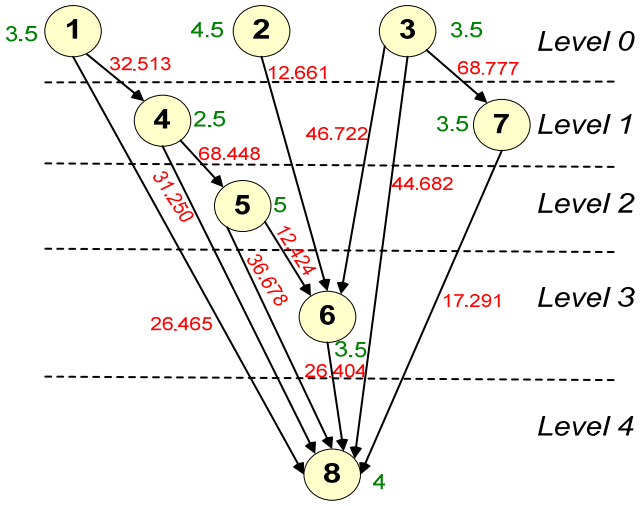


Figure 116 Level-Based DAG for Code Verification

After converting the DAG into level-based DAG as shown in Figure 116, the tasks in each level are ordered in decreasing manner with respect to their number of computational cycles. Therefore, the BTMS will begin to map task 2 at level 0. For all group nodes, it uses Equation (5.6) and (5.7) to calculate the required energy consumption and the finishing execution time of task 2. Then, using Equation (5.9), the fitness function of mapping task 2 to each node is calculated. Obviously, at this stage all the nodes will have the same fitness function. Therefore task 2 will map to one of the group nodes, which is node 47. This is shown in Figure 117.

The same procedures are applied to map task 3. All the nodes apart from node 47 will have the same fitness function. The availability of node 47 is greater than other nodes. Therefore, the fitness function of node 47 is greater than all other node. Task 3 will be mapped to one of the group nodes except node 47, which will be node 159. By using the same mechanism, task 1 will be mapped to node 318.

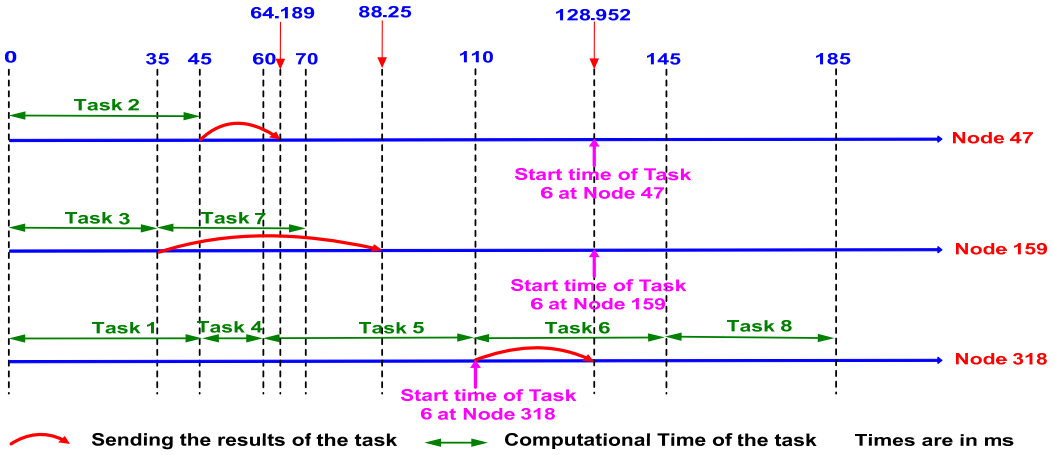


Figure 117 BTMS Analytical Analysis

At level 1, task 7 will be mapped first. Task 3, which is mapped to node 159, is the immediate predecessor of task 7. Therefore, node 159 will have the smallest fitness function defined and in turn, task 7 will be mapped to node 159. Like task 7 mapping strategy, task 4 at level 1 will be mapped to node 318 and task 5 at level 2 will be mapped to node 318.

At level 3, the fitness function of all nodes to map task 6 is calculated. Tasks 2, 3 and 5 are the immediate predecessors of task 6. Therefore, using Equation (4.24), the start execution time of task 6 at each node is calculated. After, that the finish executing time and the total energy consumption needed of task 6 at each node are calculated and passed to compute the fitness function. As shown in Figure 117, the start execution time of task 6 at node 318 is the smallest one and it will affect the calculation of the fitness function. Therefore, task 6 is mapped to node 318. Finally, task 8 is mapped to node 318. As shown in Figure 117, based on the analytical analysis, the CET will be 185ms.

B.3.3 Simulation Results

Figure 118 shows the results of tasks generator. It shows the task ID, computational clock cycles and immediate predecessors. The edge sized of the tasks dependencies are shown in bits. The output of level-based DAG algorithm is also shown.

BTMS algorithm starts to map each task. For each node, it computes the total energy consumption (i.e., computational and communication energy consumptions) required time to execute the task, the start and finish time of task's execution, and the fitness function. Figure 119 shows a snapshot of this process.

