Recovery Mechanism on Sensor Networks

by

Bai Li
B.Sc (Deakin University)
B.Sc Honors (Deakin University)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

July, 2011
DEAKIN UNIVERSITY

ACCESS TO THESIS – A

I am the author of the thesis entitled "Recovery Mechanisms on Sensor Networks"

submitted for the degree of Doctor of Philosophy

This thesis may be made available for consultation, loan and limited copying in accordance with the
Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name .............................................................

(Please Print)

Signed .............................................................

Signature Redacted by Library

Date .............................................................

15/05/2012
DEAKIN UNIVERSITY

CANDIDATE DECLARATION

I certify that the thesis entitled "Recovery Mechanism on Sensor Networks"

submitted for the degree of Doctor of Philosophy

is the result of my own work and that where reference is made to the work of others, due
acknowledgment is given.

I also certify that any material in the thesis which has been accepted for a degree or diploma by any
university or institution is identified in the text.

Full Name .................................................................

(Please Print)

Signed .................................................................

Signature Redacted by Library

Date .................................................................

15/05/2012
Dedicated to

my mother Qiuping Bai,

my father Caiwang Li

and

my wife Yanan Cheng

for their unconditional love and support.
Publications

• Conference Papers


• Journal Papers


- **Book Chapters**

Acknowledgements

The past four years have been the most challenging, interesting, and rewarding part of my life. I am very thankful and grateful that I have crossed paths with many wonderful people who have helped me in my pursuit of a PhD at Deakin University, Australia.

First and foremost, I would like to express my gratitude and heart felt thanks to my supervisors, Prof. Lynn Margaret Batten and Dr. Robin Doss. Their kindness, wisdom, passion and determination have helped to shape my attitudes, behaviours, and habits towards both research and life. Their breadth of knowledge, vision for the future, and their enthusiasm for research have been an inspiration to me. Their excellent supervision, teaching, guidance, patience and support during this research have been the key to help me through the unlimited and vast world of thought towards completing my PhD. It also has been a privilege interacting with some wonderful, bright and talented people whose advice, feedback and friendship have made my PhD experience both pleasurable and educational, especially Dr. Gang Li, Dr. Lei Pan, Longxiang Gao, Dr. Morshed Chowdhury, Milan Sangani, Rocky Saini, Prof. Wanlei Zhou, Dr. Wolfgang Schott Dr. Yang Xiang and Dr. Yu Shui. My gratitude also goes to Georgi Cahill, Judy Chow,
Kathy Giulieri, Lauren Fisher, Nghia Dang, and other staff in the school for their kind help and support.

Secondly, without the support of my family, I would not have found the time to study. I thank my parents, my mother Qiuping Bai and my father Caiwang Li, for their unconditional love and ongoing support towards my studies. They have been a great support when times were good and more importantly, when times were not so good. I also thank my wife, Yanan Cheng, for taking care of everything at home so I could concentrate on my study.

Thirdly, I also wish to thank Robert Prandolini of NSST and our project partner Brian Palm for their input and advice throughout the project, and the NSST for their financial support.

Finally, I would also like to take this opportunity to thank the examiners for their time and devotion in reading my PhD thesis.

Above all, I would like to dedicate this thesis to my supervisors Prof. Lynn Margaret Batten, Dr. Robin Doss and my family, who have all been key people in guiding, motivating and supporting me throughout the work done in this thesis.

I love deadlines. I especially like the whooshing sound they make as they fly by.

—Douglas Adams
Abstract

Wireless Sensor Networks (WSNs) are susceptible to a wide range of security attacks in hostile environments due to the limited processing and energy capabilities of sensor nodes. Consequently, the use of WSNs in mission critical applications requires reliable detection and fast recovery from these attacks. While much research has been devoted to detecting security attacks, very little attention has yet been paid to the recovery task. In this thesis, we present a novel mechanism that is based on dynamic network re-clustering and node reprogramming for recovering compromised nodes. In response to node compromise, the proposed recovery approach re-clusters the network excluding compromised nodes; thus allowing normal network operation while initiating node recovery procedures. We propose a novel reclustering algorithm that uses 2-hop neighbourhood information for this purpose. For node reprogramming we propose the modified Deluge protocol. The proposed node recovery mechanism is both decentralized and scalable. We also propose a novel, lightweight authentication protocol that can secure network and node recovery operations such as reclustering and reprogramming. Moreover, we demonstrate through its implementation on a TelosB and MicaZ based sensor network testbed with up to 30 motes that the proposed recovery method
performs well in a low-resource WSN.

We also demonstrate the effectiveness of our recovery protocols on low-resourced WSNs with up to 1000 nodes using the NS-2 simulation platform. We show that our method successfully identifies nodes with high energy levels and is also capable of identifying the number of higher neighbours these nodes have. These nodes are then used as effective local data Aggregator Nodes (AGs) in the recovery process.

In extending this to mobile sensor networks, it is necessary to formalize the mobility support mechanisms. We establish the mobility support needed to ensure the speedy recovery of a mobile WSN and also analyze the impact of mobility on the recovery procedures developed in our earlier work. We discuss mobility of member nodes and of AGs in the cluster model, and also identify triggers for reclustering of the network based on loss of connectedness due to mobility. The challenge is to provide mobility support to a mobile WSN which produces the same recovery performance as in a static WSN.

In addition, our results indicate that the recovery protocols we have introduced for both stationary and mobile WSNs are as efficient for a mobile WSN as for a static WSN in producing a connected set of AGs capable of maintaining communication with the Base Station (BS). Indeed, for networks in which most nodes are mobile at any time, our results clearly indicate that the network is unable to recover from natural breakages of the Connect Dominating Sets (CDSs) due to motion. Furthermore, a mobile network under attack has no chance of survival without mobility support.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publications</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation for the Research</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Research Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3 System Assumptions</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Design Models</td>
<td>6</td>
</tr>
<tr>
<td>1.4.1 Detection</td>
<td>7</td>
</tr>
<tr>
<td>1.4.2 Response</td>
<td>7</td>
</tr>
<tr>
<td>1.4.3 Recovery</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Thesis Structure</td>
<td>8</td>
</tr>
<tr>
<td>1.6 Summary</td>
<td>10</td>
</tr>
<tr>
<td><strong>2 Related Work</strong></td>
<td>11</td>
</tr>
<tr>
<td>2.1 Sensor Networks</td>
<td>12</td>
</tr>
</tbody>
</table>

vii
4 Reprogramming Protocols
  4.1 Design .............................................. 70
    4.1.1 Scenario ......................................... 71
    4.1.2 AG Reprogramming ................................. 73
    4.1.3 Member Node Reprogramming ....................... 74
  4.2 Experiments and Evaluation ............................ 75
  4.3 Standalone Modified Deluge on TelosB and MicaZ ...... 81
  4.4 Summary .............................................. 82

5 Authentication Protocols .................................. 84
  5.1 Design .............................................. 85
    5.1.1 Objectives ........................................ 85
    5.1.2 Assumptions ........................................ 87
    5.1.3 Attack Scenarios .................................. 88
  5.2 Approaches .......................................... 89
    5.2.1 Rabin Algorithms .................................. 91
    5.2.2 SHA-1 Algorithms .................................. 93
  5.3 Technical Problems and Experience of Implementation .... 93
  5.4 Experiments and Evaluation ............................ 96
    5.4.1 Hash Function Performance ........................ 96
    5.4.2 Hash Function Comparison on TelosB and MicaZ .... 98
    5.4.3 Performance Comparison with the Work of Other Researchers 99
  5.5 Summary .............................................. 101

6 Recovery Protocols in Stationary WSNs .................... 103
6.1 Recovery Protocols ...................................... 104
  6.1.1 Reprogramming Protocol from the BS to a Compromised
        Node ............................................. 105
  6.1.2 Reprogramming Protocol from an AG to the BS on Behalf
        of a Compromised Node ........................ 107
  6.1.3 Re-Clustering Protocol from the BS to the Network .... 109
6.2 Experiments and Evaluation ............................ 110
6.3 Implementation Platform Comparison .................. 112
6.4 Protocol Comparison between TelosB and MicaZ ........ 113
6.5 Summary .............................................. 116

7 Mobility Support on Recovery .......................... 118
  7.1 A Motivating Scenario ............................... 121
  7.2 Theoretical Overview ................................ 122
    7.2.1 Assumptions ..................................... 122
    7.2.2 Comparison on Recovery with Other Researchers .... 124
    7.2.3 Mobility Support for the Re-Clustering Algorithm ... 128
    7.2.4 Constructing CDS ................................ 131
    7.2.5 Mobility of AGs .................................. 134
    7.2.6 Mobility of Member Nodes ........................ 137
  7.3 Mobility Impact ...................................... 140
    7.3.1 Clustering and Re-Clustering ...................... 140
    7.3.2 Impact of Mobility on Re-Clustering, Reprogramming and
           Authentication Procedures ........................ 141
    7.3.3 The Impact of Mobility on Recovery Procedures .... 145
### CONTENTS

7.4 Mobile WSN Protocol Description .......................................................... 146  
7.4.1 Mobile Reprogramming Protocol from the BS to a Compromised Node ........ 146  
7.4.2 Mobile Reprogramming Protocol from an AG to the BS on Behalf of a Compromised Node .................................................. 147  
7.4.3 Mobile Re-Clustering Protocol from the BS to the Network .................. 147  
7.5 Experiments and Evaluation ........................................................................ 148  
7.5.1 TelosB Experiments ............................................................................. 149  
7.5.2 Simulation Experiments ....................................................................... 154  
7.6 Resource Needed to Implement Our Protocol ........................................... 163  
7.7 Summary ................................................................................................. 167  

8 Recovery Standards ....................................................................................... 171  
8.1 Background .............................................................................................. 171  
8.2 System Assumptions - A Review ............................................................... 171  
8.3 Summary of the Needed Protocols - Stationary Case ................................ 174  
8.4 The Integrated Recovery Protocol - Modified EMPR and Reprogramming with Authentication ......................................................... 177  
8.5 Implementation Issues ............................................................................. 178  
8.6 Comparison with Other Work .................................................................... 182  
8.7 A Recovery Standard Proposal for WSNs ............................................... 186  
8.7.1 Scope ................................................................................................. 187  
8.7.2 Terms and Definitions ........................................................................ 187  
8.7.3 General Requirements ........................................................................ 187  
8.7.4 Design ............................................................................................... 188
List of Figures

3.1 Clustered Topology of WSN .................................................. 45
3.2 Formation of C(v) .............................................................. 47
3.3 Schemata of the reclustering algorithm ...................................... 49
3.4 Scenario A ........................................................................... 53
3.5 An Example of Storage Format on neighbour Tables .................... 56
3.6 The network comprises a hierarchical structure of the BS, clusters,
    aggregator and member nodes ............................................... 59
3.7 Experimental Set-up on Re-Clustering Protocols .......................... 60
3.8 Simulation energy levels of clusters and aggregators .................... 66
3.9 Simulation Performance of the clustering algorithm ....................... 67
4.1 Scenario B ........................................................................... 73
4.2 AG reprogramming ............................................................. 74
4.3 Member node reprogramming .................................................. 75
6.1 Authenticating a reprogramming request ..................................... 106
6.2 Message diagram of authenticating a reprogramming request ........ 106
LIST OF FIGURES

6.3 Message diagram of authenticating a reprogramming request from an AG ........................................ 108
6.4 Authenticating a reprogramming request ........................................ 108
6.5 Authenticating a reclustering request ........................................ 110

7.1 The BS sends an authenticated reclustering command ............ 129
7.2 The network re-clusters ........................................ 130
7.3 An AG moves out of range of its neighbour AGs .................. 135
7.4 No AG is in range of the BS .................................. 136
7.5 Checking for a broken CDS .................................. 137
7.6 A member node moves out of ranges of its cluster .............. 139
7.7 20-node grid topology for the experiments of Protocols 7.4.1 and 7.4.2 ........................................ 151
7.8 10-node grid topology for the experiments of Protocol 7.4.3 .. 151
7.9 15-node grid topology for the experiments of Protocol 7.4.3 .. 152
7.10 Number of AGs chosen: The static case and the mobile cases with
20% of the nodes mobile with a 2 sec pause at waypoints ........ 156
7.11 Number of AGs chosen: The static case and the mobile cases with
20% of the nodes mobile with a 5 sec pause at waypoints ........ 157
7.12 Number of AGs chosen: The static case and the mobile cases with
40% of the nodes mobile with a 2 sec pause at waypoints ........ 158
7.13 Number of AGs chosen: The static case and the mobile cases with
40% of the nodes mobile with a 5 sec pause at waypoints ........ 158
7.14 Number of AGs chosen: The static case and the mobile cases with
80% of the nodes mobile with a 2 sec pause at waypoints ........ 159
7.15 Number of AGs chosen: The static case and the mobile cases with 80% of the nodes mobile with a 5 sec pause at waypoints . . . . . . 160
7.16 Comparative metrics of CDS and network on 1000 nodes . . . . . . 161
7.17 Percentage of nodes included in a cluster in mobile WSN without mobility support and with a 2 sec pause . . . . . . . . . . . . . . . . . . . . . . 162
7.18 Percentage of nodes included in a cluster in a mobile WSN without mobility support and with a 5 sec pause . . . . . . . . . . . . . . . . . . . . . . 163
List of Tables

3.1 Testing results 1 ........................................ 62
3.2 Testing results 2 ........................................ 63
3.3 Performance of the reclustering algorithm ............... 63
3.4 Energy levels of clusters and AGs ..................... 64

4.1 Reprogramming the nodes with BlinkM.nc 2 metres from BS . . 76
4.2 Reprogramming the nodes with BlinkM.nc 11 metres from BS . . 77
4.3 Reprogramming the nodes with BlinkM.nc 21 metres from BS . . 77
4.4 Reprogramming the nodes with GoldenImage.nc 2 metres from BS 78
4.5 Reprogramming the nodes with GoldenImage.nc 11 metres from BS 78
4.6 Reprogramming the nodes with GoldenImage.nc 21 metres from BS 79
4.7 Reprogramming and reclustering the network using modified deluge 81
4.8 Performance of standalone reprogramming on TelosB and MicaZ . 81

5.1 Comparison of the Rabin and SHA-1 hash functions ............ 97
5.2 Performance of standalone hash functions on different platforms . 99
5.3 Comparison of two implementations of protocol III ............ 100
6.1 Performance of a reprogramming protocol from the BS to a compromised node ........................................ 111
6.2 Performance of a reprogramming protocol from an AG to the BS on behalf of a compromised node .................. 111
6.3 Performance of a reclustering protocol from the BS to the network 112
6.4 Hardware performance comparison of TelosB and MicaZ motes . . 113
6.5 Performance of a reprogramming protocol from the BS to a compromised node on different platforms ................. 114
6.6 Performance of a reprogramming protocol from an AG to the BS on behalf of a compromised node on different platforms ....... 115
6.7 Performance of a reclustering protocol from the BS to the network on different platforms .................................. 116

7.1 Comparison of solutions to restoring connectivity in mobile WSNs 126
7.2 Comparative performance of Rabin authentication and reclustering from BS to network .................................. 152
7.3 Times of first CDS break in mobile WSN without mobility support 164
7.4 Attributes of the AG nodes in respective papers .................. 168

8.1 Specification comparison of MicaZ and TelosB .................. 173
8.2 Capabilities of the two platforms .................................. 182
8.3 Implementation issues identified in recent work .................. 186
Chapter 1

Introduction

Wireless Sensor Networks (WSNs) represent a new generation of real-time embedded systems with significantly different communication constraints from traditional networked systems. These constraints are important because WSNs can be applied in a wide range of applications and on various types of equipment. Sensors are employed to monitor a diverse range of conditions at different locations, such as temperature, sound, vibration, pressure, motion and pollutants.

With their development, various attacks have occurred. The aims of these attacks are usually to take over or destroy nodes in the network, or to disrupt data flow. Detection and recovery have become major challenges in protecting sensor networks from these attacks.

Section 1.1 started with the research motivation. Section 1.2 gives the research objectives in regards to recovery mechanisms on sensor networks. Section 1.3 discusses the system assumptions. Section 1.4 presents the design model which
includes detection, response and recovery stages. Section 1.5 outlines the thesis structure. Section 1.6 summarises the chapter.

1.1 Motivation for the Research

WSNs are susceptible to a wide range of security attacks in hostile environments due to the limited processing and energy capabilities of sensor nodes. Consequently, the use of WSNs in mission critical applications requires reliable detection and fast recovery from these attacks. While much research has been devoted to detecting security attacks, very little attention has yet been given to the recovery task. In this thesis, we present a novel mechanism that is based on dynamic network reclustering and node reprogramming for recovery from node compromise. We compare our approach with the current detection and recovery protocols so that we can use the best from existing work alongside our own ideas.

1.2 Research Objectives

The goal is to develop and test several protocols for efficient detection of and run a recovery from attacks in simply structured stationary and mobile WSNs. These protocols are proposed in the following chapters.

We will set up a WSN with TelosB and MicaZ motes and simulation on NS-2 [NS-] and test to identify which detection and recovery protocols have the highest potential for use. A NS-2 simulation platform will also be set up to perform the
testing on the large scale networks.

In this research, additional cryptographic techniques will be added to the functionality of the WSN and embedded in the recovery protocols. We will compare our approach to that of other researchers in both static and mobile cases.

For the purposes of an attack situation in which nodes can be lost or compromised, efficient detection and recovery can only take place if the network can function as normally as possible. Therefore, we propose to retain connectivity and maximize flexibility in the network. This is achieved by allowing each node to play the role of either a member node or a Aggregator Node (AG) as appropriate under the conditions that arise for the entire network to efficiently re-organize itself in order to remain connected. The objectives of recovery are to:

- R1. Restore the network functions to a target level established by the Base Station (BS).
- R2. Maintain the network for as long as possible.

1.3 System Assumptions

WSNs can be built in a number of ways depending on the desired application [LR02]. Often, several types of nodes are present, classified in terms of the role they play, such as gathering data, analyzing data, and deploying applications etc. Networks may be built with various power resources to assist in performing these functions. Mobile components can be added to increase the functionality
and flexibility of the network (for instance mobile agents are used in LiB2). Recent research [KS04] has found that significant energy savings can be achieved by dynamic management of node duty cycles in sensor networks with high node density. In this approach, some nodes are scheduled to sleep (or enter a power saving mode) while the remaining active nodes provide continuous service. A fundamental problem is to minimize the number of nodes that remain active, while still achieving an acceptable quality of service for applications. In particular, maintaining sufficient sensing coverage and network connectivity with active nodes are critical requirements in sensor networks.

WSN operations include data discovery, which is achieved by way of sensing application specified target events. Additionally, the sensor network needs to process this information in a distributed manner and then forward it to any interested data sink or remote BS. These sensor network tasks can be managed individually by a sensor node or by several nodes simultaneously.

An intuitive analysis of the sensor network activities of a simple network leads to mapping tasks to roles as follows:

- **Member node** - senses data and transmits it.
- **AG** - senses, collects, analyses and transmits data.
- **BS** - controls the system, senses, collects, analyses, transmits and stores data.

Since all sensor nodes in the network are essentially deployed to collaboratively sense target events, all nodes must assume the functions of a member node or AG.
In addition, AGs take on the responsibility of coordinating the sensing activities in their neighbouring region (also known as a sensing zone) and aggregate and forward the information to the BS. The task of coordination is not a simple one and it is also not a short term job. In order to provide instantaneous sensing and reporting capability (dependent upon sensing applications) each AG may need to systematically rotate its responsibilities transparently among neighbouring nodes without much overhead communication.

1. We assume the existence of a globally unique ID for each sensor node. We assume that all node locations are fixed for the duration of their lifetime.

2. We do not assume that sensor nodes are tamper resistant; on the contrary, we assume that if an adversary compromises a node, they can extract all key material, data, and code stored on that node. While tamper resistance may be a viable defense for physical node compromise for some networks, we do not see it as a general purpose solution. Additionally, effective tamper resistance tends to add significant cost per unit, and sensor nodes are intended to be inexpensive.

3. We assume that the BS is trustworthy in the sense that it has never been compromised and always behaves correctly. Most, but not all routing protocols depend on nodes to trust messages from the BSs.
1.4 Design Models

We base our work on a simply structured WSN with two types of nodes: member
nodes which do basic collection, transmission and receiving of data, and AGs
which have additional responsibilities of assimilating and analysing data. We
assume that very little power resources are available.