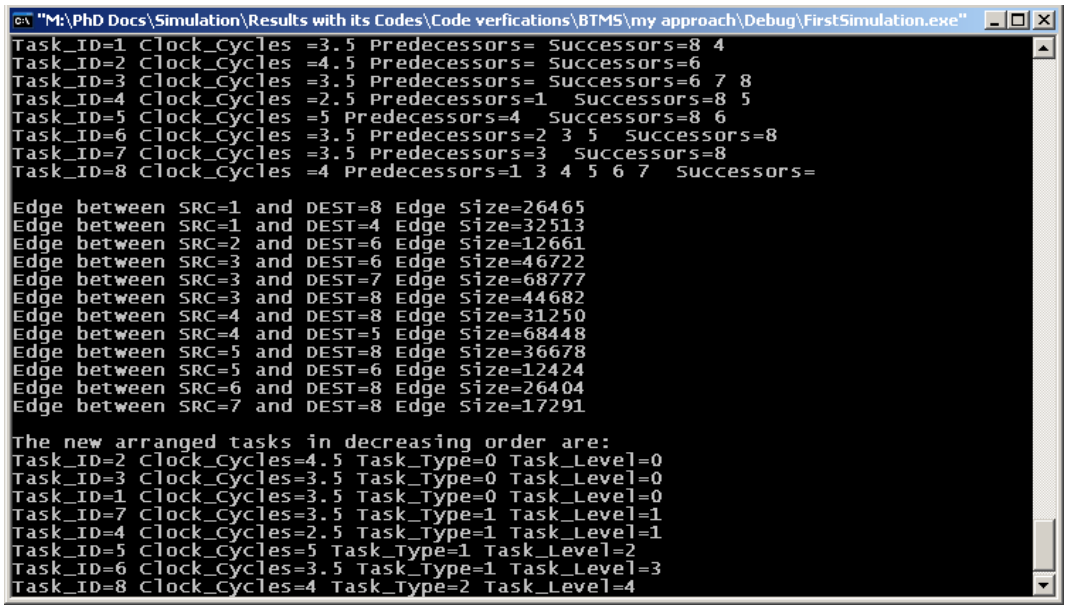


Figure 118 Task Generator Results

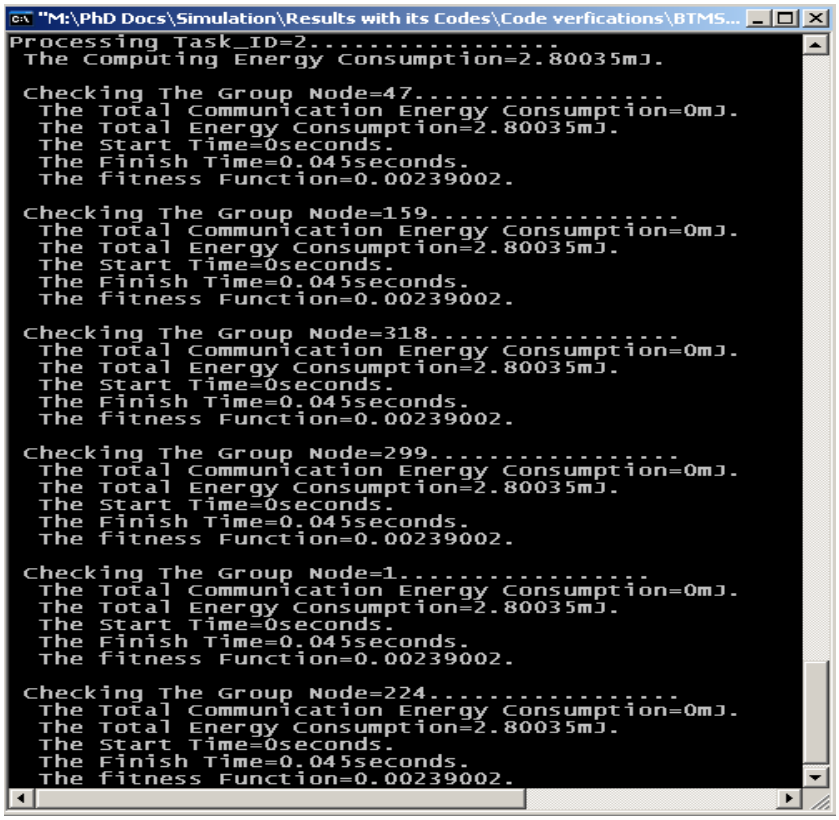


Figure 119 Decomposition Fitness Function Results

For each task, after calculating the fitness functions for all group nodes, the task will be mapped to the group node that has the smallest fitness function as shown in Figure 120.

Figure 121 shows the final results of BTMS algorithm. It shows the nodes that each task is mapped to and the start, execution and finish times of the task. The total energy consumption to execute the task is also shown. Finally, it shows the total overall communication and computing energy consumption to execute all the tasks. The

collaborative execution time (CET) of the application is shown I the bottom of Figure 121.

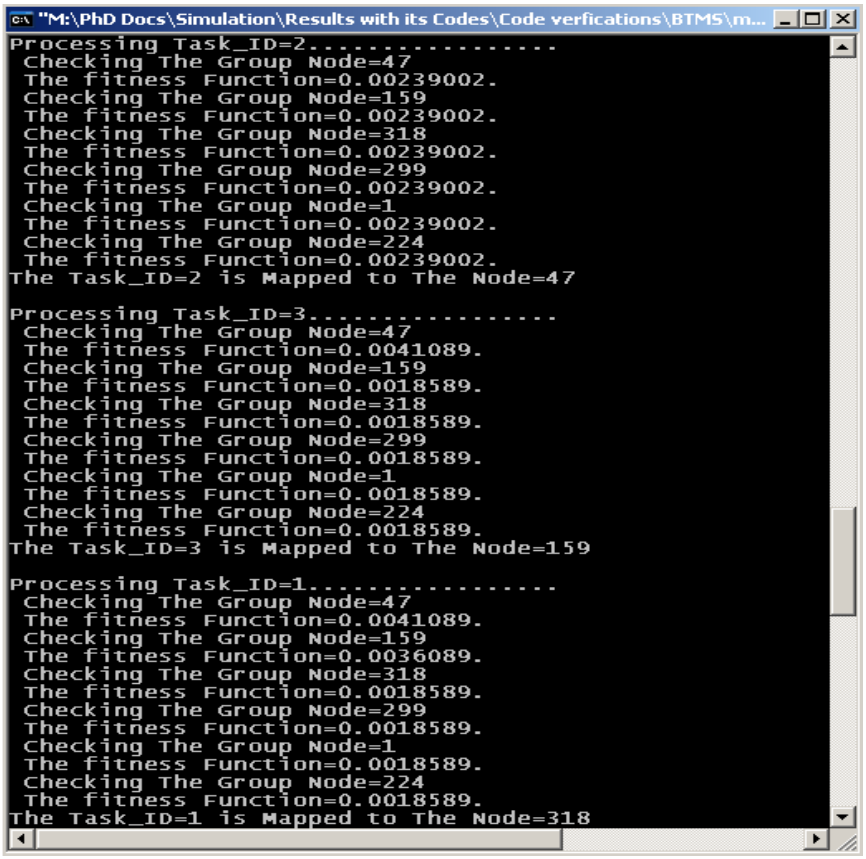


Figure 120 Task Mapping

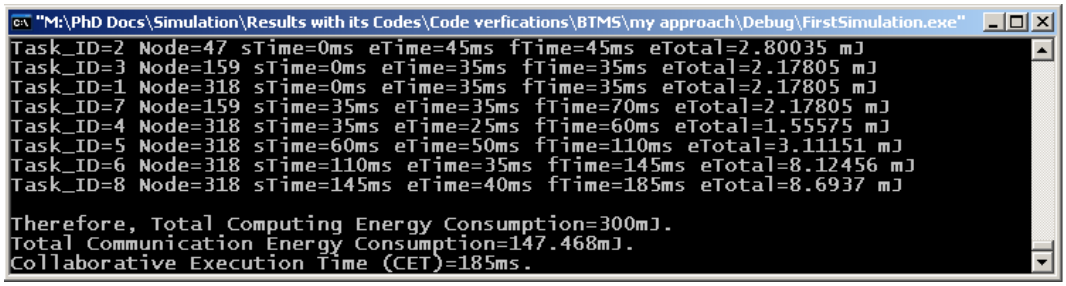


Figure 121 Summary of BTMS Results

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "*Wireless sensor networks: A survey*", Computer Networks: The International Journal of Computer and Telecommunications Networking, ACM, Vol. 38, No. 4, pp. 393-422, 2002.

[2] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, "*Resource management in energy-limited, bandwidth-limited, transceiver-limited wireless networks for session based multicasting*", Comput. Netw.: Int. J. Comput. Telecommun. Netw., Vol. 39, No. 5, pp. 113-131, 2002.

[3] V. Potdar, A. Sharif, E. Chang, "*Wireless Sensor Networks: A Survey*", 2009 IEEE International Conference on Advanced Information Networking and Applications Workshops, pp. 636-641, 2009.

[4] Th. Arampatzis, J. Lygeros and S. Manesis, "*A Survey of Applications of Wireless Sensors and Wireless Sensor Networks*", Proceedings of the IEEE 13th Mediterranean Conference on Control and Automation, pp. 719-724, 2005.

[5] J. Lin, W. Xiao, F.L. Lewis and L. Xie, "*Energy-Efficient Distributed Adaptive Multisensor Scheduling for Target Tracking in Wireless Sensor Networks*", IEEE Trans. Instrum. Meas., Vol. 58, No. 6, pp. 1886-1896, 2009.

[6] B. Rinner and W. Wolf, "*An Introduction to Distributed Smart Cameras*", Proceedings of the IEEE , Vol. 96, No. 10, pp.1565-1575, 2008.

[7] D. Fernandez-Baca, "*Allocating modules to processors in a distributed system*", IEEE Trans. Softw. Eng. , Vol. 15, No. 11, pp. 1427-1436, 1989.

[8] T. Hagras and J. Janecek, "*A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments*", Proceedings of IEEE International Conference on Parallel Processing Workshops (ICPPW'03), pp. 149–155, 2003.

[9] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "*Next Century Challenges: Scalable Coordination in Sensor Networks*", Proc. MobiCom, ACM, pp. 263-270, 1999.

[10] G.J. Pottie and W.J. Kaiser, "*Wireless Integrated Network Sensors*", Comm. ACM, Vol. 43, No. 5, pp. 51-58, 2000.

[11] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "*A Two-tier data dissemination model for large-scale wireless sensor networks*", Proceedings of the 8th annual international conference on Mobile computing and networking, ACM, pp. 148-159, 2002.

[12] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "*Collaborative Signal and Information Processing: An Information Directed Approach*", Proceedings of the IEEE, pp. 1199-1209, 2003.

[13] J.N. Al-Karaki and A.E. Kamal, "*Routing techniques in wireless sensor networks: a survey*", IEEE Wireless Communications, Vol. 11, No. 6, pp. 1536-1284, 2004.

[14] C.P. Singh, O.P. Vyas and M.K. Tiwari , "*A Profound Survey of Sensor Networks & Related Routing Protocols*", 4th IEEE International Conference on Wireless Communications, Networking and Mobile Computing 2008, pp. 1-5, 2008.

[15] *W. Hu, T. Tan, L. Wang, and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors",* IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., Vol. 34, No. 3, pp. 334-352, 2004.

[16] F. Zhao, J. Shin and J. Reich., "*Information-driven dynamic sensor collaboration for Tracking Applications*", IEEE Signal Process. Mag. , Vol. 19, No. 2, pp. 61-72, 2002.

[17] M. Ali, A. Bohm and M. Jonsson, "*Wireless Sensor Networks for Surveillance Applications? A Comparative Survey of MAC Protocols*", Proceedings of the fourth IEEE international conference on wireless and mobile communications (ICWMC 2008), pp. 399-403, 2008.

[18] W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "*Energy-Efficient Communication Protocol for Wireless Microsensor Networks*", Proceedings of the IEEE 33rd Annual Hawaii International Conference on System Sciences (HICSS '00), pp. 1-10, 2000.

[19] W. Ye and J. Heidemann, "*Medium Access Control in Wireless Sensor Networks*", USC/OSO Technical Report ISI-TR-580, 2003.