We propose a decentralized approach to node configuration with two conditions:
first, that each node not be ’too far away’ from an AG, secondly, that the network
is divided into a set of clusters of about equal size. WSNs configured in this way
are expected to be able to implement detection and recovery more quickly than
WSNs which are not configured in the same manner.

We also use a method of 'clustering' in which, after an attack, the network dy-
amically re-organizes itself to achieve both of the above conditions, thus permit
the implementation of recovery protocols to continue.

This involves the following work to be undertaken throughout my PhD thesis.

- Development of protocols and techniques for recovering from attacks in
  wireless networks including reclustering, reprogramming, authentication
  and mobility support.

- Testing of protocols for recovery from attacks in wireless networks.

- Production of a wireless network structure prototype which recovers quickly
  from an attack in stationary and mobile sensor networks.

We identify three stages needed for our recovery model. The first is detection of
an attack or compromised node, the second is a response to this and the third is recovery from it. Each stage employs different strategies and methods.

1.4.1 Detection

There are a number of procedures available in the literature for the detection of attacks. Authentication procedures and voting algorithms [LB07a] are often used. We have developed algorithms for detection which we will deploy in this work [LB07a], [LB07b]. We will distinguish between insider and outsider attacks later in this thesis.

1.4.2 Response

In the response stage, we identify what kind of attack has been launched and which nodes, if any, have been compromised. In case a node has been compromised, we use the behaviour of the node to assist in determining this fact. This may include some of the following: the node cannot manipulate data including writing, reading, sending and receiving; the node corrupts, modifies or deletes data; the node transmits large amounts of data, jamming the traffic in the network; the node sends fake data to sniff for information (such as keys, algorithms or passwords).
1.4.3 Recovery

Once a response has been executed, a distributed recovery methodology is implemented. Our ideas here are novel, using self-organization of the WSN with a specific target structure to optimize the recovery process. We propose a decentralized approach to node configuration with two conditions: first, that each node be no more than two hops away from an AG, secondly, that the network is divided into a set of clusters of about equal size in such a way that any two nodes in the cluster are at most four hops away. These features, along with self-organization, will be deployed in order to ensure that communication between the network nodes and the BS is not disrupted.

WSNs configured in this way are expected to be able to implement detection and recovery more quickly than WSNs which are not. This is because we will employ a localized approach, cluster by cluster, in detecting, responding to and recovering from an attack. A detailed description of self-organization and cluster building are provided in the next section.

1.5 Thesis Structure

The thesis comprises eight chapters and is organized as follows:

Chapter 2 provides an introduction to the research topic. It lays the foundation of the thesis model by giving an overview of the theoretical basis for the research.

From Chapter 3 to Chapter 7, the development of each key technique used in our recovery mechanism is discussed and comprehensive testing is conducted.
accordingly.

Chapter 3 focuses on the reclustering protocols used to partition the networks automatically. The network determines the time to trigger the reclustering process when a portion of networks are under attack and some of the nodes lose connections with their neighbours.

Chapter 4 presents a new reprogramming technique to recover the nodes from attacks based on the Deluge [YYS03]. The BS or AGs can utilize this technique to restore the compromised nodes without reclustering the entire network.

Chapter 5 specifically examines a few of the possible hash functions and proposes the RABIN which is a lightweight hash function used for authentication purposes in sensor networks.

Chapter 6 presents the theoretical development of the recovery mechanism and integrates three techniques discussed in Chapters 3, 4 and 5 together.

Chapter 7 introduces the mobility support and extends the research work we have done in previous chapters to mobile sensor networks.

Chapter 8 gives an overview of the comparison with other researchers’ work on recovery in sensor networks.

Finally, Chapter 9 summarizes the contributions of this thesis and some possible avenues for future work.

Appendix A provides the original Multi-point Relays (MPR) and enhanced versions related to the reclustering which is part of our recovery mechanism design.
1.6 Summary

In this chapter, the foundations for this thesis have been laid by introducing the research problems and issues. We then discuss the motivation and goals of the research. This introductory chapter has later discussed the system assumptions and the design model which will be used to frame our research work.

In the next chapter we will briefly introduce the sensor networks and discuss related research that has been conducted on recovery in sensor networks.

A serious and good philosophical work could be written consisting entirely of jokes.

—Henry Dribble
Chapter 2

Related Work

In this chapter, the literature review was conducted focusing on reclustering, reprogramming, authentication and mobility support. The research regarding recovery on mobile WSNs is also presented.

Section 2.1 addresses the important elements for sensor networks. Section 2.2 and 2.3 further investigates the threat models and attack scenarios which can be adopted for our recovery mechanisms. Section 2.3 surveys the current development of reprogramming techniques. Section 2.4 surveys the current development and techniques used for reclustering, reprogramming, authentication and mobility support in sensor networks. Section 2.4 concludes this chapter.
2.1 Sensor Networks

Advances in wireless communication, low-power electronics, and low-power radio frequency design have enabled the development of low-power sensor nodes with integrated sensing, processing and wireless communication capabilities. These sensors are tiny devices designed to sense and collect data of particular interest. Equipped with routing capabilities, a network with these wireless sensors can offer an efficient and easily deployable solution for information gathering in complex and large-scale environments.

WSNs are envisioned to consist of hundreds, even thousands of low-power, low-cost nodes, possibly mobile, but more likely at fixed locations, deployed en masse to monitor and affect the environment. They do not rely on a fixed infrastructure (i.e., wired connections or wireless BSs) and are self-configuring, multi-hop networks. These properties make them particularly attractive for military applications as they offer reconnaissance and surveillance capabilities not possible with traditional communication networks [CK03]. For example, in battlefield situations they can be deployed to detect enemy movements or serve as early warning systems by sensing the presence of chemicals and gases used in chemical and biological warfare. They can also find application in disaster response, habitat modelling, environmental monitoring and industrial sensing. Typically, communication is achieved by multi-hop wireless communication between the sensor nodes and a central point commonly referred to as a sink. However, other communication patterns such as one to one communication between sensor nodes can also be observed in many emerging applications.
CHAPTER 2. RELATED WORK

One of the main concerns with WSN operation is information and communication security. The broadcast nature of the wireless medium makes these networks inherently insecure. Further, sensors are limited in their computational and communication capabilities, and being battery powered, they are also energy constrained. As a result established security mechanisms become either computationally infeasible or energy inefficient [GR02]. In addition, most protocols proposed for WSNs were not designed with security as a requirement. Consequently, sensor networks suffer from a wide range of security attacks that can degrade network performance and compromise the confidentiality and integrity of the data collected.

The security techniques developed in this project will be targeted at present day sensor node platforms, such as the Berkeley MicaZ mote [HC02]. The MicaZ mote is a small (several cubic inch) sensor/actuator unit with a CPU, power source, radio, and several optional sensing elements. The processor is a 4 MHz 8-bit Atmel ATMEGA103 CPU with 128 KB of instruction memory, 4 KB of RAM for data, and 512 KB of flash memory. The CPU consumes 5.5 mA (at 3 volts) when active, and two orders of magnitude less power when sleeping. The radio is a 916 MHz low-power radio from Radio Frequency Module (RFM), delivering up to 40 Kbps bandwidth on a single shared channel and with a range up to a few dozen metres or so. The RFM radio consumes 4.8 mA (at 3 volts) in receive mode, up to 12 mA in transmit mode, and 5 uA in sleep mode. An optional sensor board allows mounting of a temperature sensor, magnetometer, accelerometer, microphone, sounder, and other sensing elements. The whole device is powered by two AA batteries, which provide approximately 2850 mA hours at 3 volts. Since
we do not have access to the source code in the MicaZ mote, our experiments will be carried on the open-sourced Telos B research platform, which is held to be an equivalent platform by the research community.

A critical factor in sensor network operation is efficient power management. At full power, the Berkeley MicaZ mote can run for two weeks before exhausting its batteries. Consequently, if long-term network operation is required it is critical they run at very low duty cycles. Similarly, since the power consumption of the radio is three orders of magnitude higher when transmitting or listening than when in sleep mode, efficient radio management is another critical factor [KW03].

Sensor networks are characterized by data redundancies due to the large number of nodes deployed and the spatial distribution of the individual sensors and sensed events. This redundancy can be exploited to reduce the number of transmissions from individual sensors resulting in energy savings. Further, sensor readings from multiple nodes can be aggregated at one of many possible AGs. An AG collects sensor readings from surrounding nodes and forwards a single message representing an aggregate of the individual values. Typically, AGs are regular sensor nodes, and their selection is not necessarily static. AGs could be chosen dynamically for each query or event. It is also possible that every node in the network functions as an aggregation point, delaying transmission of an outgoing message until a sufficient number of incoming messages have been received and aggregated.

Sensor networks differ from other distributed systems in important ways. The resource-starved nature of sensor networks pose greater challenges for security. These devices have very little computational power; public-key cryptography is
too expensive to be unusable, and even fast symmetric-key ciphers must be used sparingly. With only 4 KB of Random Access Memory (RAM), memory is a resource that must be husbanded carefully and hence practical security protocols should not maintain much state. Further, communication bandwidth is extremely dear, with each bit transmitted consuming the equivalent power to execute 800/1000 instructions, and as a consequence, any message expansion caused by a security mechanism comes at a significant cost. Power is the scarcest resource of all, with each milliamp consumed being one milliamp closer to node exhaustion, and it is important that every aspect of sensor network operation is designed with the aim of optimizing the lifetime of nodes and the network.

2.2 Threat Models

Many attacks on WSNs have emerged in recent years (see [LNLP06] for a survey). Most of the work in this area has concentrated on detection and identification of the attacks rather than on recovery. The detection and recovery methods developed in this project should be capable of dealing with the most important of these, which we list here.

All attacks below are physical attacks on the nodes and data transferred in the network. However, we will classify them into those where nodes inside the WSN are compromised and taken over and those in which an attacker acts as a malicious node working from the exterior of the network. In some sense, all attacks can be classified as a 'denial of service'; that is, the objective is to stop the WSN from functioning as intended.
We shall consider an attack to be ‘internal’ if a compromised node is initiating it and ‘external’ if a node outside the network is initiating the attack. This will be important in terms of the response mechanisms needed for recovery.

- **Flooding Attack**
  
  This attack floods the network with useless traffic [KW03]. This has two effects on sensor networks. First, the attack traffic consumes network resources, and prevents legitimate traffic from reaching the BS. More importantly, it causes sleep deprivation of sensor nodes and wastes their energy. Such an attack is often combined with other attacks such as altering of routing information in order to maximize the effect of flooding. (An outsider attack).

- **Spoofing and Altering of Routing Information**
  
  Spoofing refers to an attacker impersonating a network node by falsifying the identity field in routing messages [KW03]. This can enable an attacker to create routing loops in the network, or increase the length of routes. This in turn causes increased traffic congestion and deprives the network of resources (An insider attack).

- **Selective Forwarding**
  
  Selective forwarding occurs when a compromised node drops a packet bound for a particular destination [DQW03]. In this way, an attacker can selectively filter traffic from a particular part of the network. Other possible variations of selective forwarding can involve dropping all packets or ran-
domly dropping packets. Random dropping can be hard to reliably detect and trace (An insider attack).

- Sinkhole Attack

The main purpose of the sinkhole attack is to lure all traffic from nodes in a region to a compromised node [KW03]. This is achieved by forging or altering route packet information to make a compromised node look very attractive to the routing algorithm, causing neighbouring nodes to assume that the compromised node is the best path to their destinations. Sinkhole attacks can also act as a platform for launching other attacks. An example would be to combine it with a selective forwarding attack. Since all traffic basically flows through the compromised node, a selective forwarding attack would thus become more effective and easier to achieve (An insider attack).

- Sybil Attack

In Sybil attacks [NSSP04], a malicious node pretends to be a number of different nodes in the network. The malicious node can acquire identities either by fabricating new ones or by learning the identity of other nodes. To attack a network, the malicious node can use the impersonated identity to communicate with legitimate nodes directly, or by indirect communication where the malicious node advertises that it has a path to the impersonated node (Both insider and outsider attack).

- Wormholes

In a wormhole attack, a malicious node tunnels messages between two different parts of the network via a high speed link [HPJ02b]. This can make
distant nodes appear ‘closer’ in the network, which can be useful as part of a Sybil attack. Moreover, if the attacker is appropriately positioned, it can disrupt the entire network by diverting traffic from the BS (An insider attack).

The major risks in sensor networks are:

- loss of nodes because of physical damage.
- nodes compromised and the internal code altered in order to implement functions of an attacker.
- attacks on the network from outside the network while the nodes function normally.

We will not address the physical loss problem in this thesis as this relies on an outside network to physically replace the node. Our focus will be on node compromise and on attacks from outside the network (such as flooding) while nodes still operate normally.

The WSN must be capable of determining that an attack has taken place and assess what damage has been done before it can begin recovery.

2.3 Attack Scenarios

The standard attack scenarios in a WSN setting are known as the passive and active attack [Opp96] models for communication compromises.
2.3.1 Passive attacks

In a passive attack, the intruder is able to capture and interpret data, thus extracting information. The protocols described in the next section are not designed to protect against a passive attack.

2.3.2 Active attacks

In an active attack, both the integrity and the availability of the communication are threatened.

An intruder implementing an active attack may destroy the integrity of communication in two ways: by changing the content of the communication and by changing the authorship of the communication. (Message Content Changes and Message Authorship changes.)

An intruder implementing an active attack may also

- capture a communication and delete it altogether (Message Deletion),
- resend it at a later time (Message Replay), or
- resend several communications a number of times during a flooding attack (Flooding).

An additional active attack, which we will consider here, is a situation where a compromised node that has received a reprogramming message sends an acknowledgement without actually reprogramming. In other words, it lies about being reprogrammed (A compromised node lies).
2.4 Related Work

In this section, the literature review intends to give an overview of reclustering protocols, reprogramming, authentication algorithm and mobile support respectively in sensor networks. This helps us to develop an understanding of current available technologies on the market which we can use to implement the recovery mechanism and even compare the work of other researchers.

2.4.1 Re-Clustering

ZRP protocol HAPS, and the terminodes approach: for mobility management, ZRP uses zones similar to clusters whereas the terminodes based approach uses the concept of self organized virtual regions. Routing in both these approaches involves two different schemes; a proactive routing scheme for nodes within a local virtual-region or zone, and a reactive scheme for nodes located in remote virtual-regions or zones. In mobile ad hoc networks, the availability of the network is dependent on each user’s discretion, with incentive for cooperation by way of virtual money called nuglets employed in the terminodes. However, in real life, it is difficult to design and implement, as routing for nodes involves two different schemes simultaneously.

Krishnan and Starobinski [KS03] present two algorithms that produce clusters of bounded size and low diameter by having nodes allocate local growth budgets to neighbours. Unlike the expanding ring approach, their algorithms do not involve the initiator in each round and do not violate the specified upper bound on the cluster size at any time, thus having a low message overhead. However,
CHAPTER 2. RELATED WORK

the cluster size can be very large which cannot provide efficient recovery at the recovery stage.

Meguerdichian et al. [MKQP01] and [MKPS01] have formulated the exposure and coverage properties of sensor networks using computational geometry-based techniques such as the Voronoi diagram and the Delaunay Triangulation. The sensor models used in their analysis include two concepts. One is that the sensing ability diminishes with increasing distance. Second, noise bursts diminish the sensing ability but the effect of noise can be minimized by allowing sensors to sense over longer periods of time (exposure). However, a distributed and localized algorithm measuring coverage and exposure of a sensor node or of a region of WSN deployment is not discussed in the work of Meguerdician et al. This method asks sensors to monitor over longer time periods which can cause the nodes to die earlier.

Slijepcevic and Potkonjak [SP01] propose a heuristic that organizes the sensor network by selecting mutually exclusive sets of sensor nodes that completely cover the monitored area together. However the sensing zones are based on cumulative (rather than individual) sensing coverage. In this way, for any target event (genuine or spurious), the sensing zones either report individually or collaboratively with some degree of fault tolerance. Since the sensors are deployed randomly rather than deterministically, there may be regions of the monitored area that are covered by a higher number of sensors. This redundancy in sensing coverage could be utilized to save energy if the energy required for continuous sensing is comparable to that consumed for message transmission. A thorough analysis of increased energy savings achieved by utilizing redundant sensing coverage, and
analysis of the overall lifetime of the network and its fault tolerant sensing ability is needed. A higher number of sensors deployed in the same area can waste our resource in this algorithm.

Heinzelman et al. [HCB00] proposed an alternative clustering-based approach, called LEACH (Low-Energy Adaptive Clustering Hierarchy). This approach relies on the following two main assumptions: (i) there exists a unique BS with which all the sensors want to communicate; (ii) all the sensors have the ability to communicate directly with the BS. In order to save energy, the LEACH protocol selects a fraction $p$ of the sensors to serve as cluster-heads, where $p$ is a design parameter that must be engineered off-line. Cluster-heads communicate directly with the BS whereas other nodes forward their data through the cluster-heads (typically, the one closest to them). In order to share the energy load, the LEACH protocol implements a load-balancing procedure that allows different nodes to become cluster-heads at different times. The assumption that all the sensors have the ability to communicate with the BS directly is impractical for the large size sensor network.

Other work related to LEACH includes the PEGASIS [LR02] protocol, in which nodes form a chain to achieve further energy savings. The main idea in PEGASIS is for each node to receive from and transmit to close neighbours and take turns being the leader for transmission to the BS. This approach will distribute the energy load evenly among the sensor nodes in the network. The nodes are initially placed randomly in the play field, and therefore, the $i$'th node is at a random location. The nodes are organized to form a chain, which can be accomplished by the sensor nodes themselves using a greedy algorithm starting from some node.
Alternatively, the BS can compute this chain and broadcast it to all the sensor nodes. PEGASIS improves on LEACH by saving energy in several stages. First, in the local gathering, the distances that most of the nodes transmit are much less compared to transmitting to a cluster-head in LEACH. Second, the amount of data for the leader to receive is at most two messages instead of 20 (20 nodes per cluster in LEACH for a 100-node network). Finally, only one node transmits to the BS in each round of communication. This approach cannot be used on recovery as the roles of each sensor have to be specified when we start the recovery and is also very expensive.

Subramanian and Katz [SK00] propose general architectural guidelines for designing self-organizing WSNs; a self configuration architecture that leads to a hierarchical network with address auto-configuration and a number of other useful properties. Their self organizing algorithm lists four phases of operation. These are the discovery phase, organizational phase, maintenance phase, and self reorganization phase. Our method can be designed based on these four phases.

Tian and Georganas [TG02] propose to increase the system lifetime and at the same time preserve original sensing coverage by using a node scheduling scheme that turns off redundant sensor nodes in a network of wireless sensors. This scheme allows nodes in the network to autonomously turn themselves on/off using local neighbour information. This local neighbour information is used to determine if a node needs to be ON so as to cover a region of some neighbour that is not being covered by any other neighbours. Sensing coverage determination employs geometrical techniques that calculate shared neighbouring sectors modelled from a circular sensing region with central angles being interpreted from
the AoA (Angle of Arrival) of incoming signals. AoA measurements need a multi-
directional antenna, which is very sophisticated hardware in sensor technology.
Nodes that find themselves redundant with respect to sensing coverage advertise
status advertisement messages (SAM) to neighbours. This SAM advertisement
employs a random back-off timer to avoid having all neighbours turning them-
selves off, in turn leaving a blind spot. This randomization may sometimes lead
to a situation where neighbouring nodes will come to know of a blind spot only
after some time has elapsed. Too much calculation is involved in this algorithm
which is also expensive to implement. However, turning the nodes on/off automati-
cally based on their local neighbour information is a useful action to take in
order to save the network life.

The Rapid algorithm presented in [KS04] has a low message complexity. In the
Rapid algorithm, the initiator is assigned a budget B, which accounts for itself
and evenly distributes B-1 among its neighbours by sending a message to each one
of them. An arbitrary subset is chosen if there are more neighbours than budget.
The neighbours that receive the message account for themselves and distribute the
remaining messages among all their neighbours except the parent. The messages
propagate until they reach a stage where the budget is exhausted. Each node
that receives a message sends an acknowledgment to its parent when either the
budget is exhausted or it has received acknowledgments from all its children. The
algorithm terminates when the initiator receives acknowledgments from all the
neighbours it sent a message to. The acknowledgments can be used to convey
the size and depth of the sub-tree and the maximum hop count reached. The
initiator can use this to determine the size and depth of the cluster. Furthermore,
by including hop count information in the messages and acknowledgments, a modified algorithm can limit the depth of the cluster, but this can restrict the maximum cluster size achieved.