[20] I. Demirkol, C. Ersoy, and F. Alagoz, "*MAC Protocols for Wireless Sensor Networks: A Survey*", IEEE Commun. Mag., Vol. 44, No. 4, pp. 115-121, 2006.

[21] Mark Stemm and Randy H Katz, "*Measuring and reducing energy consumption of network interfaces in hand-held devices*", IEICE Transactions on Communications, Vol. E80-B, No. 8, pp. 1125-1131, 1997.

[22] T. S. Rappaport, "*Wireless Communications, Principles and Practice*", Prentice Hall, ISBN 0780311671,1996.

[23] W. Stallings, "*Wireless Communications and Networks*", 2nd Edition, Prentice Hall, ISBN 0131967908, 2005.

[24] N. Abramson, "*Development of the ALOHANET*", IEEE Trans. Inf. Theory, Vol. 31, No. 2, pp. 119-123, 1985.

[25] L. Kleinrock and F. Tobagi, "*Packet switching in radio channels: Part I-carrier sense multiple-access modes and their throughput-delay characteristics*", IEEE Trans. Commun., Vol. 23, No. 12, pp. 1400-1416, 1975.

[26] LAN MAN Standards Committee of the IEEE Computer Society, "*Wireless LAN medium access control (MAC) and physical layer (PHY) specification*", IEEE, IEEE Std 802.11, 1999 edition, 1999.

[27] A. Woo and D. Culler, "*A transmission control scheme for media access in sensor networks*", International Conference on Mobile Computing and Networking, Proceedings of the 7th annual international conference on Mobile computing and networking, ACM, pp. 221-235, 2001.

[28] F. Tobagi and L. Kleinrock, "*Packet switching in radio channels: Part II - the hidden terminal problem in carrier sense multiple access and the busy-tone solution*", EEE Trans. Commun., Vol. 23, No. 12, pp. 1417-1433, 1975.

[29] M.S. Gast, "*802.11 Wireless Networks: The Definition Guide*", O'Reilly & Associates, ISBN 0596001835, 2002.

[30] P. Karn, "*MACA: A new channel access method for packet radio*", In Proceedings of the 9th ARRL Computer Networking Conference, pp. 134-140, 1990.

[31] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "*MACAW: A media access protocol for wireless lans*", In Proceedings of the ACM SIGCOMM Conference, pp. 212-225, 1994.

[32] W. Ye, J. Heidemann, and D. Estrin, "*An energy-efficient mac protocol for wireless sensor networks*", Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1567-1576, 2002.

[33] W. Ye, J. Heidemann, and D. Estrin, "*Medium access control with coordinated, adaptive sleeping for wireless sensor networks*", IEEE/ACM Trans. Netw., Vol. 12, No. 3, pp. 493-506, 2004.

[34] J.F. Kurose and K.W. Ross, "*Computer Networking*,", second Edition, Addison Wesley, ISBN 0321213939, 2003.

[35] C.K. Toh, "*Ad Hoc Mobile Wireless Networks: Protocols and Systems*", Prentice Hall, ISBN 0130078174, 2002.

[36] C.S.R. Murthy and B.S. Manoj, "*Ad Hoc Wireless Networks: Architectures and Protocols*", Prentice Hall, ISBN 013147023X, 2004.

[37] N. Bulusu, J. Heidemann and D. Estrin, "*GPS-less low cost outdoor localization for very small devices*", Technical Report 00-729, Computer science department, University of Southern California, 2000.

[38] T.C. Karalar, S. Yamashita, M. Sheets and J. Rabaey, "*A low power localization architecture and system for wireless sensor networks*", IEEE Workshop on Signal Processing Systems, pp. 89-94, 2004.

[39] W. Heinzelman, J. Kulik, and H. Balakrishnan, "*Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*", 5th ACM/IEEE Mobicom Conference (MobiCom '99), pp. 174-85, 1999.

[40] J. Kulik, W. R. Heinzelman and H. Balakrishnan, "*Negotiation-based protocols for disseminating information in wireless sensor networks*", Wireless Networks, Selected Papers from Mobicom 1999, ACM, Vol. 8, No. 2/3, pp. 169-185, 2002.

[41] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "*A scalable and robust communication paradigm for sensor networks*", Proceedings of ACM MobiCom 2000, pp. 56-67, 2000.

[42] D. Braginsky and D. Estrin, "*Rumor Routing Algorithm for Sensor Networks*", in the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), ACM, pp. 22-31, 2002.

[43] F. Ye, A. Chen, S. Liu and L. Zhang, "*A scalable solution to minimum cost forwarding in large sensor networks*", Proceedings of the IEEE tenth International Conference on Computer Communications and Networks (ICCCN), pp. 304-309, 2001.

[44] M.X. Gong, S.F. Midkiff and R.M. Buehrer, "*A Self-Organized Clustering Algorithm for UWB Ad Hoc Networks*", IEEE Wireless Communications and Networking Conference 2004 (WCNC 2004), pp. 1806-1811, 2004.

[45] S. Lindsey and C. Raghavendra, "*PEGASIS: Power-Efficient Gathering in Sensor Information Systems*", IEEE Aerospace Conference Proceedings, pp. 1125-1130, 2002.

[46] A. Manjeshwar and D.P. Agarwal, "*TEEN: a routing protocol for enhanced efficiency in wireless sensor networks*", Proceedings in IEEE 15th International Symposium Parallel and Distributed Processing, pp. 2009-2015, 2001.

[47] A. Manjeshwar and D.P. Agarwal, "*APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks*", Proceedings in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 195-202, 2002.

[48] V. Rodoplu and T.H. Meng, "*Minimum Energy Mobile Wireless Networks*", IEEE J. Sel. Areas Commun., Vol. 17, No. 8 , pp. 1333-1344, 1999.

[49] Y. Xu, J. Heidemann and D. Estrin, "*Geography-informed Energy Conservation for Ad-hoc Routing*", In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 70-84, 2001.

[50] Y. Yu, D. Estrin, and R. Govindan, "*Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*", UCLA Computer Science Department Technical Report, Computer science department, University of Southern California, 2001.

[51] F. Kuhn, R. Wattenhofer and A. Zollinger, "*Worst-Case optimal and average-case efficient geometric ad-hoc routing*", Proceedings of the 4th ACM International Conference on Mobile Computing and Networking, pp. 267-278, 2003.

[52] C.E. Perkins and P. Bhagwat, "*Highly Dynamic Destination Sequence-Vector Routing (DSDV) for Mobile Computers*", ACM SIGCOMM Computer Communication Review, Vol. 24, No. 4, pp. 234-244, 1994.

[53] M. Valera and S.A.Velastin, "*Intelligent distributed surveillance systems: a review*", IEE Proceedings -Vision, Image and Signal Processing, Vol. 152, No. 2, pp. 192-204, 2005.

[54] T. Ko, "*a survey on behavior analysis in video surveillance for homeland security applications*", 37th IEEE Applied Imagery Pattern Recognition Workshop, 2008 (AIPR 2008), pp. 1-8, 2008.

[55] A.C. Sankaranarayanan, A. Veeraraghavan and R. Chellappa, "Object Detection, Tracking and Recognition for Multiple Smart Cameras", Proceedings of the IEEE, Vol. 96, No.10, pp.1606-1624, 2008.

[56] X. Sheng and Y. Hu, "*Energy Based Acoustic Source Localization*", Proceedings of the 2nd international conference on Information processing in sensor networks (IPSN), Springer-Verlag, pp. 285-300, 2003.

[57] Y.E.M. Hamouda and C. Phillips, "*Biologically Inspired, Cooperative Target Tracking Framework for Wireless Sensor Networks*", London Communication Symposium 2009 (LCS 2009), University College London, 2009.

[58] T. Clouqueur, K.K. Saluja and P. Ramanathan, "*Fault tolerance in collaborative sensor networks for target detection*", IEEE Trans. Comput., Vol. 53, No. 3, pp. 320-333, 2004.

[59] A. Discant, A. Rogozan, C. Rusu and A. Bensrhair, "*Sensors for Obstacle Detection - A Survey*", IEEE 30th International Spring Seminar on Electronics Technology, pp.100-105, 2007.

[60] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, M. Arumugam, M. Nesterenko and A. Vora, "*A line in the sand: a wireless sensor network for target detection, classification, and tracking*", ScienceDirect Computer Networks
Vol. 46, No. 5, , pp. 605-634, 2004.

[61] H. Zhou, M. Taj and A. Cavallaro, "*Target Detection and Tracking With Heterogeneous Sensors*", IEEE Trans. Sel. Topics in Signal Process., Vol. 2, No. 4, pp. 503-513, 2008.

[62] E.D. Manley, H.A. Nahas and J.S. Deogun, "*Localization and Tracking in Sensor Systems Sensor Networks*", IEEE International Conference on Ubiquitous, and Trustworthy Computing, pp. 237-242, 2006.

[63] R.R. Brooks, P. Ramanathan and A.M. Sayeed, "*Distributed target classification and tracking in sensor networks*", Proceedings of the IEEE , Vol. 91, No. 8, pp. 1163-1171, 2003.

[64] C. Rohrig and S. Spieker, "*Tracking of transport vehicles for warehouse management using a wireless sensor network*", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008 (IROS 2008), pp. 3260-3265, 2008.

[65] C. Schurgers, V. Tsiatsis and M.B. Srivastava, "*STEM: Topology Management for Energy-Efficient Sensor Networks*", Proceedings in IEEE Aerospace Conference, pp. 135-145, 2002.

[66] J.C. Chen, R.E. Hudson and K. Yao, "*Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field*", IEEE Trans. Signal Process. , Vol. 50, No. 8 , pp. 1843 -1854, 2002.

[67] K.M. Kaplan, Q. Le and P. Molnar, "*Maximum likelihood methods for bearings only target localization*", Proceedings in IEEE International Conference on Acoustics, Speech, and Signal Processing 2001 (ICASSP 2001), pp. 3001-3004, 2001.

[68] K.T. Soe, "Increasing Lifetime of Target Tracking Wireless Sensor Networks", Proceedings of World Academy of Science, Engineering and Technology, pp. 410-415, 2008.

[69] W.P. Chen, J.C. Hou and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks", IEEE Trans. Mobile Comput., Vol. 3, No. 3, pp. 258-281, 2004.

[70] M. Di, E.M. Joo and L.H. Beng, "*A Comprehensive Study of Kalman Filter and Extended Kalman Filter for Target Tracking in Wireless Sensor Networks*", 2008 IEEE International Conference on Systems, Man and Cybernetics (SMC 2008), pp. 2792-2797, 2008.

[71] S. Maskell and N. Gordon, "*A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking*", IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174) Workshop, pp. 2/1-2/15, 2001.

[72] B. Ristic, S. Arulampalam and N. Gordon, "*Beyond the Kalman Filter: Particle Filters for Tracking Applications*", Artech Print House, ISBN 158053631X, 2004.

[73] X.R. Li and V.P. Jilkov, "*A Survey of Maneuvering Target Tracking-Part III: Measurement Models*", In Proceedings SPIE Conference on Signal and Data Processing of Small Targets, 2001.

[74] Y. Bar-Shalom, X.R. Li, T. Kirubarajan, "*Estimation With Applications to Tracking and Navigation*", Wiley, ISBN 047141655X, 2001.

[75] F. L. Lewis, "*Optimal Estimation*", Wiley, ISBN 0471837415, 1986.

[76] S. Arulampalam and B. Ristic, "*Comparison of the particle filter with range parameterized and modified polar EKF's for angle-only tracking*", Proceedings of SPIE, Vol. 4048, pp. 288-299, 2000.