The Persistent algorithm [KS04] is a recursive elaboration of the Rapid algorithm that significantly improves the worst-case behaviour in terms of the cluster size produced. Using the Persistent algorithm, the initiator is provided a budget B, which accounts for itself and evenly distributes B-1 among its neighbours via messages. An arbitrary subset is chosen if there are more neighbours than budget. The neighbours that receive the message, account for themselves and distribute the remaining among all their neighbours except the parent. The messages propagate until they reach a stage where the budget is exhausted. In this algorithm, each node that receives a message does not send an acknowledgment to its parent immediately on receiving acknowledgments from all its children. It computes the size of the sub-tree and compares it to the budget allocated to it. It distributes the shortfall among its neighbours, which were either not explored previously or met all previously allocated budgets. When either the budget is met or when further growth is not possible, it returns an acknowledgment to its parent. The algorithm terminates when the initiator determines that the bound has been met or when no further growth is possible (for example, there are not enough nodes adjacent to the clusters that are free to join this cluster). In the presence of a single initiator, the Persistent algorithm produces a cluster of the specified bound. When a cluster of the specified size cannot be constructed, because not enough unclustered nodes are available, it attempts to build the largest cluster possible.

The clustering method applied in our WSN is based on the Enhanced Multi-point
Relay (EMPR) algorithm that has been proposed by J. Wu [2]. This algorithm efficiently partitions a WSN with a flat topology into a hierarchically network consisting of a set of small-sized clusters. It can be shown that the selected AGs are connected among each other and the AGs thus form a Connect Dominating Set (CDS) of network nodes. The proposed method determines in a distributed, localized fashion, a set of AGs by firstly collecting 2-hop neighbourhood information in each node $v$ of the WSN, then selects the AGs by iteratively searching for the best-suited set of MPR, and finally associates member nodes to an appropriate AG. Further details on the clustering algorithm are given in the following subsections.

### 2.4.2 Reprogramming

The scenarios above indicate the need for the flexibility of options in terms of reprogramming. In particular, localization of reprogramming within a cluster ensures that no disturbances occur in the remainder of the network. Reprogramming of an AG by the BS allows the BS to temporarily take on the role of the AG, thus ensuring continuity of operation through the network. Reprogramming methods for sensor networks can be broadly categorized into two types:

1. **In-system reprogramming (ISP):** ISP is the most common method of reprogramming sensor devices as it is supported by most microcontrollers. The program code is developed in a powerful host machine and is loaded

\[ \text{A CDS is defined as a subset of nodes of a network, where every node is either in the subset or a neighbour of a node in the subset, and the graph introduced by the subset is fully connected.} \]
onto the sensor device through the parallel or serial port through a direct connection. Such a method does not allow the dynamic reprogramming of sensor devices in-network which is a requirement if autonomous recovery of sensor devices is to be achieved. Further, in terms of reprogramming time, because only one node can be manually reprogrammed at a time, the time taken for network-wide reprogramming of a large number of compromised devices will be prohibitive.

2. In-network reprogramming (INP): INP enables the programming of sensor devices over the wireless channel (i.e., over the air) and hence, are more suited to large scale deployments and for recovery techniques. The process of over the air reprogramming can be generalised into the following steps: (1) encoding, (2) dissemination and (3) decoding [JC04]. In the first step, the control node (BS or AG) prepares the code packets to be distributed. The code is disseminated by the control node in the second step which is followed by the sensor device receiving the code packets, then decoding and storing the code packets. At this point within the node, the network programming module rebuilds the program code and calls the boot loader to load the code into program memory.

Current methods for over the air reprogramming include XNP from Crossbow [WZC06], while Deluge [WZC06], MOAP [TSE03] and incremental programming [JC04]. Each of these methods is designed for either multi-hop or single hop reprogramming of all nodes in a deployment. XNP broadcasts the program code over a single hop, Deluge makes use of epidemic dissemination over multi-hops to reprogram all nodes in the network, and MOAP supports multi-hop program-
Incremental programming first targets components of the code in the node to be changed, and then reprograms using only former or updated versions of these components. In order to do this, it applies the Rsync algorithm [Tri99] to sensor nodes. However, these methods do not enable the reprogramming of a specific node either over a single hop or multi-hop, and hence, are not suited for node recovery purposes.

As a basis for our design of a reprogramming protocol, we chose Deluge based on the above considerations. Deluge aims to increase the transmission throughput by using optimization techniques such as adjusting the packet transmission rate and spatial multiplexing. As a starting point, we modified the Deluge protocol to selectively reprogram nodes that have been identified as compromised. The performance of the redesigned Deluge protocol was evaluated through field experiments using our test bed.

### 2.4.3 Authentication

Several methods for detecting DoS attacks in general networks have been proposed. One solution [WS02] is to use a MAC admission control rate limit, so the network can ignore excessive requests without sending expensive radio transmissions. However, this limit cannot drop below the expected maximum data rate the network supports. If nodes fail in one area, this could lead to high traffic flow in another area because the sender can select different paths to transmit the packets across, thus avoiding dead nodes. We do not use this method due to the problem of high traffic flow.
Perrig et al. [PST+02] proposed the TESLA protocol to securely broadcast messages in a WSN. This protocol uses a one-way hash chain (OHC) number broadcast message. A different OHC number is allocated for each time slot, and this number is used to generate MACs for the packets sent in that time slot. To tolerate packet losses, TESLA has been extended by introducing multi-level OHCs [LN03]. A higher-level OHC is used to bootstrap low-level OHCs. But this method requires time synchronization which is not practical and uses a lot of resources, which we cannot supply in a WSN.

Hu et al. [HPJ02a] proposed a secure on-demand routing protocol for ad hoc networks in which an OHC is used to thwart malicious routing request floods. When an initiator node broadcasts a ROUTE REQUEST message, it attaches an OHC number on the message. Other nodes can check the authenticity of the packet by verifying the OHC number. OHC have been used in INSSENS to limit broadcast floods for control routing updates in WSNs [DHM05]. In contrast, our approach will employ OHCs to defend against PDoS attacks on unicast messages that follow a path. Problems unique to unicast messages must be addressed, e.g. maintaining OHCs when many packets are lost, and generating and storing OHCs in a highly resource-constrained node.

More recently, en-route filtering schemes have been proposed for intermediate nodes to filter false data generated by malicious AGs and also detect intruders engaged in PDoS attacks [YLLZ04]. The basic idea is that the intermediate nodes share some keys with the member nodes in a node group or cluster. Member nodes generate MACs for the reported data using the shared keys, and intermediate nodes can verify the MACs before forwarding packets [YYS03]. In the SEF
scheme proposed by Ye et al. [YLLZ04], the Bloom filter [YLLZ04] is used to reduce the size of MACs and ensure their security. The intermediate nodes and member nodes use randomly pre-distributed keys to generate and verify MACs. In this scheme, it is highly likely that the false data can be dropped by one of the intermediate nodes and not reach the BS. However, there are several problems with the SEF scheme. First, the SEF scheme uses a probabilistic approach. This means it cannot guarantee that every spurious packet will be filtered out on the path. In addition, statistically, a spurious packet will be forwarded to a certain number of nodes before it is filtered out. Second, the message overhead of the SEF scheme is still large. The size of the Bloom filter is 14 bytes long, which is about half of the data payload of a TinyOS packet. This method is also too expensive to be deployed in WSNs.

In the interleaved key scheme proposed by Zhu et al. [ZSJN04], member nodes and intermediate nodes set up interleaved keys using randomly pre-distributed keys. These interleaved keys and hop-by-hop authentication ensure that the BS will detect any false packets when no more than a certain number of nodes are compromised. The problem of the interleaved key scheme is that there is no efficient mechanism to authenticate two nodes to each other through multiple hops. In addition, the communication overhead of the pair-wise key establishment for multi-hop nodes is large, and the process is slow.

Pires et al. [PFWL04] proposed a mechanism based on signal strength and geographical information for detecting malicious nodes staging HELLO flood and wormhole attacks. The idea is to compare the signal strength of a reception with its expected value and calculate using geographical position. A protocol for
disseminating information about detection of malicious nodes is also proposed. The detection rate of their solution depends on a number of parameters and this mechanism is not available to detect path-based DoS attacks.

Parno et al. [PPG05] developed two protocols which seek to minimize power consumption by limiting communication, while still operating within the extremely limited memory capacity of typical sensor nodes for node replication attacks. Randomized Multicast distributes location claims to a randomly selected set of witness nodes. Line-Selected Multicast exploits the routing topology of the network to select witnesses for a node’s location and utilizes geometric probability to detect replicated nodes. As this method cannot cooperate with our clustering algorithm we do not utilize it.

CODA [WEC03] uses a combination of past and present channel loading conditions, along with the current buffer occupancy, to infer accurate detection of congestion at each receiver with low cost. Sensor networks must know the state of the channel, since the transmission medium is shared and may be congested with traffic between other devices in the neighbourhood. Listening to the channel to measure local loading incurs high energy costs if performed all the time. Therefore, CODA uses a sampling scheme that activates local channel monitoring at the appropriate time to minimize cost while forming an accurate estimate. Once congestion is detected, nodes signal their upstream neighbours via a backpressure mechanism. If congestion has occurred, it will be very hard for nodes to signal their upstream neighbours. At the same time, the whole system relies on this mechanism which is risky and can be compromised by attackers.

Hierarchical Sensor Network Debugging (H-SEND) [HLBL06] observes the health
of a sensor network and repairs any error remotely by reprogramming through the wireless network. At initiation, the programmers specify important properties as 'invariants' that should never be violated in the network’s operation. The checking can be done locally or remotely, depending on the nature of the invariant. If an error is detected at run-time, the logs of the observed variables are examined to analyze and correct the error. After errors are corrected, new programs or patches can be uploaded to the nodes through the wireless network. But this approach is very expensive to implement as it costs too much in terms of power in a real scenario.

Deng et al. [DHM05] developed a method of using OHCs to protect end-to-end communications in WSNs against PDoS attacks. It prevents PDoS attacks from compromised intermediate nodes or from outside sources capable of launching PDoS attacks, because an adversary cannot generate the next valid OHC number. Any old, replayed OHC numbers will simply be dropped. Furthermore, the memory and computational costs of OHC execution are quite lightweight. In addition, this scheme tolerates packet losses. But there are also some obvious disadvantages. One of these is that it cannot constrain PDoS attacks to compromised nodes in WSNs, which can specifically penalize those nodes which store a hash function.

All of the work using MAC results in high local computing costs and the subsequent death of nodes. The inter-leaved key scheme has an extremely high communication overhead. We therefore reject these but use the idea of Deng et al., which is based on OHCs as it works to effectively prevent PDoS attacks against WSNs. We extend their work in [LB07a] and [LB07b] to include detection
of such attacks by efficiently employing an MA to detect if the sender of the hash has been compromised.

General crypto methods, namely encryption and authentication using public-key cryptosystems [MWS04] [WKC+04], are not reasonable because sensor nodes have a very low calculation capability and small memory, and therefore are not able to operate such crypto algorithms within sufficient time. Some researchers have proposed solving such security problems in sensor networks [KSW04], however TinySec [KSW04] does not specify any key pre-distribution method and it merely assigns a global key to the system. Thus, an attacker seeking the TinySec key need only target a well-known address or area of memory in advance, rather than downloading the complete binary image of the operating system and applications, thereby reducing the download further and enabling compromise in mere seconds [HBH05].

2.4.4 Mobility Support

The mechanisms developed for recovering WSNs have focused on stationary sensors. However, sensor nodes can be mobile when they are attached to personnel, vehicles or equipment. Consequently, a critical requirement for a recovery protocol is to enable fast recovery taking into account the mobility of sensor devices. In the literature, several mobility models have been proposed for mobile ad hoc networks (MANETs). Although none specifically exist for sensor networks, WSNs can be generalised to MANETs to gain an understanding of the impact of mobility on network communication and the flow-on effect on recovery procedures.
A mobility model is defined as a set of rules used to generate trajectories for mobile entities [SBM05]. In a mobile sensor network, one of the key characteristics that needs to be modelled accurately, and which has a significant impact on the effectiveness and performance of all mechanisms and protocols, is the mobility of the network. Mobility models can be categorised into trace models or synthetic models [SBM05]. Trace models are built on real world data (traces) gathered over a period of time to model the mobility in the area while synthetic models are based entirely on mathematics. Both model types suffer drawbacks. Trace models, while having the advantage of being based on empirical data, cannot be generalised to a large area outside the trace area without distorting its validity. Synthetic models, though lacking in realism, can be uniformly applied to large deployments without loss of generality. Due to the large scale and deployment specific nature of sensor networks, trace models are not applicable and realistic synthetic models are required.

Synthetic mobility models proposed for mobile ad hoc networks can be categorised as individual mobility models or group mobility models [SBM05]. The individual mobility models such as the random walk [GLAM99], the random waypoint [JM96], random direction mobility [RMSM01], probabilistic random walk [LH99] and Gauss-Markov [CG98] model, use mobility in an unrestrained fashion and are highly individualistic. It can be argued that mobility of sensor devices is restrained by a variety of factors such as terrain, nature of application, and user/vehicle mobility capabilities (since sensors are perceived to be attached to personnel/vehicles/equipment). Further, mobility of an individual sensor may not always be independent of the motion of other sensors as in many applications of
WSN’s, there will be some group mobility aspects observed (eg. [MY05]). Group mobility models proposed for ad hoc networks such as reference point group mobility [HGPC99], exponential correlated random mobility [HGPC99] and Pursue mobility [SBM05], work on the underlying assumption that all nodes belonging to a group travel towards a common destination. Clearly sensor node mobility will not satisfy this assumption as there can be multiple subgroups identified within a WSN that have a common objective.

The above discussion implies that a realistic WSN mobility model should integrate both individual and group mobility characteristics. Such a synthetic model for WSNs does not exist at this time. As the development of a new mobility model is outside the scope of this research, we propose the use of the random waypoint model which is the most widely used model in ad hoc network research [SBM05]. In the random waypoint model, a mobile node randomly chooses a destination within the network area and moves towards it with a fixed speed chosen from a uniform distribution. On arrival at the selected destination, it waits for a random amount of time before selecting a new destination within the network to move towards.

[DW04] addresses the issue of mobility by increasing the transmission range of BS and extending two-hop neighbourhood information to n-hop. This does not resolve the issue completely as the solution is originally used to tackle the issues in stationary sensor networks.

[AAA09] develops DARA, a Distributed Actor Recovery Algorithm, which opts to efficiently restore the connectivity of an inter-actor network to its pre-node-failure level. Depending on the type of connectivity considered, two algorithms, namely,
DARA-1C and DARA-2C, are developed to address 1 and 2-connectivity requirements, respectively. DARA is a localized scheme that avoids the involvement of every single actor in the network. DARA pursues a coordinated multi-actor relocation in order to re-establish communication links among impacted actors. The main idea of DARA-1C is to replace the dead actor with a suitable neighbour. The selection of the best candidate (BC) neighbour is based on the node degree and the physical proximity to the dead actor. The relocation procedure is recursively applied to handle actors that get disconnected due to the movement of one of their neighbours (e.g., the BC that replaced the faulty actor). Similarly, DARA-2C identifies the nodes that are affected, i.e., lost their 2-connectivity property due to the failed actor. Some of these nodes are then relocated in order to restore 2-connectivity. Although both DARA-1C and DARA-2C pursue node relocation to restore the desired level of connectivity, they fundamentally differ in the scope of the failure analysis and the recovery.

[KAU10] presents a distributed PArtition Detection and Recovery Algorithm (PADRA) to determine possible partitioning in advance and self-restoration of connectivity in case of such failures with minimized node movement and message overhead. The idea in the recovery is to find the closest dominating node and use it as a replacement for the failed node so that connectivity is preserved. The basic idea of their recovery mechanism is as follows: if there is a dominating node among the neighbours of the cut-vertex, it will be designated as the node to replace the cut-vertex upon failure since it will be the closest dominatee to the failed node. Otherwise, the cut-vertex node designates its closest neighbour to handle its failure. In case of such a failure, the closest neighbour will apply
the same idea in order to find the closest dominatee to itself. That is, it picks its closest neighbour and this continues until a dominatee is hit. The idea is to share the load of movement overhead among the nodes on the path to the failed cut-vertex node in order to extend the lifetime of each node and thus the whole network.

[MYA10] demonstrates RIM, a distributed algorithm for Recovery through Inward Motion. RIM opts to efficiently restore the connectivity of a WSN through repositioning some nodes. RIM is a localized scheme that limits the scope of the recovery process. Basically, RIM orchestrates a coordinated multi-node relocation in order to re-establish communication links in the neighbourhood of the failed node. The main idea is to move the neighbours of a failed node ‘f’ inward, toward the position of f so they can reach each other. The rationale is that these neighbours are the ones that are directly impacted and when they reach each other again, the network connectivity can be restored to its pre-failure status. The relocation procedure is recursively applied to handle any node that gets disconnected due to the movement of one of their neighbours (e.g., those which move toward the faulty node). The main advantages of RIM are its simplicity and effectiveness. RIM avoids sophisticated diagnostics for assessing the effect of a node failure on the network connectivity, e.g., by checking whether the failed node is a cut vertex or not. Instead, RIM employs a simple procedure that recovers from both serious and non-serious breaks in connectivity. The entire recovery process is distributed, enabling the network to self-heal without any external supervision.

However [AAA09], [KAU10] and [MYA10] are too high-resource intensive and
not suitable in low-resource sensor networks. Therefore, the recovery price can be extremely expensive.

### 2.4.5 Route discovery

Determining a route through a WSN from a node to the BS so that information collected can be delivered in a timely fashion is an important issue for stationary WSNs, but a critical one for mobile WSNs.

Dynamic Source Routing (DSR) is a widely used routing protocol in [Joh94], where the source node specifies the complete ordered route in the packet header before sending the packet data. It consists of two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism whereby a source wishing to send a packet to a destination node obtains a source route to the destination node; Route Maintenance is the mechanism whereby the source node is able to re-establish a route if the network topology has changed. DSR is commonly chosen because of its simplicity and performance [HLC+07].

Wu [Wu03] and [WLD04] solves the problem of connectivity and route discovery simultaneously using CDS based on the EMPR algorithm. This algorithm efficiently partitions a WSN with a flat topology into a hierarchical network consisting of a small-sized cluster of nodes around an AG. It can be shown that the selected AGs are connected amongst each other (in a graph-theoretical sense, this means that there is a path between any two AGs comprising 1-hop neighbours) and also that every node is either an AG or in range of an AG (a set of AGs is called a CDS). This method determines in a distributed, localized fashion a set
of AGs by firstly collecting 2-hop neighbourhood information in each node of the WSN, then selecting by iteratively searching for the best-suited set of MPR, then finally associating member nodes to an appropriate AG. The CDS is the basis of our reclustering method, and a fundamental part of the recovery process, as described in Section 3.

2.5 Summary

A critical issue for security in WSNs is how to detect and recover from attacks on the network in an accurate and computationally efficient manner. In this chapter, we have discussed the current detection and recovery methods used in sensor networks or distributed networks and indicated which ideas were suitable for use within our strategy.

In the next chapter, we shall introduce reclustering algorithms as a recovery mechanism.

A good guide will take you through the more important streets more often than he takes you down side streets; a bad guide will do the opposite. In philosophy I’m a rather bad guide.

—Alois Pichler and Herbert Hrachovec
Chapter 3

Re-Clustering Protocols

In this chapter, we introduce a reclustering algorithm designed as a recovery mechanism in sensor networks.

Network self organization and clustering are fundamental to our approach and are the basis for the configuration of the WSN. If a node is to play the role of member node in one network configuration and the role of an aggregator in another, then it must be equipped with the appropriate code.

WSNs can be built in a number of ways depending on the desired application [LR02]. Often, several types of nodes are present, classified in terms of the role they play, such as gathering data, analyzing data, deploying applications etc. Networks may be built with various power resources to assist in performing these functions. Mobile components can be added to increase the functionality and flexibility of the network (for instance mobile agents are used by [LB07b]).

Recent research [KS04] has found that significant energy savings can be achieved
by dynamic management of node duty cycles in sensor networks with high node density. In this approach, some nodes are scheduled to sleep (or enter a power saving mode) while the remaining active nodes provide continuous service. A fundamental problem is to minimize the number of nodes that remain active, while still achieving an acceptable quality of service for applications. In particular, maintaining sufficient sensing coverage and network connectivity with the active nodes are critical requirements in sensor networks.

WSN operations include data discovery, which is achieved by way of sensing application specified target events. Additionally, the sensor network needs to process this information in a distributed manner and then forward it to any interested data sink or remote BS. These sensor network tasks can be managed individually by a sensor node or by several nodes simultaneously.

Section 3.1 discusses the assumption of the reclustering protocols. Section 3.2 explores how reclustering is developed in sensor networks from a design perspective. Section 3.3 addresses the issues reclustering protocols face. Section 3.4 presents the experiments and evaluation of reclustering protocols on both hardware and simulation platforms. Section 3.5 concludes this chapter.

3.1 Assumptions

For the purposes of an attack situation in which nodes can be lost or compromised, efficient detection and recovery can only take place if the network can function as normally as possible. Therefore, we propose to therefore retain connectivity and
maximize flexibility in the network. This is achieved by allowing each node to play the role of either a member node or AG as appropriate under the conditions arising, and also for the entire network to efficiently re-organize itself in order to remain connected when a compromised node is detected.

We therefore assume that all nodes are programmed initially with exactly the same code but only the code needed for the fulfillment of the role it is currently playing.

To initiate the process, all nodes have the code of member node turned on. The clustering algorithm is then run three times to identify member nodes and AGs. The clustering algorithm we chose also provides a connected network and an additional property that for any two clusters, aggregators in each cluster are only one hop away from the other.

1. We assume the existence of a globally unique ID for each sensor node. We assume that all node locations are fixed for the duration of their lifetime.

2. We do not assume that sensor nodes are tamper resistant. On the contrary, we assume that if an adversary compromises a node, they can extract all key material, data, and code stored on that node. While tamper resistance might be a viable defense for physical node compromise for some networks, we do not see it as a general purpose solution. Additionally, effective tamper resistance tends to add significant per-unit cost, and sensor nodes are intended to be inexpensive.

3. We assume that the BS is trustworthy in the sense that it is never compromised and always behaves correctly. Most, but not all routing protocols depend on nodes to trust messages from BSs.
3.2 Design

3.2.1 Objectives

Considering the power resources available in the network, any recovery strategy should be designed based on the following:

Low memory usage. Memory and storage requirements should be minimized.

Energy efficiency. The energy used in code dissemination should be low so as to minimally affect the network lifetime.

Speed. New program code should be propagated and installed quickly (i.e., within seconds or minutes).

The cluster configuration of the network and the ability to self-organize repeatedly is critical to our recovery strategy. The detection algorithm will be run locally, cluster by cluster and, if the clusters are still connected, a co-ordinated response will be implemented. If the clusters have become disconnected, they will work independently.

3.2.2 System Design

WSNs can be built in a number of ways depending on the desired application [LB07a]. Often, several types of nodes are present, classified in terms of the role they play, such as gathering data, analyzing data, or deploying applications.

An intuitive analysis of the sensor network activities of a simple network leads
to mapping tasks to roles as follows:

- a member node senses and transmits data.
- an AG controls member nodes and senses, collects, aggregates, analyses and transmits data.
- a BS controls the system and collects, analyses, transmits and stores data.

Our algorithm constructs relatively small connected clusters. This algorithm will help to detect attacks in a reasonable time. A hop represents one portion of the path between source and destination. When communicating over the Internet, for example, data passes through a number of intermediate devices (like routers) rather than flowing directly over a single wire. Each such device causes data to 'hop' between one point-to-point network connection and another.

Pre-requisite: a set of connected nodes \( V \) so that each pair of nodes is joined by a multi-hop path. Node ID is unique. Each node is programmed with an identical code allowing it to play the role of a member node and an AG. At set-up, each node is assigned aggregator mode.

Figure 3.1 illustrates the topology of the envisioned WSN. It comprises a large number of low-resource sensor nodes that are connected to a BS in order to analyze the sensed data. By partitioning the WSN topology into a set of clusters, a hierarchical network topology is obtained that is power-efficient, scalable, and resilient to security attacks.

Each cluster comprises one aggregator and in general, several member nodes. Each member node is always connected to a single associated aggregator to ex-
change data and control packets. The communication between member nodes and the aggregator is controlled by establishing synchronization between the involved nodes and the use of a cyclic superframe structure of length $t_P$ as proposed, for example in [LB07a]. At the beginning of a superframe, the aggregator sends a beacon message to wake-up sleeping member nodes and coordinate the intra-cluster communication within the active period of the superframe in a TDMA based manner. During the inactive period of the superframe, member nodes are sent to sleep to preserve battery energy. A aggregator gathers the data sensed by the member nodes of its cluster and transmits the aggregated data to the BS. For this purpose, aggregators are connected among each other to form an overlay network that manages itself in a distributed fashion. The aggregators communicate with each other by using a CSMA/CA Medium Access Control (MAC) scheme and employing multi-hop packet transmissions. Therefore, aggregators also act as relays to forward packets on behalf of other aggregators to the BS.
3.2.2.1 Neighbourhood Information Collection

After establishing a WSN with a flat network topology, each node $v$ of the WSN collects 2-hop neighbourhood information by exchanging HELLO messages with its 1-hop neighbour nodes. For this purpose, each node $v$ broadcasts a HELLO message at a random time instant within the superframe time interval $t_P$. Each message carries the ID of the transmitting node $v$, the IDs of all currently known 1-hop neighbours of the transmitting node, and the metric values $M$ that characterize the capabilities of the node $v$ and its neighbours to act as an AG.

The metric $M(v)$ of node $v$ is defined in Equation 3.1 as

$$M(v) = a \frac{e(v)}{e_{max}} + (1 - a) \frac{d(v)}{d_{max}}, a \in [0, 1]$$

(3.1)

where $e$ is the available battery energy of the node and $e_{max}$ its maximum value, $d$ is the node degree that should not exceed a pre-defined value $d_{max}$, and $a$ is a pre-defined weighting parameter. To successfully distribute all neighbourhood information between the involved network nodes, broadcasting of HELLO messages has to be repeated with the updated neighbour node list and metric values in at least two succeeding time intervals $t_P$. After the exchange of the HELLO messages, each node $v$ knows the IDs of all 1-hop and 2-hop neighbours, the connectivity between these nodes and their corresponding metric values $M$. We denote the set of all 1-hop and 2-hop neighbours as $N_1(v)$ and $N_2(v)$, respectively.
3.2.2.2 MPR Selection

After collecting the neighbourhood information, each node $v$ selects a set of MPR that can be viewed as candidate aggregators. This set comprises a small subset of nodes $C(v)$ from the 1-hop neighbour set $N_1(v)$ of node $v$ that fully covers the 2-hop neighbour set $N_2(v)$ of node $v$. Thus $C(v)$ is also called the coverage set of node $v$, and it can be shown that the $C(v) \cup v$ forms a CDS for $N_2(v)$. The coverage set $C(v)$ is obtained by executing the modified EMPR algorithm given below, which takes into account the known metric values $M$ of the involved network nodes. The formation of $C(v)$ is also shown in Figure 3.2. Those nodes with larger metrics are the favored candidates for the aggregators. This preferential choice is employed in the modified EMPR algorithm.

After the MPR have been selected, each node broadcasts its coverage set $C(v)$ to
Algorithm 1 The Modified EMPR

Add all free neighbours of \( N_1(v) \) to the coverage set \( C(v) \). Node \( u \) is a free neighbour of \( v \) if \( v \) is not the highest metric neighbour of \( u \).
Add node \( u \in N_1(v) \) to the coverage set \( C(v) \), if there is an uncovered node in \( N_2(v) \) that is only covered by \( u \). Any node in \( N_2(v) \) that is not covered by \( C(v) \) is called an uncovered node.
Add node \( u \in N_1(v) \) to the coverage set \( C(v) \), if \( u \) covers the largest number of uncovered nodes in \( N_2(v) \). Use metric \( M \) of the nodes to break a tie when two nodes cover the same number of uncovered nodes.
Repeat step 3 until all nodes in \( N_2(v) \) are covered.

its 1-hop neighbours at a random time instant in the next time interval \( t_P \).

3.2.2.3 Cluster Forming

A node \( v \) decides to act as an AG if it has never been compromised, and

1. it has a larger metric \( M(v) \) than all of its 1-hop neighbours and has at least two unconnected neighbours, or if

2. it is in the coverage set formed by its neighbour with the largest metric \( M \)

When the nodes have decided on their role in the network, the aggregators broadcast their newly accepted role to its 1-hop neighbours at random time instants in the next time interval \( t_P \). After a member node has received this status message from all of its candidate aggregators in its neighbourhood, the member node selects the best-suited aggregator by sending an associate-request message. The aggregator acknowledges successful association with an associate-confirm message. The cluster forming process is completed after all nodes in the network
CHAPTER 3. RE-CLUSTERING PROTOCOLS

Figure 3.3: Schemata of the reclustering algorithm
have taken on their appropriate roles in the network. For understanding purpose, the schemata of the reclustering algorithms are presented in Figure 3.3.

The proposed hierarchical network topology offers several significant advantages compared to a flat topology in terms of network energy consumption and recovery from network security attacks. Using a clustered topology reduces the energy consumption in the network because, firstly, member nodes can transmit with a lower power than AGs and, secondly, member nodes can sleep while the AGs manage and control the network. Moreover, AGs can aggregate data before forwarding the aggregated data which in turn reduces the overall size of the relayed packets to the BS. Using a clustered topology also helps to recover faster and more reliably from security attacks on the WSN. Recovery from security attacks on one or several member nodes can be locally performed by the associated AG without affecting the operation of other clusters, while the recovery from security attacks on AGs or several member nodes belonging to separate clusters requires cooperation of all AGs in the WSN.

3.2.3 Network Design

In order to standardize the clustering algorithm, Institute of Electrical and Electronics Engineers (IEEE) 802.15.4/Zigbee protocol stack is followed for development.

While the IEEE 802.15.4/Zigbee protocol stack is considered as a promising technology for low-cost low-power WSNs, several issues in their specifications are still questionable. One of those ambiguous issues is how to build a synchronized
cluster-tree network. In fact, the current IEEE 802.15.4/Zigbee specifications restrict the synchronization in the beacon-enabled mode (by the generation of periodic beacon frames) to star-based networks, while they support multi-hop networking using the peer-to-peer mesh topology, but with no synchronization. Even though both specifications mention the possible use of cluster-tree topologies, the description of how to effectively construct such a network topology is missing.

The ZigBee Network Layer is developed on top of our implementation IEEE 802.15.4 for nesC/TinyOS. The Hurray project [Pro] has implemented and enabled the cluster-tree network topology with their proposed mechanism for beacon scheduling in order to enable an efficient use of synchronized cluster-tree networks. This implementation induces minor changes to IEEE 802.15.4 implementation, and the open-ZB [SC06] in order to implement the Time Division Beacon Scheduling approach. However, in our algorithm, we are going to use the CDMA/CA approach for time synchronization purposes because there are no clusters formed after node deployment. Once the nodes complete running our proposed clustering algorithm, each coordinator in its cluster can start sending a beacon message to manage the cluster traffic and avoid the collision.

3.2.4 Scenario

We assume a network architecture, as in Figure 3.4, with options for the BS and AGs as indicated. In this case, all AGs are in one hop range of the BS.

25 nodes are set in fixed positions alongside an earthquake table 15mx15m es-
tablished over a known geological fault likely to be the location of an earthquake within the next three years. Every few months the network is checked and malfunctioning nodes are replaced. A BS is located in a reasonably secure location 20 metres from the table and within direct range of a majority of the sensors.

The nodes are placed in a fixed position alongside the table and can detect movements in the table in three directions: two perpendicular horizontal directions, labelled X and Y, and one vertical direction, labelled Z [APSS06]. After set-up, there are nine AGs, all in direct reach of the BS. The BS is supplied with benchmark data indicating those figures deemed to be 'normal' and those indicating an earthquake tremor.

2008

July 5 11:32 - An earthquake tremor hits the table damaging seven nodes including four AGs. Data received by the BS from the nodes is reduced to half the usual amount. The BS issues an alert.

July 5 11:35 - The BS initiates tests to determine which nodes are possible reprogramming targets.

July 5 11:45 - The BS broadcasts a reprogramming message to the network requesting those nodes with identified IDs to reprogram.

July 5 11:50 - Three of the damaged AGs and two of the member nodes are now functioning again. The BS is receiving a normal load of data.

July 5 11:55 - The BS initiates the clustering algorithm in order to ensure that maximum data is being received.
CHAPTER 3. RE-CLUSTERING PROTOCOLS

The network continues to operate successfully for several months.

3.3 Technical Problems and Experience of Implementation

In this section, I will list and describe several technical problems I encountered during development.

We chose to use TelosB for all reclustering work because the size of reclustering code cannot be supported by MicaZ, hence in this section. Later in the thesis, we use MicaZ to do the comparison, specifically reprogramming (Chapter 4) and authentication (Chapter 5) as MicaZ can store for the code size.

As TinyOS has few debugging options, we decided to use the printf to test our algorithms by printing messages over the serial port. The printf formats the given arguments according to the format string, and writes them to the standard
output.

The TinyOS printf library provides this terminal printing functionality to TinyOS applications through motes connected to a pc via their serial interface. Messages are printed by calling printf commands using a familiar syntax borrowed from the C programming language. In order to use this functionality, developers simply need to include a single component in the top level configuration file (PrintfC), and include a ‘printf.h’ header file in any components that actually call printf().

Currently, the printf library is only supported on msp430 and atmega128x based platforms (e.g. mica2, micaZ, telos, eyesIFX). Even for now, this only available debugging tool on TinyOS without proper GUI support can be stil very difficult and challenging for TinyOS learners.

When I implemented one-hop neighbour information exchange and two-hop neighbour information exchange by storing the mote addresses into neighbour tables for the clustering algorithm, we designed the neighbour table data structure for this purpose. It can be a difficult job to manage the relationship between one-hop neighbour tables and two-hop neighbour tables. The design objective was to make sure the neighbour tables are clear to each node and also the designers so applications can be built on top of them. We have modified the data structure by using one one-dimensional array and one two-dimensional array instead of one two-dimensional array. The idea of using both arrays is explained as follows.

The one-dimensional array (Array[]) is used to store the one-hop neighbour information. For example, if Node A (Node ID 2) has one-hop neighbours, Node B (Node ID 3) and Node C (Node ID 4). Node A receives the Hello message from
Node C first, then it will store the Node C address in the Array[1]=4 and Node B address in Array[2]=3 later. Array[0]=2 is reserved for TOS_LOCAL_ADDRESS for future use.

The two-dimensional array (Array[][]) is used to store the two-hop neighbour information. From the above example, we extended the scenario. If Node C has one-hop neighbours, Node D (Node ID 5) and Node E (Node ID 6). Basically, Node C should have its one-hop neighbour table like Array[1]=5, Array[2]=6 and Array[3]=2 if the order to receive the Hello message is Node D, Node E and Node A. Node C injects its neighbour table information in the packet to send to Node A. Node A compares its current one-hop neighbour and two-hop neighbour table it receives from Node C to eliminate the redundant address information (i.e. one address could appear in one-hop neighbour table as well as two-hop one). How to store the two-hop neighbour information in Node A’s two-hop neighbour table will be very easy. We put the node ID as the first dimension and the order of neighbour nodes in Node C as the second dimension in the array. Figure 3.5 describes how this process works.

Currently, the TinyOS Core Working Group supports TinyOS on two platforms: Cygwin which runs on Windows and Linux. Installing a TinyOS environment has five basic steps. Windows requires an extra step, installing Cygwin, which is a UNIX-like environment as this provides a shell and many UNIX tools which the TinyOS environment uses, such as perl and shell scripts. Our telosB platform uses a USB port instead of a COM port as a result because it can be slightly difficult to manage the COM ports on occasion. The COM management can cause an error you can’t install or load the program on the nodes The recommended
solution can be the following:

1. Restart the computer.

2. Use the software called FTClean to reset the PORT configuration in your PC in order to let Windows assign the PORT again.

3. Switch the USB to a different one.

As Cygwin is not a real UNIX environment, these problems can not be avoided as far as we are aware of. Some basic knowledge of UNIX is also required for its development.

At the beginning of forming the clusters, we have to avoid traffic collision during the message broadcasting if all the nodes start at the same time. However, we can’t use the Time Division Beacon Scheduling approach because there are no coordinators yet. Therefore, we decided to choose a random timer to broadcast
the message. This method is a relatively random approach because the random number is still seeded based on the node ID and there is no complicated powerful random number generator that has been developed in TinyOS due to the hardware restrictions. Normally we resend the message at least three times in order to make sure nodes can receive all the incoming messages. However, we still can’t avoid packet loss since broadcasting is the only way to spread the message.

Another problem we discovered at the development stage is data consistency. Our clustering algorithm has been divided into four stages: one-hop neighbour information exchange, two-hop neighbour information exchange, metric information exchange and coverage set information exchange. Nodes have to strictly follow the stages and can’t skip any of the stages in order to make the collected information consistent. For example, Node A, Node B and Node C are one hop from each other. Node A and Node B has completed the one-hop neighbour information exchange but Node C is still at the stage of one-hop neighbour information exchange and only receives the information from Node B. If Node A doesn’t have the information of Node C, Node C could become the two-hop neighbour of Node A because it is the one-hop neighbour of Node B. We can extend the waiting time of Node A and it could involve a more global information exchange such as retransmission time of a node, since Node A, Node B and Node C do not have knowledge of each other. This can make network traffic much worse, and because of this, we do not take this approach. What we do is let nodes resend the packet at least five times to wait for a late incoming message. However the nodes still don’t know if another node has moved to the next stage or not and this method can’t resolve the problem effectively. This is the trade-off to using our algorithm.
3.4 Experiments and Evaluation

We work on the open-sourced TelosB research platform. We assume a small (several cubic inch) sensor/actuator unit with a CPU, power source, radio, and several sensing elements. The processor is a 8 MHz TI MSP430 microcontroller with 16 KB of instruction memory, 10 KB of RAM for data, and 48 KB of flash memory. The CPU consumes 1.8 mA (at 3 volts) when active, and 5.1uA power when sleeping. The radio is a 2400 MHz to 2483 MHz globally compatible ISM band, delivering up to 250 Kbps high data radio bandwidth on a single shared channel and with a range of up to a few dozen meters or so. The RFM radio consumes 4.8 mA (at 3 volts) in receive mode, up to 12 mA in transmit mode, and 5A in sleep mode. The whole device is powered by two AA batteries.

Our implementation involves a sensor network of 25 nodes that are deployed within one building. The range of these devices varies from 25m to 40m in indoor and outdoor environments respectively. The network has a hierarchical structure with one BS node and multiple sensors connected through multihop paths via AGs to the BS as shown in Figure 3.6.

The simulation is developed in NS-2 with up to 1000 nodes running. It is easier to manage a large scale of sensor networks on NS-2 simulation platforms.

3.4.1 Hardware Implementation

In running the clustering algorithm, we assume that the network has been set up and is running in a stable situation. If the BS identifies a compromised
AG, it will trigger a command to re-cluster. At this point, all nodes have the ability to become aggregators. Each node sends out an invitation to join it in a cluster. When a node decides to be an aggregator because nodes have joined it, it reports this to the BS. Thus, at the start of clustering, there are several clusters and aggregators with each node belonging to a cluster. At the end of a cluster algorithm run, there may be a different number of clusters and AGs.

The clustering algorithm is installed on each TelosB mote. We test the time it takes a network to re-cluster from the time the command is issued to re-cluster to the time that each node has joined a cluster. In our test scenarios as shown in Figure 3.7, we start by manually powering each node, with start time being the time at which each node has power. We use LED lights to indicate when each node has joined a cluster; a red LED indicates that the node has become an AG; a blue LED indicates that a node is a member node. We take the stop time to be the point at which each node is lit.

We test four scenarios. In the first two, nodes are all one hop from each other
and each node is one metre from a fixed point. In the next two scenarios, nodes are either one or two hops from other nodes. In scenario 3, the maximum node distance is 60 metres and in scenario 4, the maximum node distance is 42 metres. The following shows how we deployed the nodes in the experiments.

I implemented a WSN testbed with Crossbow’s TelosB motes [Cro]. In our testbed, up to 20 sensor nodes were located on a regular grid of 10 (5x2), 15 (5x3), and 20 (5x4). The closest distance between nodes was set to 1 m and the radio transmission range of each node was set to approximately 1.5 m.