[77] A. Doucet, N.D. Freitas and N.J. Gordon, "*An introduction to sequential Monte Carlo methods*", in Sequential Monte Carlo Methods in Practice, Springer-Verlag, 2001.

[78] A. Doucet, S. Godsill and C. Andrieu, "*On sequential Monte Carlo sampling methods for Bayesian filtering*", Statist Dept. Eng., Univ. Cambridge, UK, Tech. Rep., 1998.

[79] P.M. Djuric, J.H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M.F. Bugallo, J. Miguez, "*Particle filtering*", IEEE Signal Process. Mag., Vol. 20 , No. 5, pp. 19- 38, 2003.

[80] D. Li, K.D. Wong, Y.H. Hu and A.M. Sayeed, "*Detection, classification and tracking of targets in distributed sensor networks*", IEEE Signal Process. Mag., Vol. 19, No. 2, pp. 17-29, 2002.

[81] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee and C.-F. Huang, "*Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies*", The Computer Journal, Vol. 47, No. 4, pp. 448-460, 2004.

[82] M. Chu, H. Haussecker, and F. Zhao, "*Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks*", Int'l J. of High-Performance Computing Applications, Vol. 16, No. 3, 2002.

[83] M.R. Nami and K. Bertels, "*A Survey of Autonomic Computing Systems*", Third International Conference on Autonomic and Autonomous Systems, 2007 (ICAS07), pp. 26-26, 2007.

[84] X. Gu, J. Strassner, J. Xie, L.C. Wolf and T. Suda, "*Autonomic Multimedia Communications: Where Are We Now?*", Proceedings of the IEEE, Vol. 96, No. 1, pp.143-154, 2008.

[85] T. Nakano and T. Suda, "*Self-organizing network services with evolutionary adaptation*", IEEE Trans. Neural Netw., Vol. 16, No. 5, pp. 1269-1278, 2005.

[86] T. Nakano and T. Suda, "*Adaptive and evolvable network services*", Proceedings in Genetic and Evolutionary Computation Conference, pp. 151-162, 2004.

[87] T. Suda, T. Itao and M. Matsuo, "*The bio-networking architecture: The biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications*", Proceedings in 2001 Symposium on Applications and the Internet, pp. 43-53, 2001.

[88] S. Balasubramaniam, D. Botvich, W. Donnelly, M.O. Foghlu and J. Strassner, "*Biologically inspired self-governance and self-organisation for autonomic networks*", In ACM Proceedings of the 1st international Conference on Bio inspired Models of Network, information and Computing Systems, Vol. 275, 2006.

[89] P. Champrasert and J. Suzuki, "*Towards Self-Adaptive Networking with Symbiotic Behaviours of Multi-Agents*", IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems 2007 (KIMAS 2007), pp. 103-108, 2007.

[90] J. Moon and J. Nang, "*Design and Implementation of a bio-inspired system platform*", TENCON 2007 - 2007 IEEE Region 10 Conference, pp. 1-4, 2007.

[91] Y. Meng, O. Kazeem and J.C. Muller, "*A Swarm Intelligence Based Coordination Algorithm for Distributed Multi-Agent Systems*", Proceedings in IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp. 294-299, 2007.

[92] L. Song and D. Hatzinakos, "*A Cross-Layer Architecture of Wireless Sensor Networks for Target Tracking*", IEEE/ACM Trans. Netw., Vol. 15, No. 1, pp. 145-158, 2007.

[93] H. Yang and B. Sikdar, "*A protocol for tracking mobile targets using sensor networks*", Proceedings of the First IEEE International Workshop on Sensor Networks Protocols and Applications, pp. 71-81, 2003

[94] H. Yang, B. Sikdar, "*Lightweight target tracking protocol using ad-hoc sensor network*", 2005 IEEE 61st Vehicular Technology Conference, pp. 2850-2854, 2005.

[95] V.P. Sadaphal and B.N. Jain,"*Tracking mobile target using selected sensors*", First International IEEE Communication Systems and Networks and Workshops 2009 (COMSNETS 2009), pp. 1-10, 2009.

[96] H. Wang, K. Yao, G. Pottie and D. Estrin, "*Entropy-based sensor selection heuristic for target localization*", In Proceedings Of 3rd International Symposium on Information Processing in Sensor Networks, pp. 36-45, 2004.

[97] T. Onel, C. Ersoy and H. Delic, "*Information Content-Based Sensor Selection and Transmission Power Adjustment for Collaborative Target Tracking*", IEEE Trans. Mobile Comput., Vol. 8, No. 8, pp. 1103-1116 , 2009.

[98] Y. Han and W. Zhao, "*A Novel Node Selection Method of Bearings-only Sensors for Target Tracking in Wireless Sensor Networks*", IEEE International Conference on Communications and Mobile Computing 2009 (CMC 2009), pp.136-140, 2009.

[99] C. Hue, J-P.L. Cadre and P. Perez, "*Posterior Cramer-Rao Bounds for Multi Target Tracking*", IEEE Trans. Aerosp. Electron. Syst., Vol. 42, No. 1, pp. 37-49., 2006.

[100] S. Oh, S. Russel and S. Sastry, "*Markov Chain Carlo Data Association for Multi-Target Tracking*", Trans, Autom. Control, Vol. 54, No. 3, pp. 481-497, 2009.

[101] J. Vermaak, S.J. Godsill and P. Perez, "*Monte Carlo Filtering for Multi-Target Tracking and Data Association*", IEEE Trans. Aerosp. Electron. Syst., Vol. 41, No. 1, pp. 309-331, 2005.

[102] A. Oka and L. Lampe, "*Distributed Scalable Multi-Target Tracking with a Wireless Sensor Network*", Proceedings in IEEE International Conference on Communication, pp. 1-6, 2009.

[103] R. Tharmarasa, T. Kirubarajan and M. I. Hernandez, "*Large-Scale Optimal Sensor Array Management for Multitarget Tracking*", IEEE Trans. Syst., Vol. 37, No. 5, pp. 803-814, 2007.

[104] R. Tharmarasa, T. Kirubarajan, J. Peng and T. Lang "*Optimization-Based Dynamic Sensor Management for Distributed Multitarget Tracking*", IEEE Trans. Syst., Vol. 39, No. 5, pp. 534-546, 2009.

[105] L. Liu, X. Zhang and H. Ma, "*Dynamic Node Collaboration for Mobile Target Tracking in Wireless Camera Sensor Networks*", IEEE INFOCOM 2009, pp. 1188-1196, 2009.

[106] H. Xue, B. Chen and J. Wan, "*A Distributed Target Tracking Algorithm Based on Asynchronous Wireless Sensor Networks*", IEEE International Conference on Electronic Computer Technology, pp.549-553, 2009.

[107] S. Pino-Povedano and F.-J. Gonzalez Serrano, "*Distributed tracking and classification of targets with sensor networks*", 16th IEEE International Conference on Software, Telecommunications and Computer Networks 2008 (SoftCOM 2008), pp.213-217, 2008.

[108] S. Aeron, V. Saligrama and D.A. Castaon, "*Efficient Sensor Management Policies for Distributed Target Tracking in Multihop Sensor Networks*", IEEE Trans. Signal Process., Vol. 56, No. 6, pp. 2562-2574, 2008.

[109] H.J. Rad, M. Azarafrooz, H.S. Shahhoseini, and B. Abolhassani, "*A new adaptive power optimization scheme for target tracking Wireless Sensor Networks*", 2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009), pp. 307-312, 2009.

[110] W.Xiao, W. Zhang, S., J. Lin and C.K. Tham, "*Energy-efficient adaptive sensor scheduling for target tracking in wireless sensor networks*", Journal of Control Theory and Applications, Vol. 8, No. 1, pp. 86-927, 2010.

[111] T.D. Braun, H.J. Siegal, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, D. Hensgen and R.F. Freund , "*A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems*", Proceedings in eighth IEEE Heterogeneous Computing Workshop 1999. (HCW 1999), pp. 15-29, 1999.

[112] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "*A*

comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp. 810-837, 2001.

[113] O. H. Ibarra and C. E. Kim, "*Heuristic algorithms for scheduling independent tasks on non-identical processors*", Journal of the ACM, Vol. 24, No. 2, pp. 280-289, 1977.

[114] M. A. Iverson, F. Ozguner, and G. J. Follen, "*Parallelizing existing applications in a distributed heterogeneous environment*", 4th IEEE Heterogeneous Computing Workshop (HCW 1995), pp. 93-100, 1995.

[115] J.A. Gonzalez, "*A Hyper-heuristic for scheduling independent jobs in Computational Grids*", Accessed in 22/08/2009, Available At URL: http://www-sop.inria.fr/mascotte/WorkshopScheduling/Slides/Gonzalez.pdf.

[116] R. Armstrong, D. Hensgen and T. Kidd, "*The relative performance of various mapping algorithm is independent of sizable variance in run-time predictions*", Proceedings of ACM 7th Heterogeneous Computing Workshop (HCW 1998), pp. 79-87, 1998.

[117] R. Freund, T. kidd, D. Hensgen, L. Moore, "*SmartNet: a scheduling framework for heterogeneous computing*", Proceedings on The IEEE Second International Symposium on Parallel Architectures, Algorithms, and Networks, pp. 514-521, 1996.

[118] A. Sudarsanam, M. Srinivasan and S. Panchanathan, "*Resource estimation and task scheduling for multithreaded reconfigurable architectures*", Proceedings of the IEEE tenth international conference on parallel and distributed system, pp. 323-330, 2004.

[119] M. Maheswaran and H.J. Siegel, "*A dynamic matching and scheduling algorithm for heterogeneous computing systems*", Proceedings in IEEE Seventh Heterogeneous Computing Workshop 1998. (HCW 1998), pp. 57-69, 1998.

[120] A.H. Alhusaini, V.K. Prasanna and C.S. Raghavendra "*A unified resource scheduling framework for heterogeneous computing environments*", Proceedings in IEEE 8th Heterogeneous Computing Workshop 1999. (HCW 1999), pp. 156-165, 1999.

[121] S. Giannecchini, M. Caccamo and C.-S. Shih, "*Collaborative resource allocation in wireless sensor networks*", Proceedings in IEEE 16th Euromicro Conference on Real-Time Systems, pp. 35-44, 2004.

[122] S. Shivle, R. Castain, H. J. Siegel, A.A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, W. Saylor, D. Sendek, J. Sousa, J. Sridharan, P. Sugavanam and J. Velazco, "*Static mapping of subtasks in a heterogeneous ad hoc grid environment*", In Proceedings of IEEE eighteenth International Parallel and Distributed Processing Symposium, pp. 110, 2004.

[123] M. Srinivas and L.M. Patnaik, "*Genetic algorithms: A survey*", IEEE Computer, Vol. 27, No. 6, pp. 17-26, 1994.

[124] Y. Yu and V.K. Prasanna, "*Energy-balanced task allocation for collaborative processing in wireless sensor networks*", ACM Mobile Networks and Applications, Vol. 10, No. 1-2, pp. 115-131, 2005.

[125] S. Poslad, "*Ubiquitous Computing: Smart Devices, Environments and Interactions*", John Wiley & Sons Ltd., ISBN 0470035609, 2009.

[126] Y. Tian, E. Ekici, and F. Ozguner, "*Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks*", Proceedings of IEEE International Workshop on Resource Provisioning and Management in Sensor Networks 2005 (RPMSN 2005), pp. 211-218, 2005.

[127] Y. Tian, B. Jarupan, E. Ekici, and F. Ozguner, "*Real Time Task Mapping and Scheduling for Collaborative In Network Processing in DVS-Enabled Wireless Sensor Networks*", Proceedings in IEEE International Parallel and Distributed Processing Symposium 2006 (IPDPS 2006), pp. 1-10, 2006.