The proposed recovery mechanism was implemented on a TinyOS v2.1 based TelosB programming platform. The metric M was always computed with a parameter value set to 0.5, while the values for e(v) and d(v) were pre-specified at each run. Therefore, we incorporated both the energy and degree metrics to en-
sure that the AG selection maintains energy-efficiency and network connectivity. In our experiments, we ran 15 tests for each network size. We considered two cases: (1) energy levels in all nodes are the same (i.e., the metric $M$ is fixed), and (2) energy levels across nodes are variable (i.e., the metric $M$ is node-dependent). Further, since the network topology is fixed, the node degree does not vary with time.

The focus of the experiments was to test the performance of the reclustering and reprogramming methods with respect to fast recovery from security attacks, while at the same time maintaining reasonable battery energy levels. In our experiments, we assumed that the network was capable of detecting compromised nodes, and any compromised node which cannot be recovered will be deleted from the re-clustered network. We assessed the performance of our method by measuring the following performance metrics:

- Average clustering time: Average time for reclustering to be completed. The clustering time for each run is calculated from the time the BS issues a 'recluster' command to the time that all nodes are included in a cluster.
- Average battery energy level of all WSN AGs.
- Average battery energy level of all WSN member nodes.

In addition, we also recorded the number of AGs elected by the algorithm. The expected number of AGs for a given network varies with the number of nodes in the network as shown through simulations in [Wu03].
3.4.2 Data Collection

Because our clustering algorithm needs to use information about hops between the nodes in our stationary network, we stored a table containing this information in the algorithm. For each node, a table containing the IDs of each node in the network (at most two hops away) is dynamically constructed and stored in the node at network set-up. Each row in the table corresponds to a node and its ID. Thus, if a network has thirty nodes, each table will have size thirty. When reclustering, a node searches through its table to identify nodes with which it can cluster. The larger the table, the longer this is expected to take.

Therefore, we ran tests with 3-entry neighbour tables, corresponding to scenarios 1 and 3, where these are assumed to be the entire network. With 30-entry neighbour tables, the nodes in the four scenarios above are assumed to be more than two hops away from any other nodes in the network. This is in order to compare reclustering where table size is expected to impact on the speed.

Table 3.1 and Table 3.2 show the results:

We saw, as expected, that using a 3-entry neighbour table took less time to run
CHAPTER 3. RE-CLUSTERING PROTOCOLS

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>1st Round (mins)</th>
<th>2nd Round (mins)</th>
<th>3rd Round (mins)</th>
<th>4th Round (mins)</th>
<th>5th Round (mins)</th>
<th>Average (mins)</th>
<th>Std. Dev (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11‘07</td>
<td>11’23</td>
<td>11’28</td>
<td>11’30</td>
<td>11’02</td>
<td>11’18</td>
<td>11.37</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>11’53</td>
<td>11’57</td>
<td>12’20</td>
<td>12’05</td>
<td>12’09</td>
<td>12’05</td>
<td>9.48</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>13’21</td>
<td>13’39</td>
<td>14’41</td>
<td>13’58</td>
<td>14’20</td>
<td>14’00</td>
<td>28.45</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>16’29</td>
<td>16’47</td>
<td>16’20</td>
<td>16’06</td>
<td>16’39</td>
<td>16’30</td>
<td>17.47</td>
</tr>
</tbody>
</table>

Table 3.2: Testing results 2

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Clustering Time (secs)</th>
<th>Std. Dev. (secs)</th>
<th>Average No. of AGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 nodes</td>
<td>162</td>
<td>1.49</td>
<td>2</td>
</tr>
<tr>
<td>15 nodes</td>
<td>167</td>
<td>1.42</td>
<td>3</td>
</tr>
<tr>
<td>20 nodes</td>
<td>175</td>
<td>1.47</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3.3: Performance of the reclustering algorithm

the clustering algorithm than a 30-entry neighbour table. The data indicates that the completion time to run the algorithm increases when the network structure is more complicated and the network size is bigger.

Table 3.3 shows the performance results of the reclustering algorithm. In terms of the number of AGs elected, these results compare well with the simulations of Wu [Wu03], in which the transmission range is relatively high compared to the density of the nodes in the WSN. Wu’s work, however, assumes no collisions, while in our implementation one node per run was lost on average, either because of collisions or because of data errors when 1- and 2-hop neighbour tables were being compiled.

Table 3.4 shows the average metric value per network versus that of the selected
AGs. In calculating $M(v)$, floating point errors were avoided by using Equation 3.2

$M(v) = e(v)d_{max} + d(v)e_{max}$ \hspace{1cm} (3.2)

The results support the energy efficiency of the proposed clustering method as the average energy levels of AGs is above that of member nodes across the network. Further, we note that the proposed method works on localized (2-hop) information to form clusters. This ensures that the method is scalable and the clustering times obtained for the 10, 15 and 20 node networks prove this.

In a large scale sensor network, energy-efficient techniques will limit the number of active nodes to a subset of all nodes so the network area is fully covered and the network remains fully connected. This observation means that the number of active two-hop neighbours for a given node will tend to remain fairly constant. This, in conjunction with the two-hop localised nature of the reclustering algorithm, will ensure that the reclustering time remains fairly constant and is independent of the total number of nodes in the network, as only active nodes will participate in the reclustering process. However, we do note that since a

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Average $M(v)$ for Network</th>
<th>Average $M(v)$ for AGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 nodes</td>
<td>5.5</td>
<td>5.6</td>
</tr>
<tr>
<td>15 nodes</td>
<td>8</td>
<td>8.25</td>
</tr>
<tr>
<td>20 nodes</td>
<td>10.5</td>
<td>13.95</td>
</tr>
</tbody>
</table>

Table 3.4: Energy levels of clusters and AGs
command to re-cluster from the BS is passed through AGs to nodes which are not necessarily in range of the BS, this transmission time is likely to increase as the network grows while the clustering time remains fairly stable.

One of the important differentiators between our work and that of Wu [Wu03] is that the set of chosen AGs will vary over time as we take into account the state of individual sensors with respect to their energy levels and node degree. This is especially useful because the number of active nodes available to join a cluster diminishes with time as nodes lose all energy or are compromised beyond recovery. It is also useful in terms of the proposed scheme being extensible to mobile sensor networks where each sensor will have a periodically changed node degree.

3.4.3 Simulations

We tested reclustering protocols on the NS-2 simulator as our experimental testbed platform. NS-2 is regarded as the most popular network simulator. The results are presented in Figure 3.8 and 3.9. We used different network sizes in order to test the performance of the algorithm.

We deployed the nodes in a 9x20 metre area. The network size was increased by 10 nodes starting from the 20 nodes. In each different network size, the simulation experiment was conducted 50 times.

From Figure 3.8, we can see the average metric value per network versus that of the selected aggregators. The results are also consistent with our hardware implementation. The average metric of AGs is always higher than the average
metric of networks. The average metric of AGs ranged from 8.5 to 11.5, while the average metric of networks ranged from 9.2 to 10.5. This can prove our reclustering protocols perform better for energy efficiency.

Figure 3.9 presents the average performance of the clustering algorithm on simulation. The result was averaged and plotted on the graph. We observed that the aggregator number increased with the network size as well. The aggregator number is roughly between 16 nodes and 110 nodes for the network size from 20 nodes to 180 nodes. In particular, we also tested the network size in 10 and 15 nodes with the same results as in Table 3.3.
CHAPTER 3. RE-CLUSTERING PROTOCOLS

Figure 3.9: Simulation Performance of the clustering algorithm

3.5 Summary

In this chapter, we have made important contributions to the recovery problems. We have presented a self organization plan and clustering procedure for sensor networks which is used to form closely connected small-size clusters and to connect clusters with each other. This structure can detect and respond to attacks effectively. In case some nodes die or are compromised, we can run this algorithm to reform the structure. In addition to this, we have also proposed a cluster partnership detection algorithm which uses the ‘overhearing’ features of nodes to detect compromise or attacks from inside or from outside the clusters. To identify the best-suited AGs for the reconfigured network, the clustering algorithm incorporates a metric that takes into account the energy levels of individual sensors as well as their connectivity to other nodes. This detection or ‘voting’
algorithm is also applied to quickly make a decision on which node or cluster has been compromised. Our cluster configuration is also amenable to efficient recovery techniques, ensuring that the network remains connected and that, where necessary, compromised nodes are recovered.

We have proposed a recovery mechanism for WSNs that enables fast and reliable recovery from security attacks by applying dynamic network reclustering. To identify the best-suited AGs for the reconfigured network, the clustering algorithm incorporates a metric that takes into account the energy levels of individual sensors as well as their connectivity to other nodes. Since the reclastered network does not include compromised nodes, the network can be kept operational. We have demonstrated that the proposed recovery mechanism comprising reclustering can be efficiently implemented on a sensor network testbed using TelosB motes. Our experimental results show that our recovery mechanism is suitable for low resource sensor devices and efficient both in terms of recovery time and scalability due to its decentralized approach.

The reprogramming techniques used to recover the compromised nodes are discussed in Chapter 4.

The little prince, who asked me so many questions, never seemed to hear the ones I asked him. It was from words dropped by chance that, little by little, everything was revealed to me. . .

—'The Little Prince’ by Antoine de Saint-Exupéry
Chapter 4

Reprogramming Protocols

In earlier chapters, we discussed the decision making process which assists the network in determining when a node has been compromised. A decision then occurs at the associated AG, or at the BS, to determine the response. In general, once a node has been identified as being compromised, a network will attempt to recover it by reprogramming it with the original code. If the node is presumed to be irretrievable because it is being managed by an attacker, it can then be quarantined from the rest of the network.

In order to maintain network life for as long as possible, many implementations use redundancy. The argument is, that deploying several times more nodes than needed (redundancy) ensures that necessary data continues to be gathered as nodes drop out of the network for various reasons (break-down, damage, attack etc.) [PP06]. Redundancy may be effective in a situation where nodes are cheap, deployment is in a remote, inaccessible area and the network lifetime is bounded.
CHAPTER 4. REPROGRAMMING PROTOCOLS

with stable deterioration. However, there are many situations in which one or
more of these conditions does not hold. For example, in a deployment of thou-
sands of nodes over a large geographic area, an entire section of nodes at a site
may be wiped out by a natural disaster resulting in no further data from that area
reaching the BS and no data critically needed at that time is provided. In this
section, we describe in detail two situations in which maintaining the network by
recovering nodes is a superior approach to redundancy.

The reprogramming methods described in this chapter are capable of implement-
tion in stationary, limited-resource networks.

Section 4.1 demonstrates the scenarios where the reprogramming could be used
and how the reprogramming works. Section 4.2 shows the testing results of how
the reprogramming performs. Section 4.3 summarizes the reprogramming work
we have done.

4.1 Design

Deluge [YYS03] is a reliable data dissemination protocol for large objects, such
as program binaries. Together with a bootloader, Deluge provides a way to re-
program sensor motes in a network. Since Deluge only supports network wide
reprogramming, we modified the dissemination engine of the protocol to individu-
ally address sensor nodes. This was done by replacing AM_BROADCAST_ADDR
parameter in the engine with the node ID of the node to be recovered. This mod-
ification allowed Deluge to disseminate the program binary to a specific compro-
mised node. The compromised node in our scenarios can still listen to the BS and respond to the messages. This will make the compromised node accept the reprogramming commands from the BS.

4.1.1 Scenario

Forty-five (45) nodes are distributed at ceiling points in a commercial tent complex set up for food during a large political rally. The sensors detect sound levels (a gunshot or explosion would be picked up as being exceptional) and vibration (a fight or someone running would be detected). Information is relayed to a BS computer in a building 15 metres from the complex. All sensors are within a 25 metre range and one hop of this BS. Figure 4.1 shows the set-up for scenario B.

The tent complex has seven sections:

- One main area where traffic is high and food is regularly being served (15 sensors).
- Three service areas; one used as a pantry, one as a kitchen and one as a food pick-up area for serving staff (3, 5 and 7 sensors respectively).
- Two smaller meeting room areas for private caucus by officials (5 sensors in each).
- One press area (5 sensors).

At set-up, the sensors are in the physical positions indicated while the clustering algorithm has logical clusters cutting across the tents.
The main tent has two AGs, M1 and M2, while each of the other tents has a single AG n, each case with subscript 1.

The event runs for three days. During the third day, someone reprograms three of the sensors in the main tent area to stop transmitting information.

The description below details the response of the sensor network. In the previous section, the entire set of sensor nodes was reprogrammed which possibly resulted in the loss of reliable data from uncompromised nodes. In this scenario, only those nodes identified as compromised are reprogrammed.

Day 3

9:25 - The BS assesses data received from the network and notices a diminished amount of data from the main tent. It requests the two AGs in this tent to initiate the voting algorithm in the cluster in order to determine the damage.

9:27 - M1 and M2 initiate the voting algorithm whose results identify that three member nodes have become inactive.

9:30 - One of the three nodes in the cluster managed by M1 and the other two in the cluster are managed by M2. Each aggregator then initiates a reprogramming of the suspect nodes.

9:31 - The three member nodes are functioning again.

The network resumes normal operations.
4.1.2 AG Reprogramming

This section introduces the network-based reprogramming algorithm for AGs. Recall that each node carries a unique ID which identifies it to the rest of the network.

Once a decision is reached by the BS to reprogram an AG with, for example, ID A, a message is broadcast to the network requiring that any node with ID A must reinstall the original AG. (In a normal situation, this code is the same used for all nodes but there may be exceptional circumstances where it is different for AGs). While all nodes receive the broadcast, only those identified by ID implement the reprogramming.

Several AGs can be reprogrammed simultaneously using this method, so that the cost of reprogramming is a flat rate, independent of the number of nodes involved.
4.1.3 Member Node Reprogramming

This section introduces the cluster-based reprogramming algorithm for member nodes.

Once a decision is reached by the BS or AG A to reprogram a member node of A’s cluster having ID T, for example, a message is then broadcast by A to the cluster requiring that any node with ID T must reinstall the original member node code in a normal situation, this code is the same used for all nodes but there may be exceptional circumstances where it is different for member nodes). While all nodes in the cluster receive the broadcast, only those identified by ID implement the reprogramming.

Several member nodes can be reprogrammed simultaneously by A using this method, so that the cost of reprogramming is a flat rate, independent of the
number of nodes involved.

4.2 Experiments and Evaluation

We again tested networks with various sizes in terms of the speed of reprogramming on TelosB. In the tests, the BS issued a reprogramming command to 1 node, 5 nodes and 10 nodes respectively. Start time was recorded at the instant the BS issued the reprogramming command. Finish time was recorded when the program was fully installed on the last node, which was indicated by rapidly flashing LEDs. Note that the entire set of nodes is reprogrammed in these tests. Although at this point in the project, we are able to identify and reprogram a single node in a cluster. These results will be presented shortly.
CHAPTER 4. REPROGRAMMING PROTOCOLS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>58.54</td>
<td>61.86</td>
<td>61.01</td>
<td>57.45</td>
<td>60.34</td>
<td>59.84</td>
<td>1.62</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>54.14</td>
<td>56.27</td>
<td>54.41</td>
<td>56.54</td>
<td>54.47</td>
<td>55.17</td>
<td>1.02</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>59.40</td>
<td>59.61</td>
<td>65.29</td>
<td>56.28</td>
<td>63.93</td>
<td>60.90</td>
<td>3.28</td>
</tr>
</tbody>
</table>

Table 4.1: Reprogramming the nodes with BlinkM.nc 2 metres from BS

We tested reprogramming with code of two different sizes. We chose the BlinkM.nc and GoldenImage.nc applications packaged with the TinyOS code. BlinkM.nc is a standard application used in reprogramming and was chosen because its size, with 50 lines of code, is indicative of the size of a program that would actually be used. The GoldenImage application, with 5 lines of code, provides support for network programming and was chosen for its size in order to compare with the program with more code.

We tested nodes placed at 2 meters, 11 meters and 21 meters, but assumed all were one hop away from the BS. For each distance, five rounds of testing were implemented in order to collect more accurate experimental data.

Table 4.1, Table 4.2 and Table 4.3 for GoldenImage.nc Application (5 lines of code).

Table 4.4, Table 4.5 and Table 4.6 for Blink application (50 lines of code).

The tables show the results of the reprogramming tests on TelosB; the first three
### Table 4.2: Reprogramming the nodes with BlinkM.nc 11 metres from BS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>60.32</td>
<td>64.71</td>
<td>72.74</td>
<td>79.19</td>
<td>65.93</td>
<td>68.58</td>
<td>6.63</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>62.42</td>
<td>55.76</td>
<td>60.63</td>
<td>58.39</td>
<td>56.28</td>
<td>58.70</td>
<td>2.53</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>54.78</td>
<td>59.75</td>
<td>56.37</td>
<td>60.03</td>
<td>57.48</td>
<td>57.68</td>
<td>2.00</td>
</tr>
</tbody>
</table>

### Table 4.3: Reprogramming the nodes with BlinkM.nc 21 metres from BS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>58.13</td>
<td>66.84</td>
<td>61.03</td>
<td>58.54</td>
<td>58.07</td>
<td>60.52</td>
<td>3.34</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>53.64</td>
<td>55.89</td>
<td>55.89</td>
<td>55.48</td>
<td>54.38</td>
<td>55.06</td>
<td>0.96</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>67.23</td>
<td>70.91</td>
<td>63.85</td>
<td>55.34</td>
<td>62.32</td>
<td>63.93</td>
<td>5.22</td>
</tr>
</tbody>
</table>

Table 4.2: Reprogramming the nodes with BlinkM.nc 11 metres from BS

Table 4.3: Reprogramming the nodes with BlinkM.nc 21 metres from BS
### Table 4.4: Reprogramming the nodes with GoldenImage.nc 2 metres from BS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>60.81</td>
<td>62.76</td>
<td>60.96</td>
<td>61.28</td>
<td>62.73</td>
<td>61.71</td>
<td>0.86</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>55.14</td>
<td>56.32</td>
<td>54.41</td>
<td>55.38</td>
<td>55.94</td>
<td>55.44</td>
<td>0.66</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>57.90</td>
<td>63.42</td>
<td>68.92</td>
<td>58.52</td>
<td>57.39</td>
<td>61.23</td>
<td>4.53</td>
</tr>
</tbody>
</table>

### Table 4.5: Reprogramming the nodes with GoldenImage.nc 11 metres from BS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>65.87</td>
<td>64.17</td>
<td>67.54</td>
<td>73.66</td>
<td>71.38</td>
<td>68.52</td>
<td>3.51</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>61.98</td>
<td>57.28</td>
<td>57.12</td>
<td>57.38</td>
<td>58.62</td>
<td>58.48</td>
<td>1.84</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>58.78</td>
<td>58.03</td>
<td>56.03</td>
<td>59.09</td>
<td>57.67</td>
<td>57.92</td>
<td>1.07</td>
</tr>
</tbody>
</table>

Table 4.4: Reprogramming the nodes with GoldenImage.nc 2 metres from BS

Table 4.5: Reprogramming the nodes with GoldenImage.nc 11 metres from BS
### CHAPTER 4. REPROGRAMMING PROTOCOLS

<table>
<thead>
<tr>
<th>Reprogramming Time</th>
<th>1st Round (secs)</th>
<th>2nd Round (secs)</th>
<th>3rd Round (secs)</th>
<th>4th Round (secs)</th>
<th>5th Round (secs)</th>
<th>Average (secs)</th>
<th>Std. Dev (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 node)</td>
<td>58.38</td>
<td>58.18</td>
<td>56.25</td>
<td>59.43</td>
<td>59.21</td>
<td>58.29</td>
<td>1.12</td>
</tr>
<tr>
<td>(5 nodes)</td>
<td>56.26</td>
<td>58.83</td>
<td>62.93</td>
<td>58.51</td>
<td>58.23</td>
<td>58.95</td>
<td>2.18</td>
</tr>
<tr>
<td>(10 nodes)</td>
<td>67.43</td>
<td>80.31</td>
<td>67.22</td>
<td>59.25</td>
<td>55.56</td>
<td>65.95</td>
<td>8.52</td>
</tr>
</tbody>
</table>

Table 4.6: Reprogramming the nodes with GoldenImage.nc 21 metres from BS with the smaller code and the last three with the larger code. In each case, the average reprogramming time and standard deviations are computed in the last column.

Ad hoc testing of reprogramming over more than 22 meters from the BS revealed that reprogramming is possible up to a maximum of approximately 55 meters. A natural assumption is that reprogramming should depend on the size of the code and the distance from the BS (or from the AG initiating the reprogramming). In these tests, all nodes were one hop from the BS, but the distances varied between 2 and 21 metres. The reprogramming protocol appears to be independant to difference of code size. In any event, it is unlikely to be useful to test with much larger code sizes, because in practice they would not be used.