[128] Y. Tian and E. Ekici, "*Cross-Layer Collaborative In Network Processing in Multihop Wireless Sensor Networks*", IEEE Trans. Mobile Comput., Vol. 6, No. 3, pp. 297-310, 2007.

[129] Y. Zhai, M. Yeary and J.-C. Noyer, "*Target Tracking In a Sensor Network Based on Particle Filtering and Power-Aware Design*", Proceedings of the IEEE Instrumentation and Measurement Technology Conference 2006 (IMTC 2006), pp. 1988-1992, 2006.

[130] O. Ozdemir, R. Niu and P.K. Varshney, "*Tracking in Wireless Sensor Networks Using Particle Filtering: Physical Layer Considerations*", IEEE Trans. Signal Process., Vol. 57, No. 5, pp. 1987-1999, 2009.

[131] C. Meesookho, U. Mitra and S. Narayanan, "*On Energy-Based Acoustic Source Localization for Sensor Networks*", IEEE Trans. Signal Process., Vol. 56, No. 1, pp. 365-377, 2008.

[132] W. B. Heinzelman, A. Chandrakasan and H. Balakrishnan, "*An application-specific protocol architecture for wireless microsensor networks*", IEEE Trans. Wireless Commun., Vol. 1, No. 4, pp. 660-670, 2002.

[133] A. Wang and A. Chandrakasan, "*Energy-efficient DSPs for wireless sensor networks*", IEEE Trans. Signal Process. Mag., pp. 68-78, 2002.

[134] M.J. Miller and N.H. Vaidya, "*AMAC protocol to reduce sensor network energy consumption using a wakeup radio*", IEEE Trans. Mobile Comput., Vol. 4, No. 3, pp. 228-242, 2005.

[135] W. Xiao, L. Xie, J. Chen and L. Shue, "*Multi-Step Adaptive Sensor Scheduling for Target Tracking in Wireless Sensor Networks*", Processing in IEEE International Conference on Acoustics, Speech and Signal Processing 2006, pp. IV-IV, 2006.

[136] G.J. McLachlan1, "*Mahalanobis distance*", Springer India, in co-publication with Indian Academy of Sciences, General Article, Vol. 4, No. 6, pp. 20-26, 1999.

[137] W. Mendenhall, R.J. Beaver and B.M. Beaver, "*Introduction to Probability and Statistics*", 11th Edition, Brooks/Cole Publishing, ISBN: 0534395198, 2003.

[138] W. Wang, V. Srinivasan, B. Wang and K.C. Chua, "*Coverage for target localization in wireless sensor networks*", IEEE Trans. Wireless Commun., Vol. 7, No. 2, pp. 667-676, 2008.

[139] MM Student Projects, "*Embryology Homepage*", Available at: http://sprojects.mmi.mcgill.ca/embryology/ (URL accessible as of 27/04/2008).

[140] L. Haihao, L. Mei, S. Yi and Q. Deli, "*Research on Task Allocation Technique for Multi-Target Tracking in Wireless Sensor Network*", Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, 360-365, 2007.

[141] Y.E.M. Hamouda and C. Phillips "*Biological Task Mapping and Scheduling in Wireless Sensor Networks*", IEEE International Conference on Communication Technology and Applications 2009, pp. 914-919, 2009.

[142] C. H. Papadimitriou and K. Steiglitz, "*Combinatorial Optimization Algorithms and Complexity*", Englewood Cliffs, NJ: Prentice-Hall, ISBN 0486402584,1988.

[143] K. L. Hoffman, "*Combinatorial optimization: Current successes and directions for the future*", J. Comput. Appl. Math., Vol. 124, No. 1-2, pp. 341–360, 2000.

[144] E. Aarts and J. K. Lenstra, "*Local Search in Combinatorial Optimization*", Princeton University Press, ISBN 0691115222, 2003.

[145] J. Hurink, "*Solving complex optimization problems by local search*", University of Twente, Department of Applied Mathematics, 1999.

[146] L. Wang, H.J. Siegel, V.P. Roychowdhury and A.A. Maciejewski, "*Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach*", Journal of Parallel and Distributed Computing, Vol. 47, No. 1, pp. 8-22, 1997.

[147] J. Banks, J. Carson, B.L. Nelson and D. Nicol, "*Discrete-Event System Simulation*", Prentice Hall, Inc., ISBN 0130887021, 2001.

[148] A.M. Law, "*Simulation Modeling and Analysis*", McGraw-Hill Higher Education , ISBN 0071103368, 2007.

[149] H. Schildt, "*C++: The Complete Reference*", Fourth edition, McGraw-Hill, ISBN 0072226803, 2003.

[150] H.M. Deitel and P.J. Deitel," *C++ How to Program*", Prentice Hall, ISBN 0131426443, 2004.

[151] M. Matsumoto and T. Nishimura, "*Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*", ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, pp. 3-30, 1998.

[152] G. Bianchi, "*Performance Analysis of the IEEE 802.11 Distributed Coordination Function*", IEEE IEEE J. Sel. Areas Commun., Vol. 18, No. 3, 2000.

[153] A. Munawar, M. Wahib, M. Munetomo and K. Akama , "*A Survey: Genetic Algorithms and the Fast Evolving World of Parallel Computing*", The 10th IEEE International Conference on High Performance Computing and Communications, pp. 897-902, 2008.

[154] A. Jayasuriya, S. Perreau, A. Dadej and S. Gordon, "*Hidden vs. exposed terminal problem in ad hoc networks*", Proceedings of the Australian Telecommunication Networks and Applications Conference, Sydney, Australia, 2004.

[155] D. Baron and Y. Birk, "*Coding schemes for multislot messages in multichannel ALOHA with deadlines*", IEEE Transactions on Wireless Communications, Vol. 1, No. 2, pp. 292-301, 2002