However, other factors including reflection of the transmission off objects, temperature, humidity etc. might have also affected the results. In addition, the accuracy of the time clock, again a standard stopwatch, must also be considered.
a factor.

In any event, the average times in the above tables are very close, and we conclude that neither distance nor the number of nodes to be reprogrammed affects the essential reprogramming time of a single node.

Table 4.7 shows the results for the reprogramming tests on TelosB using the modified Deluge method. The performance tests were conducted using similar network topologies as the reclustering set up. However, for reprogramming, we assume that each node in the WSN is within range of the BS, while the converse is not necessarily the case. This is a much stronger assumption than that needed for reclustering. In the tests, the BS issued a reprogramming command to a single compromised node that was chosen at random in the network. Reprogramming time was calculated from the time the command was issued by the BS to when the node was fully functional. The results have been obtained by averaging over 30 runs with the compromised node being chosen at random from the network. Table 4.7 indicates no impact on reprogramming time as a function of network size.

Based on the observed reprogramming and reclustering times, we propose that network wide reclustering is triggered only when an AG is compromised. In the event of a member node being compromised, recovery can be restricted to isolating the node and reprogramming without the need for reclustering. Such an approach is both energy-efficient and minimizes the disruption of normal network operation.

In each case, time starts when the BS issues its first message and stops when
CHAPTER 4. REPROGRAMMING PROTOCOLS

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Reprogramming Time (secs)</th>
<th>Reclustering Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 nodes</td>
<td>63.20</td>
<td>87.67</td>
</tr>
<tr>
<td>15 nodes</td>
<td>85.33</td>
<td>87.79</td>
</tr>
<tr>
<td>20 nodes</td>
<td>99.54</td>
<td>87.21</td>
</tr>
</tbody>
</table>

Table 4.7: Reprogramming and reclustering the network using modified deluge

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Execution Time on TelosB (secs)</th>
<th>Execution Time on MicaZ (secs)</th>
<th>Std. Dev. on TelosB (secs)</th>
<th>Std. Dev. on MicaZ (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 nodes</td>
<td>87.12</td>
<td>90.48</td>
<td>0.34</td>
<td>0.43</td>
</tr>
<tr>
<td>6 nodes</td>
<td>87.28</td>
<td>90.29</td>
<td>0.36</td>
<td>0.39</td>
</tr>
<tr>
<td>10 nodes</td>
<td>87.67</td>
<td>90.27</td>
<td>0.38</td>
<td>0.41</td>
</tr>
</tbody>
</table>

Table 4.8: Performance of standalone reprogramming on TelosB and MicaZ

the last node verifies the hash message. The authentication time increased as the network size. Depending on the network size, the whole protocol can be completed in a reasonable time period between 150.87 seconds and 186.75 seconds.

4.3 Standalone Modified Deluge on TelosB and MicaZ

We would not expect the network size to affect the speed of standalone reprogramming on different hardware platforms. However in order to demonstrate it, we do a comparison on the reprogramming time with different network sizes. Table 4.8 shows the results for the reprogramming tests using the modified Deluge method on both TelosB and MicaZ.

In the tests, the BS issued a reprogramming command to a single compromised node that was chosen at random in the network. Reprogramming time was
calculated from the time the command was issued by the BS to when the node was fully functional. The results have been obtained by averaging over 30 runs, with the compromised node being chosen at random from within the network. We again observed that the reprogramming time is fairly stable.

The above results indicate that TelosB motes are a better overall choice for implementing our recovery algorithms than MicaZ motes. Our requirements for speed and sufficient RAM to execute appropriately appear to be the deciding factors.

4.4 Summary

In this chapter, we have proposed a recovery mechanism for WSNs that enables fast and reliable recovery from security attacks by applying reprogramming. Since the reclustered network does not include compromised nodes, the network can be kept operational while the comprised nodes can be recovered by executing the node reprogramming procedure. We have demonstrated that the proposed recovery mechanism comprising reclustering and reprogramming can be implemented on a sensor network testbed using TelosB motes. Our experimental results show that our recovery mechanism is suitable for low resources sensor devices and is efficient both in terms of recovery time and scalability due to its decentralized approach.

We derive the requirements and design for running the authentication protocols which are used to secure the recovery communication, in the next chapter.
My belief is that a program, in the absolute, means absolutely nothing. A program only means something *relative* to a certain intention, that must predate it, in one form or another.

—Jean-Raymond Abrial
Chapter 5

Authentication Protocols

In this chapter we add authentication, showing how the addition of hash function checks can prevent several types of attacks aimed at interfering with recovery mechanisms. Complete descriptions of the authenticated recovery and recluster ing protocols are given, along with a discussion of what type of security is added. We then discuss options for hash functions and compare the performance with the standard SHA-1 with a little used hash function, Rabin, based on an idea of Shamir. We also compare the performance of our protocols with that of other researchers in the area.

In sensor networks, generally the sensor nodes communicate effectively with broadcast communication in short-range space. The wireless broadcast communication is exposed to security risks, or to put it more directly, an adversary can eavesdrop and alter communication messages, and insert malicious messages. For preventing these attacks, encryption of the communication data and mutual
authentication between sensor nodes is needed.

In our whole recovery system, both reclustering and reprogramming require the nodes to send the command respectively in order to initiate either of the processes. To prevent the replayed attacks, a cryptographically strong hash function has to be implemented to protect data integrity. The receiver verifies the packet was not modified in transit using the same hash function and then proceeds to the next recovery step.

In this chapter, I implement security components for message hash and authentication using Rabin hash function instead of using SHA-1, which is offered in [LCK05] and [KHYM08].

Section 5.1 discusses the design of authentication protocols. Section 5.2 presents the Rabin hash function used in our authentication. Section 5.3 documents the issues and restrictions encountered during the implementation. Section 5.4 evaluates the authentication protocols on both TelosB and MicaZ hardware platforms, and Section 5.5 concludes the authentication protocols.

5.1 Design

5.1.1 Objectives

Due to the development of many existing authentication methods, we decided to simply choose one as our authentication scheme. Basically we have to compare a few authentication methods in order to decide which one is low cost and high
efficiency. In the context of the recovery, we have to consider if this authentication can quickly address attacks. In the mean time, we investigate the energy efficiency of the authentication protocols especially for a sensor network. The standard interfaces for the Rabin and SHA-1 are developed for the programmers to use in the future development based on those two hash function protocols.

Besides several prior objectives, there are also four main aims for the design architecture:

- **Access control**
  Only authorized nodes should be able to participate in the network. Authorized nodes are designated as those nodes that possess the shared group key.

- **Integrity**
  A message should only be accepted if it was not altered in transit. This prevents, for example, man-in-the-middle attacks where an adversary overhears, alters, and re-broadcasts messages.

- **Confidentiality**
  Unauthorized parties should not be able to infer the content of messages.

- **Ease of use**
  Taking into account the diversity of sensor network users, the hash function should not be difficult to use.

We hope to provide a communication interface that provides the above four goals and are compatible with other security applications.
5.1.2 Assumptions

In order to implement recovery, we have a set of assumptions regarding the make-up of the network. In order to ensure that the recovery protocols are effective against attacks which try to prevent them, we add an authentication procedure to both the reprogramming and reclustering protocols. Authentication is established by means of secrets shared by each node and the BS.

1. We assume the existence of a globally unique ID for each sensor node. The BS keeps track of all IDs.

2. We assume that every node in the WSN is in transmission range of the BS, while it is not necessarily the case that each node can transmit to the BS. Because some nodes may not be in range of other nodes, all reprogramming and reclustering commands will therefore be run through the BS.

3. We assume that each sensor node has a secret value which can be used for authentication. This secret is known only to the BS and the node, and is allocated when the WSN is set up. We also assume that this secret is stored in a tamper-resistant section of the node. (Note, that while tamper resistance might be a viable defence for physical node compromise for some networks, we do not see it as a general purpose solution. Thus, we assume that an attacker may be capable of retrieving the entire set of codes residing in the node. However, in order to implement authentication, it is important to have some item residing on the node which cannot be retrieved or identified as being a ‘secret’).
4. We assume that the BS is trustworthy in the sense that it is never compromised and always behaves correctly. (Most, but not all routing protocols depend on nodes to trust messages from BSs).

5. We assume that an essentially infinite timer is available to each of the nodes and the BS. (In practice, timers on nodes may overflow and re-use previous times which invalidate our protocol).

6. We assume that the same hash function code is programmed into each node at set-up. This code will be used for authentication.

5.1.3 Attack Scenarios

In our threat model, we assume that an attack can be as powerful as a desktop PC and therefore, is capable of unlimited computation, can compromise nodes and can introduce malicious nodes. An attacker may also eavesdrop on the network, picking up and changing un-encrypted messages. In case the BS broadcasts a message to the entire network, an attacker will not be able to stop receipt of the message by the network. However, it may stop receipt by several nodes.

Our focus in this chapter is on ensuring that the recovery process runs smoothly, unimpeded by attacks. In recovering a network, we therefore consider the following attacks:

1. Replay of reclustering or reprogramming messages.

   Because we add a timer to these messages, any replayed message will be discarded.
2. Obfuscation and Denial of Service via unauthorized commands.

The authentication procedure implemented for the recovery protocols reclustering and reprogramming, ensures that only the BS can send these commands, or in the case of an AG request to the BS, ensures that the AG is indeed legitimate.

3. Impersonation attack.

An attacker attempting to impersonate the BS or an AG will not have the appropriate secret (shared with the BS) and so will not be able to authenticate itself.

5.2 Approaches

Hash functions are used in our protocols in the areas below and here we discuss some options for such functions which can be used in a limited resource situation.

At set up, a hash function is employed to hash the program code in each sensor along with its ID. This value is divided between two other nodes in the network. The hash function chosen will be SHA-1 as a preferred hash in government authentication protocols [IC07a]. SHA-1 was used by [SV06] in their paper on node recovery to detect compromised nodes. In addition, a hash chain was implemented by [LB08] for detection, and SHA-1 is an appropriate function for this chain.

A recent proposal by A. Shamir [Sha08b], and implemented by [GSQ08], is a hash function which is a variation of the RSA scheme, designed specifically for
implementation with low resource RFID technologies. From [GSQ08]: "SQUASH is based on the one-way function $c = m^2 \mod n$ coming from the Rabin cryptosystem. To make it secure, the binary length of $n$ must be at least 1000 bits long. [Sha08b] suggests that using a 64-bit non-linear feedback shift register to generate $m$, a not yet factorized Mersenne’s number $(2^x-1)$ as modulus $n$, and to send out the bits of $c$ without storing them. This process avoids storing three 1000-bit long numbers. The multiplications are achieved by on-the-fly convolutions, sending each bit as soon as it is computed. Consequently, the only needed memory aims to store the carry of the previous steps in the convolution. For the output, a window of 32 or 64 (or more) bits is used. It yields a hash function with inputs of 64 bits that is scalable in output.

In this chapter, we propose implementations designed in order to minimize the resources, possibly at the cost of an increased execution time. The target device is a Xilinx Virtex-4 XC4VLX200-10 FPGA. The algorithm recommended by Adi Shamir has 64 bits in input and 32 in output, and $n = 2^{1277} - 1$. To reduce the hardware cost, we minimized the number of registers in the implementation data and control part. On the XC4VLX200, the design results require 377 slices.

The full execution time to produce 32 bits is 63,250 clock cycles at 222 MHz, so we reach a throughput of 112,300 bits per second. We also implemented the algorithm with other size numbers. For 128 bits in input and 64 in output, we get 619 slices and 104,114 clock cycles at 206 MHz. In general, the length of the output influences the execution time, while the length of the input influences the number of registers and slices.”

We will examine the implementation of this hash function as a comparative al-
ternative to SHA-1.

5.2.1 Rabin Algorithms

In order to implement the Rabin and SHA-1, we also have to understand the authentication and hash function [ZFB05].

The authentication scheme is shown as follows:

- Sender and receiver share a 64 bit secret S.

- A 64 bit challenge R is issued by the Sender.

- Sender returns H (R xor S).

- Receiver verifies sender’s knowledge of S by computing H (R xor S).

By simplifying the Rabin hash function, a few steps have been summarized as follows:

\begin{algorithm}
\caption{The Rabin Hash Function}
\begin{itemize}
    \item Returns a subset of bits from \( c \) where \( c = m^2 \mod n \).
    \item Words of \( m \) are generated from an IV = R xor S using a LFSR with a length of processor word.
    \item A modulus of the form \( 2^{k-1} \) allows the calculation of \( m^2 \mod n \) with no modular operations since \( 2^{k-1} \mod n \).
    \item Calculating a bit requires carry-in which is approximated by computing the carries of up to 11 previous bits. In addition, to carry some shifts is required to align the bits in chunks.
\end{itemize}
\end{algorithm}

However, we are unable to provide a powerful LFSR in order to implement the SQUASH protocol due to the limited resource of sensor nodes. We decided to
implement a simple version of the SQUASH instead. Our authentication protocol is only used in two areas of the communication: the BS to the member nodes and the AGs to the BS involving a third party. When the BS sends the commands to the nodes, it transmits the part of the hash value $H(R, S, CMD)$ where $H$ is some publicly known hash function, the command $CMD$ and random number $R$ respectively to the nodes. The command $CMD$ and the random number $R$ has to be sent because the adversaries could play replayed attacks by capturing the whole hash value without doing any computation. After that, the receiver simply calculates the hash value in the same way as the sender does to verify the message it receives. Once the authentication process is successful, the appropriate recovery methods can be taken afterwards. If the nodes to be recovered are not covered by the transmission range of the BS, the AG in charge will report to the BS. It transmits the part of the hash value $H(R, S, CMD)$, the command $CMD$, random number $R$ and the IDs of the nodes to be reprogrammed. When the BS verifies the message from the AGs, it will either recluster or reprogram the nodes accordingly.

The simple version of the Rabin is described as follows:

- Generate a message $m$ by using a particular choice of mixing function $M=H(R,S,CMD)$. The $R$, $S$ and $CMD$ are all 64 bit variables as the 64 bit is the maximum bit variable which is supported by TelosB motes and most sensor nodes.

- Use the numbers of the modulus form $n=2^k-1$ with $k=59$ which can make sure the $n$ is 64 bit value. $k=59$ would be fractionally more difficult to
factor in a reasonable of 0 to 64 bits.

- Compute the cipher text \( c = m^2 \mod n \).

- Extract a subnet of 32 bits (10th bit to 42nd bit) from \( c \), as we know the exact location of this window within \( c \) is not important, and send this window of consecutive bits to the receiver later.

### 5.2.2 SHA-1 Algorithms

SHA-1 has been implemented by researchers previously. However, we are unable to acquire the NesC source code from them so we developed this algorithm on our own based on the C source code on the Internet [SHA].

### 5.3 Technical Problems and Experience of Implementation

In this sections, I will discuss a few technical problems I received during the development and some experience related to authentication development in TinyOS.

First of all, the standard random number used in TinyOS, like many in computer-science, are pseudo-random. That is, they use some feedback of previous state(s) to generate a new state. There are some rather complex approaches, such as Mersenne-Twister [MN98], but there are also some rather simple approaches like RandomLfsrC component in TinyOS.
In TinyOS 2.x, the random number generation is to start at 'new places' in the sequence generated by the random number generator. Looking at RandomMl-cgC, it is seeded with TOS_NODE_ID. That is, each time it starts, it uses the TOS_NODE_ID to get the initial state. At least multiple nodes should start with the same sequence.

Since we use the node ID, also known as TOS_NODE_ID in TinyOS as the seed for the random number generator, we will keep getting the same random number over and over again if the TOS_NODE_ID remains the same. As our assumption is every node has a fixed node ID. After all, we have to decide to use local time instead. The local time can be activated after the node is booted. We assume the time is infinite in the system, so every time you call the LocalTime.get() command in TinyOS, the system will acquire a different time stamp. This time stamp is also sent to the receiver to calculate the H (R, S, CMD) value due to no synchronized time mechanism on the LocalTime component in TinyOS.

Secondly, Unsupported Math.h library on TelosB platform is another serious problem in our Rabin authentication implementation. Even if math.h library is included in TinyOS, the TelosB platform doesn’t support it. The function such as MOD(), SQRT(), POW() etc. cannot be used in the code. It could be another possible direction for TinyOS Alliance to focus on in the future. I implemented our own math functions which can be used in the Rabin protocol.

Thirdly, the limitations of the message structure in TinyOS, also referred as message_t, needs to be increased as the default size of a message is 28 bytes including 11 bytes header data. Therefore, only 17 bytes can be used for data payload. In both SHA-1 and Rabin protocols, the message to be sent exceeds the
default size. It can raise some serious problems during compilation time and can be very hard to detect. This means the default size of the message needs to be adjusted in order to proceed with larger message size.

Next, some knowledge of bit operations, for example, bitwise operation, is also required in order to implement the hash function. It is very common for developers to manipulate the bits frequently in order to work on low-power hardware.

Furthermore, the basic debugging tool, printf(), doesn’t work well in printing the 64 bits message. The solution to this problem is to print out the high 32 bit and low 32 bit respectively. The default printing configuration used to print out debugging the message in which the message size is bigger than the default size is only to print the low bits in default. So when the system needs to display the high bits, it has to shift the message 32 bit left and then call the printf() function.

The last problem I encountered during the development is a bug triggered by the _nesc_hton_uint64 (conversion used for nx_uint64_t) when you want to transmit an nx_uint64 type of message data over the networks. This is an internal compiler bug of msp430-gcc 3.2.3. It only happens to the MSP430 compiler. As far as we are aware of, this bug has not been officially fixed. So we are constrained to only using the maximum 32-bit message size in order to test protocol versions of hash functions.

To sum up, there may be still a lot of issues related to TinyOS, especially in authentication, as this operating platform is still in development. In our project, we have, however, found some workarounds to bypass the above unresolved existing technical issues. With all the experience we had, it will definitely benefit other
researchers when they develop any authentication protocols in TinyOS.

5.4 Experiments and Evaluation

We use two well-known functions, SHA-1 and Rabin, for the purposes of comparison.

5.4.1 Hash Function Performance

SHA-1 is a preferred hash in government authentication protocols [IC07b], [oST02] and has been employed by several researchers in TinyOS applications [KHYM08], [LCK05]. Our second choice is the Rabin encryption system [Opp96] adapted for use as a hash function, as proposed by Shamir in [Rab79]. While Shamir suggests an adaptation of Rabin’s scheme to what he calls SQUASH, based on improved methods of computing the hash output, our implementation is constrained by TinyOS requirements, thus we use smaller values than those proposed to ensure the security of SQUASH. These smaller values do not warrant use of the improved SQUASH computations, and so we use Rabin’s scheme as proposed in [Opp96]. Shamir points out [Rab79] that, when used as a hash function, the Rabin encryption scheme can be implemented securely on fewer bits than needed for encryption. The drawback of Rabin’s scheme is that up to four input messages can result in the same ciphertext. Since we use the scheme only for authentication based on a small set of standard message inputs, this does not pose a problem for us.


<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution Time (sec)</th>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 (8 bits)</td>
<td>7</td>
<td>126</td>
<td>3892</td>
</tr>
<tr>
<td>SHA-1 (32 bits)</td>
<td>7</td>
<td>126</td>
<td>3896</td>
</tr>
<tr>
<td>SHA-1 (64 bits)</td>
<td>7</td>
<td>126</td>
<td>3900</td>
</tr>
<tr>
<td>Rabin (64 bits)</td>
<td>2</td>
<td>46</td>
<td>1870</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of the Rabin and SHA-1 hash functions

It is worthwhile to compare, independently of the protocols, the performance of our two hash functions. While SHA-1 is a standard choice, it is clear from Table 6.3 that Rabin is faster with smaller RAM and Read-only Memory (ROM).

SHA-1 and Rabin are implemented as follows on TinyOS Version 2.1. For comparison, we take a data input of 8 bits, 32 bits and 64 bits for SHA-1 and 64 bits for Rabin. As shown in Table 5.1, for SHA-1, the code consumes 126 bytes of RAM, 3892 bytes for 8 bits, 3896 bytes for 32 bits and 3900 bytes for 64 bits of ROM, and takes approximately 7 ms to produce a 160-bit hash of different message sizes. Rabin produces a consecutive 32-bit hash window for a 64-bit calculated hash. The code consumes 46 bytes of RAM and 1870 bytes of ROM, and takes approximately 2 ms to hash a message of 64-bits.

Lee, Choi and Kim [LCK05] have introduced a Hash component for TinyOS, designed to be used as an independent hash module with TinySec. According to Lee, Choi and Kim, this hash module accepts any hash function, and can be implemented on 8, 16 and 32 bit words.

We compared time and memory for our implementation of SHA-1 against theirs and also against an implementation in [KHYM08]. While we found that our results, from Table 6.3, are better than those in [LCK05] and [KHYM08], for 8
or 32 bit input, we note that their research does not specify the TinyOS version used. We believe that the difference is due to their use of an earlier version of TinyOS or different coding structure or techniques.

### 5.4.2 Hash Function Comparison on TelosB and MicaZ

It is worthwhile to independently compare the performance of the protocols for our two hash functions on different platforms.

SHA-1 and Rabin are implemented alone as follows on TinyOS Version 2.1. For comparison, we take data input of 8 bits, 32 bits for SHA-1 and 32 bits for Rabin.

As shown in Table 5.2, on TelosB motes, for SHA-1 the code consumes 126 bytes of RAM, 3892 bytes for 8 bits, 3896 bytes for 32 bits and 3900 bytes for 64 bits of ROM, and takes approximately 7 ms to produce a 160-bit hash of different sizes of messages. On MicaZ motes, the SHA-1 code consumes 130 bytes of RAM and 4234 bytes of ROM for both 8 bits and 32 bits, and takes approximately 8 ms.

Rabin produces a consecutive 32-bit hash window for a 32-bit calculated hash. On TelosB, the code consumes 46 bytes of RAM and 1870 bytes of ROM, and takes approximately 2 ms to hash a message of 32-bits. The code of Rabin on MicaZ consumes 302 bytes of RAM and 4980 bytes of ROM, and takes approximately 3 ms to hash a message of 32-bits.

While SHA-1 is a standard choice, it is clear from Table 5.2 that Rabin is faster on both platform studies, and uses less ROM and RAM than SHA-1 when implemented on TelosB motes. Since MicaZ has less RAM than TelosB, Rabin would
Table 5.2: Performance of standalone hash functions on different platforms

appear to be an unsuitable choice for implementation on MicaZ motes.

5.4.3 Performance Comparison with the Work of Other Researchers

In this section, we compare our results with those who have developed authentication protocols for a similar situation. While hash functions have been implemented in authentication by a number of authors, the protocols used vary significantly and for the most part are not comparable.

However, Benenson, Gedicke and Raivio in [ZBR05], establish an authentication protocol for a WSN, which in many ways has the same goals as our reprogramming protocol without a key-discovery phase; that is a user in a fixed location broadcasts their identity and a certificate, then a node responds with a nonce.
The user replies with a hash. Theirs is based on elliptic curve cryptography, but, like ours, the hash function SHA-1 on 64 bits was used to hash the identities of the sender and receiver along with a random number.

We assume the BS can be trusted indefinitely. 1 hash and 2 verifications are needed while in our protocol only 1 hash and 1 verification were needed. The authors of [ZBR05] rely on the existence of a third party trusted certificate authority while we assume that only the BS is trusted. In summary, we avoid the need for a trusted third party and reduce the number of verifications.

Benenson, Gedicke and Raivio implemented their protocol on five TelosB motes using TinyOS. The hash function SHA-1 on 64 bits was used in [ZBR05] to hash the identities of the sender and receiver along with a random number. We see in Table 5.3 that our protocol is more than 10 times faster and requires almost 50% less in terms of memory requirements. Their execution times and the space needed are shown along with ours in Table 5.3. They point out that the execution time is disappointing due to the long execution times of the elliptic curve cryptography routines. Note also, that in their protocol, motes must be capable of executing both symmetric and public key cryptographic protocols and

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution Time (sec)</th>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benenson, Gedicke, Raivio [ZBR05]</td>
<td>440</td>
<td>2000</td>
<td>45500</td>
</tr>
<tr>
<td>Our protocol III with SHA-1</td>
<td>30</td>
<td>1454</td>
<td>26340</td>
</tr>
<tr>
<td>Our protocol III with Rabin</td>
<td>30</td>
<td>1220</td>
<td>21686</td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of two implementations of protocol III
must be able to securely store secret keys. Thus, our protocol provides the same level of authentication with a more lightweight approach.

In all schemes, the use of authentication needs some kind of key and the question of key storage is critical to maintaining a WSN. In virtually all known schemes for WSNs, keys are stored on nodes and thus require a tamper resistant location or an assumption that an attacker is unable to retrieve the keys. An exception to this is given, for example in the paper [DHCC05], where the authors use public key cryptography for authentication in which the private key is stored on a PC. A hash chain is produced by the PC and the nodes must verify all messages from the PC itself. This is a time-consuming undertaking and needs each node to be able to communicate with the PC at all times. In addition, it is important that messages be received sequentially so that they can be reconstructed. In an attack scenario, this is not an assumption that can be made.

In low resourced networks which are deployed in remote areas without the possibility of energy renewal or node replacement, the best one can achieve is relatively longer life. However, in situations where it is possible to refresh the network, the secrets used for authentication or encryption should also be refreshed.

5.5 Summary

In this chapter, we focus on recovery after an attack has been detected and provide lightweight protocols for reclustering and for reprogramming nodes in a WSN. We provide a message authentication protocol for limited resource sensor net-
works enabling the network to securely implement recovery strategies efficiently and effectively. The protocol is based on hash functions and we compare the performance of two well-known lightweight hash functions, SHA-1 and Rabin. We demonstrate that our authentication protocol can be implemented efficiently with TelosB motes in comparison with existing protocols.

The next chapter illustrates how to integrate the three techniques we have discussed in this chapter into a recovery mechanism.

I conclude that there are two ways of constructing a software design:

One is to make it so simple that there are \textit{obviously} no deficiencies and the other way is to make it so complicated that there are no \textit{obvious} deficiencies.

—Tony Hoare
Chapter 6

Recovery Protocols in Stationary WSNs

In this chapter, we describe each of the following recovery protocols in detail.

- Reprogramming Protocol from the BS to a Compromised Node.
- Reprogramming Protocol from an AG to the BS on Behalf of a Compromised Node.
- Re-Clustering Protocol from the BS to the Network.

Section 6.1 discusses each of the recovery protocols in details. Section 6.2 follows with the comprehensive experiments and evaluation. Section 6.3 gives a summary on recovery protocols.
6.1 Recovery Protocols

In this section, we describe each of the protocols in detail. In each case, a node with ID n contains secret $S_n$. R represents a random value but is in fact the local time obtained from the LocalTime.get() command in TinyOS. M represents a message to reprogram or re-cluster, or a request that another node be reprogrammed. All messages transmitted include node ID of both sender and receiver, including that of the BS.

A simple hash function check achieves authentication of a message. We use two well-known functions, SHA-1 and Rabin for the purposes of comparison. SHA-1 is a preferred hash in government authentication protocols [IC07b], [oST02] and has been employed by several researchers in TinyOS applications [KHMY08], [LCK05]. Our second choice is the Rabin encryption system [Rab79] adapted for use as a hash function as proposed by Shamir in [Sha08a]. While Shamir suggests an adaptation of Rabin’s scheme to what he calls SQUASH, based on improved methods of computing the hash output, our implementation is constrained by TinyOS requirements, and so we use smaller values than those proposed to ensure the security of SQUASH. These smaller values do not warrant use of the improved SQUASH computations, and so we use Rabin’s scheme as proposed in [Rab79]. Shamir points out [Sha08a] that, when used as a hash function, the Rabin encryption scheme can be implemented securely on fewer bits than needed for encryption. The drawback of Rabin’s scheme is that up to four input messages can result in the same ciphertext. Since we use the scheme only for authentication based on a small set of standard message inputs, this does not pose a problem
for us.

We take data input of 8 bits, 32 bits and 64 bits for SHA-1 and 64 bits for Rabin.

6.1.1 Reprogramming Protocol from the BS to a Compromised Node

Because the BS monitors network communications, it is able to detect an attack. The BS deals with it in the early stages by authenticated reprogramming and reclustering, with the objective to prolong the life of the WSN. If over 50% of the nodes were compromised, this would be deemed a failure.

In Figure 6.1, the BS has determined that node with ID n needs reprogramming. If a set of nodes is to be reprogrammed, each node must receive a separate message as each contains the secret known only by that node and by the BS. H represents a hash function. We assume that n, M and R are the appropriate size for input to H. Figure 6.2 explains the following steps in the message diagram.

1. BS XORs the secret $S_n$, the reprogramming message M and the local time R to obtain m.

2. The BS computes $H(m) = c$.

3. The BS transmits c, M and R which are received by n.

4. The node n recomputes $H$ of the XOR of n with $S_n$, M and R and checks that it is c.
Figure 6.1: Authenticating a reprogramming request

Figure 6.2: Message diagram of authenticating a reprogramming request
5. If the check is ‘true’ AND the time R has not been used in a reprogramming request earlier, the node reprograms.

6. The node updates its secret to $S_n = m$.

7. The node informs the BS that it has successfully reprogrammed and the BS then updates the node’s secret in its table.

### 6.1.2 Reprogramming Protocol from an AG to the BS on Behalf of a Compromised Node

In Figure 6.3, an AG A has determined that one of the nodes in its cluster must be reprogrammed. Since reprogramming is intensive from the initiator side, only the BS can implement it. Thus, node A requests reprogramming from the BS. The following steps are also interpreted in Figure 6.4.

1. A retrieves the ID $n$ of the node to be reprogrammed.

2. A XORs $n$, its own secret $S_A$, the request for reprogramming message $M$ and the local time $R$ to obtain $m$.

3. A computes $H(m) = c$.

4. A transmits $c$, $n$, $M$ and $R$ to the BS.

5. The BS retrieves $S_A$, recomputes the hash and checks if it is $c$.

6. If the check is ‘true’ AND the time $R$ has not been used in a reprogramming request earlier, the BS initiates Protocol 6.1.1.
Figure 6.3: Message diagram of authenticating a reprogramming request from an AG

Figure 6.4: Authenticating a reprogramming request
6.1.3 Re-Clustering Protocol from the BS to the Network

Reclustering may need to be implemented after an attack on a WSN in case several nodes are no longer trustworthy. The BS makes this decision based on information gathered from the network. Once it has been made, it sends a reclustering message to all AGs to begin the procedure. In this protocol, we describe the authenticated message to re-cluster which must be sent to each node separately, identifying it by its ID and its secret. The following steps are also illuminated in Figure 6.5.

1. The BS retrieves the secrets $S_1, S_c$ of each node and for each node ID n, XORs its secret $S_n$, the reclustering message $M$ and the local time $R$ to obtain $m_n$.

2. The BS then computes $H(m_n)$ for each n and transmits it with $M$ and $R$ to the corresponding node n.

3. Each node XORs its secret $S_n$, $M$ and $R$ to obtain $m_n$.

4. Each node computes $H(m_n)$ and compares with the messages received from the BS. If there is a match AND if no such message with time $R$ has been used in a reclustering request earlier, n accepts this as a valid reclustering message.

5. Once all nodes have received and verified such a message, reclustering commences.
6.2 Experiments and Evaluation

When running this protocol, the timing must be managed. A Timer.startOneShot(2000) command is set to fire in 2000 ms from the time of invocation at the BS. This cancels any previously running timer and will only fire once then stop. This gives enough time for the node in the network to receive and verify the hash message before the BS starts another timer. The protocol is complete when the last node verifies the hash message.

We tested all three protocols on TinyOS Version 2.1, with the results presented in the following tables. Table 6.1 shows the time, ROM and RAM used by the reprogramming protocol for four applications of the hash function. Although the execution time does not vary, Rabin uses only 83% to 85% of the RAM and ROM
Primitives | Execution Time (sec) | RAM (bytes) | ROM (bytes)
--- | --- | --- | ---
SHA-1 (8 bits) | 6 | 1230 | 19802
SHA-1 (32 bits) | 6 | 1262 | 20178
SHA-1 (64 bits) | 6 | 1334 | 21034
Rabin (64 bits) | 6 | 1100 | 17972

Table 6.1: Performance of a reprogramming protocol from the BS to a compromised node

Primitives | Execution Time (sec) | RAM (bytes) | ROM (bytes)
--- | --- | --- | ---
SHA-1 (8 bits) | 10 | 1262 | 20556
SHA-1 (32 bits) | 10 | 1296 | 20948
SHA-1 (64 bits) | 10 | 1368 | 21796
Rabin (64 bits) | 10 | 1142 | 18830

Table 6.2: Performance of a reprogramming protocol from an AG to the BS on behalf of a compromised node

The same distinctions are apparent when running Protocol 6.1.2. Again, while execution time does not vary, ROM and RAM for Rabin are only 84 to 86% that of SHA-1 on 64 bits.

Protocols 6.1.1 and 6.1.2 were run 100 times with almost identical results. Table 6.1 and Table 6.2 average these results. For Protocol 6.1.3, because the distribution of messages needs to be associated with timers, the implementation is significantly more complex than that for the previous protocols, therefore we ran this protocol only five times. Table 6.3 presents the average results of running Protocol 6.1.3 five times with a set of 5 TelosB motes. In each case, time starts when the BS issues its first message and stops when the last node verifies the hash message.

Once again, execution time does not vary. We believe this is due to the actual
### Chapter 6. Recovery Protocols in Stationary WSNS

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution Time (sec)</th>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 (8 bits)</td>
<td>30</td>
<td>1350</td>
<td>22978</td>
</tr>
<tr>
<td>SHA-1 (32 bits)</td>
<td>30</td>
<td>1382</td>
<td>23968</td>
</tr>
<tr>
<td>SHA-1 (64 bits)</td>
<td>30</td>
<td>1454</td>
<td>26340</td>
</tr>
<tr>
<td>Rabin (64 bits)</td>
<td>30</td>
<td>1220</td>
<td>21686</td>
</tr>
</tbody>
</table>

Table 6.3: Performance of a reclustering protocol from the BS to the network

The execution time of the hash function being significantly smaller than the rest of the protocol and therefore has negligible influence. It may be that in a large network with thousands of nodes, the execution time of the hash function begins to play a role.

### 6.3 Implementation Platform Comparison

All of the protocols developed in our work to date were performed on the TelosB platform with 16 bit processors. Since one of our targets is low-resource recovery, we proposed to compare the TelosB platform with the smaller MicaZ one to see where problems in implementation might occur. Table 6.4 shows the difference in resourcing of these two platforms. MicaZ does not support 64 bit computation as it has only 8 bit processors. It also has a much smaller RAM size. In compensation, its ROM size is significantly larger than that of TelosB. Thus, we might expect that some of our protocols might be adaptable to MicaZ by reducing computational overhead and increasing storage in ROM.

We attempted to implement the reclustering algorithm on MicaZ and found that it requires approximately 12620 bytes of RAM. This is not feasible due to the low RAM size. In contrast, TelosB can run reclustering results within the 10K
Table 6.4: Hardware performance comparison of TelosB and MicaZ motes

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Processor Bits</th>
<th>Integer Computation Bit Support</th>
<th>RAM (KB)</th>
<th>ROM (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TelosB</td>
<td>16</td>
<td>64</td>
<td>10</td>
<td>48</td>
</tr>
<tr>
<td>MicaZ</td>
<td>8</td>
<td>32</td>
<td>4</td>
<td>128</td>
</tr>
</tbody>
</table>

RAM space since it only consumes 9870 bytes.

It is also clear that we were unable to test the 64 bit hash function on the MicaZ platform, so this is not included in the results below.

We tested all three protocols on TinyOS Version 2.1 and on MicaZ with the results presented in the following sections.

6.4 Protocol Comparison between TelosB and MicaZ

Table 6.5 shows the time, ROM and RAM used by the reprogramming protocol for three applications of hash function on both TelosB and MicaZ platforms. In each case, the execution time on TelosB is approximately 20% less than on MicaZ. This is to be expected because of the TelosB capacity.

Rabin uses only 83 to 85% of the RAM and ROM used by SHA-1 on 32 bits on TelosB, while on MicaZ, more RAM is needed for Rabin than for SHA-1, while marginally less ROM size is needed.

Comparing SHA-1 on 8 bits and 32 bits, the RAMs used on TelosB are 29 bytes and 31 bytes more than on MicaZ respectively, thus MicaZ is using approximately...
CHAPTER 6. RECOVERY PROTOCOLS IN STATIONARY WSNS

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution time on TelosB (secs)</th>
<th>Execution time on MicaZ (secs)</th>
<th>RAM on TelosB (bytes)</th>
<th>RAM on MicaZ (bytes)</th>
<th>ROM on TelosB (bytess)</th>
<th>ROM on MicaZ bytess</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 (8 bit)</td>
<td>6.26</td>
<td>7.51</td>
<td>1230</td>
<td>1201</td>
<td>19802</td>
<td>20798</td>
</tr>
<tr>
<td>SHA-1(32 bit)</td>
<td>6.35</td>
<td>7.53</td>
<td>1262</td>
<td>1231</td>
<td>20178</td>
<td>21336</td>
</tr>
<tr>
<td>RABIN(32 bit)</td>
<td>6.22</td>
<td>7.46</td>
<td>1100</td>
<td>1265</td>
<td>17872</td>
<td>20636</td>
</tr>
</tbody>
</table>

Table 6.5: Performance of a reprogramming protocol from the BS to a compromised node on different platforms

98% of the RAM used by TelosB. Since MicaZ has much less RAM than TelosB, this is not a particularly useful situation.

Similar distinctions are apparent when running Protocol 6.1.2, the reprogramming protocol (see Table 6.6). In all cases, the execution time on TelosB is appropriately 13% less than on MicaZ.

For SHA-1, slightly more RAM, but slightly less ROM are needed for TelosB than for MicaZ. This is a good situation for MicaZ’s capacity. For Rabin, a different situation exists; TelosB needs less RAM (87.5%) and less ROM (83%) than MicaZ.

Protocols 6.1.1 and 6.1.2 were run 50 times with no substantial difference in results. Table 6.5 and Table 6.6 are the averages of the results.

For protocol 6.1.3, and BS reclustering, the distribution of messages needs to be
Table 6.6: Performance of a reprogramming protocol from an AG to the BS on behalf of a compromised node on different platforms

associated with timers because the implementation is significantly more complex than that for the previous protocols, therefore we only ran this protocol five times. Table 6.7 presents the average results of running protocol 6.1.3 five times with a set of 5 TelosB or 5 MicaZ motes. In each case, time starts when the BS issues its first message and stops when the last node verifies the hash message.

In all cases, TelosB performs better than MicaZ. We believe this is because the actual execution time of the hash function is significantly smaller than that of the rest of the protocol and therefore has negligible influence. It may be that in a large network with thousands of nodes, the repeated execution time of the hash function begins to play a role, however, this was never apparent in our experiments.

The execution time on MicaZ is appropriately 10% greater than on TelosB. On
Table 6.7: Performance of a reclustering protocol from the BS to the network on different platforms

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution time on TelosB (secs)</th>
<th>Execution time on MicaZ (secs)</th>
<th>RAM on TelosB (bytes)</th>
<th>RAM on MicaZ (bytes)</th>
<th>ROM on TelosB (bytess)</th>
<th>ROM on MicaZ (bytess)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 (8 bit)</td>
<td>30.34</td>
<td>33.37</td>
<td>1230</td>
<td>1309</td>
<td>19802</td>
<td>23600</td>
</tr>
<tr>
<td>SHA-1 (32 bit)</td>
<td>30.28</td>
<td>33.31</td>
<td>1262</td>
<td>1339</td>
<td>20178</td>
<td>25176</td>
</tr>
<tr>
<td>RABIN (32 bit)</td>
<td>30.23</td>
<td>33.27</td>
<td>1100</td>
<td>1433</td>
<td>17972</td>
<td>26190</td>
</tr>
</tbody>
</table>

SHA-1, RAM and ROM, differences are approximately 6.3% and 19-25% greater. On Rabin, these differences grow to 30-46%. Comparing Rabin and SHA-1 directly, Rabin uses less memory, RAM and ROM on TelosB than SHA-1, but uses more RAM and ROM on MicaZ. These differences could be due to different hardware architecture and compilation tool chains.

### 6.5 Summary

In this chapter we presented well-defined authentication protocols to secure the message exchange during the recovery processes. A extensive number of experiments have been conducted to evaluate the performance of hash functions. Our work has been regarded as very efficient in terms of computation. Compared to the work of other researchers, it is also a more secure and reliable process for
implementation in low-resource WSNs.

The contributions from this chapter are:

- An efficient authenticated reprogramming request protocol from the BS to a compromised node.
- An efficient authenticated reprogramming request protocol from an AG to the BS on behalf of a compromised node.
- An efficient authenticated reclustering protocol from the BS to the network.
- A comparison of two lightweight well-known hash functions used in the authentication protocols and implemented with TelosB motes.

The next chapter extends the recovery work we have done and thereafter introduces a mobility support mechanism to be applied in mobile sensor networks.

A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.

—Douglas Adams
Chapter 7

Mobility Support on Recovery

In this chapter, we develop our work in two new directions.

First, we consider the situation of mobile networks. We describe our approach to support recovery of WSNs that are mobile and we consider the impact on the previous recovery protocols in this situation. We show that, with some small changes to the key set-up assumptions, it is easy to adapt our current protocols to this situation.

Secondly, since one of our targets is low-resource recovery, we compare the TelosB platform with the MicaZ platform to see where problems in implementation might occur. Our tests show that MicaZ motes have too little power and insufficient memory to implement attack recovery to the level achieved by TelosB motes.

In our work, WSNs can be divided into three types:

- Stationary, limited-resource networks;

118
• Stationary well-resourced networks; and

• Mobile, well-resourced networks.

Thus far, we have considered only the first two types. However, all of our protocols have been designed taking into account possible deployment in mobile WSNs. In this chapter we specifically analyse the impact on node mobility on the design of the recovery procedures. In addition to this, impact of node mobility on both network reclustering and node reprogramming is also considered.

While stationary WSNs have been studied from a recovery viewpoint [LDBS09], a more difficult problem is recovering WSNs that are mobile. There are numerous scenarios in which mobility can occur. One such scenario is the deployment of a WSN in a coal mine subject to collapses as studied in [MY05]. In this case, motes fell into holes or moved towards a hole and transmission paths were reconnected. Other mobile situations see motes deployed in vehicles or on people or animals.

The main problem distinguishing this situation from the stationary one, is finding and maintaining transmission paths.

In extending our recovery work to large, mobile sensor networks, it is necessary to formalize the mobility support mechanisms. In this chapter, we establish the mobility support needed to ensure the speedy recovery of a mobile WSN and analyze the impact of mobility on the recovery procedures developed in our earlier work. We discuss mobility of member nodes and of AGs in the cluster model, and also identify triggers for reclustering of the network based on loss of connectedness due to mobility. The challenge is to provide mobility support to a mobile WSN which produces the same recovery performance as a static WSN.
We demonstrate the effectiveness of our recovery protocols on low-resourced WSNs with up to 1000 nodes using the NS-2 simulation platform. We show that our method successfully identifies those nodes with higher energy levels and a greater number of neighbours than those of the average node in the network. These nodes are then used as effective local data AGs in the recovery process. In addition, our results indicate that the recovery protocols we have introduced for mobile WSNs are as efficient for a mobile WSN as for a static WSN in producing a connected set of AGs capable of maintaining communication with the BS. Indeed, for networks in which most nodes are mobile at any time, our results clearly indicate that the network is unable to recover from natural breakages of the CDS due to motion. Furthermore, a mobile network under attack has no chance of survival without mobility support.

In this chapter, we focus on, and compare, recovery in both static and mobile WSNs after an attack has been detected. We assume that the network, comprising a fixed BS and mobile sensor nodes, is low resourced but has the capability to detect an attack on the nodes and determine which nodes have been compromised (with high probability). Thus, we may assume that the BS can set target operational levels for the WSN and determine what steps need to be taken to return to these target levels.

Section 7.1 discusses a scenario of the traffic in Shanghai, China in which the mobile WSNs are introduced. Section 7.2 gives a theoretical overview on the recovery protocols in mobile WSNs and introduces the mobility support developed for both AGs and member nodes. Section 7.3 focuses on the impacts introduced by mobility support in sensor networks. Section 7.4 provides the testing results
with mobility support and presents the performance difference between the sta-
tionary and mobile WSNs. Section 7.5 provides the summary of this chapter.

7.1 A Motivating Scenario

Here we introduce a scenario which motivates the decisions and assumptions
made in setting up our mobile WSN. We assume that WSN nodes are distributed
on taxis in a large metropolitan centre (such as Shanghai as used in [LSL+08]).

There is a single BS residing in an office in the centre. All nodes are in range of
the BS, but the BS is not always in range of all nodes. The nodes collect data
about the state of the road (potholes and so forth) and the state of the traffic.
In order for this information to reach the BS, a hierarchical topology is used
by means of AG selection and cluster set-up. A CDS is required to ensure that
all data transmitted by the nodes reaches the BS. The CDS must be regularly
maintained, and, in order to prevent denial of service attacks, reprogramming and
reclustering messages sent between nodes and the BS must be authenticated.

Due to taxis travelling along streets at varying speeds, we represent this topo-
logically as a grid pattern in which waypoints (fixed points) are street corners at
which taxis can turn in one of four directions (at boundary points there may be
fewer than four options). We simplify the topology to a grid pattern in which all
vertical and horizontal travel segments between waypoints have the same length.
However, in application, these lengths may in fact vary and, consequently, we
lose no generality in also assuming a constant travel speed for nodes in our ex-
In this scenario, the BS can support node mobility, and consequently also CDS availability, by controlling the location of the taxis based on information sent to it. It is, of course, in the best interests of the taxi service to ensure that taxis are distributed across the domain in order to have the best response to customer needs. In our experimental set-up, we therefore implement the random waypoint mobility model [SBM05], which has been shown to spread nodes across a given convex domain better than other mobility models.

A final point to be made here is that mobility of the nodes is provided by the taxis themselves. There are many WSN mobility scenarios in which the nodes are moved by other objects (the nodes reside on animals or people for example, or are moved in a mine collapse by falling debris). Thus in measuring node energy, we do not take into account any energy loss due to motion. The only energy loss we consider is that due to data transmission.

### 7.2 Theoretical Overview

#### 7.2.1 Assumptions

We follow the same approach as in [LDB10]. Our set-up comprises a single, secure and trustworthy BS along with nodes capable of operating as either member nodes or AGs. The WSN is clustered into groups of nodes each monitored by an AG. An AG node stores identifying information about those nodes in its cluster. All messages transmitted in the WSN identify the message source. Member nodes
gather data which is then sent to AGs. AGs both gather data and aggregate collected data before sending it to the BS. The BS analyses and stores data and keeps logs of this process. We program the same reprogramming, reclustering and hash function code into each node at set-up and so it does not have to be transmitted during network operation.

We make four assumptions about our environment. Each is listed below and the impact of each is discussed as needed throughout the chapter.

- We assume the existence of a globally unique ID for each sensor node.

  The BS keeps track of all IDs and uses them to identify nodes which need to be reprogrammed. Maintenance of a unique ID is necessary for general WSN business and has no additional impact on resources required to deploy the network for its essential business, thus, this assumption is natural.

- We assume that the BS shares with each WSN node a common secret not known by any other node, which is allocated when the WSN is set up.

  These secrets are used to implement authentication when messages are sent. Unauthenticated reclustering and reprogramming messages could be used in a denial of service attack.

- We assume that secrets are stored in a tamper-resistant section of the node and that calculations involving it are executed in this tamper-proof section.

  The use of a tamper-proof or secure area for storing secrets and executing computations with them is a standard solution to key management. In addition to keys, we pre-store all necessary code in this section of the mote.
However, it is difficult to design and implement such a secure area on a small device such as a mote.

- We assume that, at any time, some node is within range of the BS; however, many nodes may not be in range of the BS.

The sensors are permitted to move around a convex area which we refer to as the sensor domain.

### 7.2.2 Comparison on Recovery with Other Researchers

The requirements of the clustering algorithm to achieve complete connectedness across the static network were based on the following target conditions identified earlier in this thesis. Conditions T1 and T2 below were assumed for the static WSN recovery situation described in [LDBS09]. Condition T3 was added for the mobile TinyOS motes in [LBD09].

- T1. Each node be no more than two hops away from an AG,

- T2. The AG set be connected, enabling transmission of data along the AGs to the BS, and

- T3. At least one AG be one hop away from the BS.

In Table 7.1, we summarize the major points of difference between the algorithms developed in other papers and the one we present in this work. The features appearing in the table were used in [16] for comparison. The properties studied are the following:
• Variable AG period: if nodes sometimes act as AGs and sometimes not.

• Conditions of AG election: the factors used in determining when a node is eligible to be an AG.

• Capability: if all nodes are supplied with the same capabilities.

• Messaging complexity: the number of messages transmitted in order to restore connectivity to the WSN.

• Non-messaging costs of connectivity restoration: costs to the network other than messaging.

• Synchronization: If transmission needs to be synchronized for the restoration process.

• Global information: whether a node requires information about the entire network in order for the restoration protocol to succeed.

Protocols DARA, RIM and PCR use the mobility of the network as a solution. Thus, the non-messaging cost is that mobility which is not part of the designated mandate of the network, but which detracts from that mandate. In all three of these cases, 1-hop or 1-and 2-hop neighbour information suffices to support the protocol; this information includes the node ID and its location. In all three protocols, information about cut-vertices in the graph of the network is required (see Table 7.3). For example, in DARA, 2-hop neighbour information is used to generate a breadth-first search spanning the network in order to locate cut-vertices.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAC (1999) [Bas99]</td>
<td>Yes</td>
<td>Allocated Weight (Number)</td>
<td>Yes</td>
<td>O(d)</td>
<td>None</td>
<td>No</td>
<td>No (1-hop Neighbour Information)</td>
</tr>
<tr>
<td>Dai and Wu (2004) [DW04]</td>
<td>Yes</td>
<td>Allocated Weight (Number)</td>
<td>Yes</td>
<td>O(N)</td>
<td>None</td>
<td>No</td>
<td>Sometimes (3-hop or More Neighbour Information)</td>
</tr>
<tr>
<td>DARA (2009) [AAA09]</td>
<td>No</td>
<td>Pre-designed</td>
<td>Yes</td>
<td>O(N)</td>
<td>Distance Travelled by Nodes</td>
<td>Partial (Cascaded Relocation)</td>
<td>Yes (1- and 2-hop Neighbour Information)</td>
</tr>
<tr>
<td>RIM (2010) [MYA10]</td>
<td>No</td>
<td>Pre-designed</td>
<td>Yes</td>
<td>O(N)</td>
<td>Distance Travelled by Nodes</td>
<td>Partial (Cascaded Relocation)</td>
<td>Yes (1-hop Neighbour Information)</td>
</tr>
<tr>
<td>PCR (2011) [IYSH10]</td>
<td>No</td>
<td>Pre-designed</td>
<td>Yes</td>
<td>O(N)</td>
<td>Distance Travelled by Nodes</td>
<td>Partial (Cascaded Relocation)</td>
<td>Yes (1-hop Neighbour Information)</td>
</tr>
<tr>
<td>Our Solution</td>
<td>Yes</td>
<td>Energy and Number of 1-hop Neighbours</td>
<td>Yes</td>
<td>O(N)</td>
<td>None</td>
<td>No</td>
<td>No (1- and 2-hop Neighbour Information)</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of solutions to restoring connectivity in mobile WSNs

('d' is the maximum distance between two nodes; 'N' is the number of nodes in the network.)
In the other three protocols, the restoration costs are all associated with messaging, and, except in [Bas99], that cost is the order of the number of nodes present in the network. In the case of [Bas99], it is a function of the distance across the domain covered by the network. The authors of [DW04] need at least 3-hop neighbour information, and in some cases require global information, in order to ensure that restoration occurs, while [Bas99] and the current paper only require localized information about neighbours.

The DMAC protocol of Basagni [Bas99] satisfies T1 of our conditions above, but does not satisfy T2 or T3. We argue that a WSN based on a protocol without these conditions is susceptible to failure more quickly than is a WSN using a protocol in which all three conditions hold. This is substantiated by Figures 7.17 and Figures 7.18 of Section 7.5.2, which indicate that in the absence of T2 and T3, the number of nodes able to join a cluster diminishes rapidly as the speed of the nodes increases, and also by Table 3 of the same section, which shows that without T2 and T3, a CDS breaks more quickly.

Our proposed protocol has no associated costs other than that of messaging, needs only 1- and 2-hop neighbour information at each node and does not require synchronization to restore connectivity. We show that, in addition to T2 and T3, a weakened version of T1 can still hold and provide support to the mobile case that compares with support to a stationary network.
7.2.3 Mobility Support for the Re-Clustering Algorithm

The clustering method applied in our WSN is the modified EMPR algorithm proposed by the research team and detailed in the paper [LDBS09]. This algorithm efficiently partitions a WSN with a flat topology into a hierarchical network consisting of a set of clusters that are grouped with an AG. It can be shown that the AGs chosen are connected among each other and thus form a CDS of network nodes. The proposed method determines in a distributed, localized fashion a set of AGs by firstly collecting 2-hop neighbourhood information in each node v of the WSN, then selecting the AGs by iteratively searching for the best-suited set of MPR, and finally associating member nodes to an appropriate AG.

The connectedness of the WSN to the BS in order to ensure regular operation of the data sensing tasks is assured by the connectedness of all nodes through the AGs. In addition, we aim for energy-efficiency by choosing AGs with high battery energy levels.

We consider the impact of mobility after the clustering has been completed with the target condition achieved. We firstly consider AG mobility.

In order to re-cluster the WSN, the BS retrieves the secrets $S_n$ of each node n and computes the hash function $h(S_n \otimes M \otimes R) = c_n$, where M is the message to re-cluster and R is a time stamp or random nonce which is used to avoid re-use of the message in a denial-of-service attack. The BS then transmits $c_n$ with M and R to the corresponding node n through the CDS formed by the AGs. Each node n computes $h(S_n \otimes M \otimes R)$ in its tamper-proof area to confirm that the message is authentic. If there is a match AND if no such message with time R has been used
Figure 7.1: The BS sends an authenticated reclustering command
in a reclustering request earlier, n accepts this as a valid reclustering message.

In the process, the BS sends the messages to the corresponding nodes one by one, each node needs to verify the hash and reply to this message. The BS then makes sure it receives all the messages by checking the node ID and counting the number of received message. Once all nodes have received and verified such a message, reclustering commences. In order to select the AGs and thus determine the CDS, each node starts sending 1-hop hello messages to their 1-hop neighbours. Any
node receiving a 1-hop hello message from a neighbour then stores the sender ID in its 1-hop neighbour table. Figure 7.1 presents how the BS sends authenticated reclustering command in flow chart.

The process will be repeated three times to improve the number of the messages the nodes receive from their neighbours so that the neighbour tables can be fully established. Once the neighbour tables are established, nodes start exchanging their 1-hop neighbour lists with 1-hop neighbours in order to establish a 2-hop neighbour list. Based on the neighbour information and energy level, each node can now compute its metric and hence its coverage set information and exchange this information with neighbouring nodes. A node becomes an AG if the particular conditions are met. AGs then send the message ‘I am AG’ to their neighbours. Nodes which receive such a message become member nodes. This is described in Figure 7.2.

7.2.4 Constructing CDS

We adopt Wu’s method [Wu03] of determining a CDS and, in order to support resiliency requirement R2, employ both the battery energy level and the number of neighbours of each node in deciding which should be AGs. Values associated with each of these features are used to construct a metric M(v) used below. This adaptation of Wu’s algorithm was presented in [LBD09] and is once again presented below.

A 1-hop neighbour of a node v is a node within range of v. A 2-hop neighbour of v is a node within range of a 1-hop neighbour of v.
After collecting the neighbourhood information, each node \( v \) selects a set of MPR that can be viewed as candidate AGs. This set comprises a small subset of nodes \( C(v) \) from the 1-hop neighbour set \( N1(v) \) of node \( v \) that fully covers the 2-hop neighbour set \( N2(v) \) of node \( v \). A set \( S \) fully covers \( N2(v) \) if every node of \( N2(v) \) is in range of some node of \( S \). \( C(v) \) is thus also called the coverage set of node \( v \), and it can be shown that the \( C(v) \cup v \) forms a CDS for \( N2(v) \).

The coverage set \( C(v) \) is obtained by executing the modified EMPR algorithm [LBD09] which takes into account a known metric value \( M(v) \) associated with each node \( v \). \( M(v) \) is a function of the energy level of the node and of the number of 1-hop and 2-hop neighbours of the node. The higher each of these values is, the greater the chance of the node being chosen as an AG. In our experiments, each node is assigned the same initial energy level of 2500 units, as in practice, we would assume that each deployed mote would start out at maximum energy.

We again adopt EMPR [Wu03] as the basis of our reclustering algorithm and employ both the battery energy level and the number of neighbours of each node in deciding which should be aggregators; values associated with each of these features are used to construct the metric \( M(v) \) introduced below. Because our mobile WSN requires 2-hop neighbour information, we must adapt the metric used in the static case to this new situation. We also move from a real interval to a set of integers to facilitate computations in NS-2, our experimental platform.

We therefore define \( M(v) \) for a node \( v \) to be a function of the energy level of the node and of the number of 1-hop and 2-hop neighbours of the node. The higher each of these values is, the greater the chance of the node being chosen as an AG. In our simulation experiments, each node is assigned the same initial energy level
of 2500 units, as in practice, we would assume that each deployed node would start out at maximum energy. In addition, as indicated in Section 1.3, we include the BS as a node in the construction of the CDS in order to ensure that M3 holds. The following Equation 7.1 is then used to calculate \( M(v) \) at any fixed time \( t \):

\[
M(v) = e(v)d_{\text{max}} + d(v)e_{\text{max}} + \varepsilon(v)
\]  

(7.1)

where

\[
\varepsilon(v) = 2500 - S_t(v)
\]  

(7.2)

and \( S_t(v) \) is the energy expended by \( v \) between initialization of the network and time \( t \) at which the metric is being calculated. Thus, \( \varepsilon(v) \) keeps track of the remaining energy at \( v \) over time. The radio transceiver of a sensor node is always in one of the three states; transmit, receive or idle. To simplify our measurements, nodes in the idle state consume zero energy, nodes in the transmit state consume 1 energy unit and nodes in the receive state consume 2.5 energy units [YHE02]. Thus, setting our initial energy level at 2500 units ensures that we can run reclustering a number of times before the energy of a node is expended.

After the MPR have been selected, each node broadcasts its coverage set \( C(v) \) to its 1-hop neighbours at a random time instant in the next time interval. At this point, A node \( v \) decides to act as a AG if

- it has a larger metric \( M(v) \) than all its 1-hop neighbours and has at least
two non-1-hop neighbours, or

- it is in the coverage set formed by its neighbour with the largest metric $M$.

The set of all such AGs forms a CDS \cite{Wu03}, \cite{WLD04} for the WSN. It is important for the recovery protocols that at least one member of the CDS be able to communicate directly with the BS; that is, the BS is within its range. If this is not the case, one additional mote which can reach the BS is added to the CDS. This additional mote is chosen to have maximum metric from among all motes which can reach the BS. The CDS properties are still held by this new set.

### 7.2.5 Mobility of AGs

AG mobility will be a problem for the clustered network structure only if it leads to a breaking of one or more of the target conditions. Conditions T2 and T3 are related to AGs and can be impacted by AG mobility. Condition T2 can be violated as a result of AG mobility resulting in the breaking of the CDS formed by the AGs. If the CDS is broken, then data from a cluster will not be able to reach the BS. Hence, it is imperative that any disconnection of the CDS is detected as early as possible.

For the purpose of detection of disconnection in the CDS, a simple solution is to implement a 'keep alive' mechanism that requires all AGs to periodically exchange a small beacon with the neighbouring AGs. The time period can be tuned according to the level of relative mobility expected between nodes. Hence if relative mobility is high then the time period is small and vice versa. Using the keep
Figure 7.3: An AG moves out of range of its neighbour AGs

alive mechanism, when one of the AGs leaves the CDS, its neighbours will be able to detect the disconnection of the CDS and can report this to the BS through the remaining AGs that will still have a connected path to the BS. Figure 7.3 displays this situation; the AG in the dotted circle moves out of range of its neighbours disconnecting itself and the AG feeding data into it from the CDS. The BS will then be required to initiate networkwide reclustering procedures to establish a new a set of AGs that satisfy CDS target requirements.

However, it is possible that conditions T2 and T3 are violated concurrently. This can occur if the CDS of AGs chosen is at most one AG within one hop of the BS (note that all nodes in the WSN are within the broadcast range of the BS). As a result, even when the neighbours of the AG that has caused a disconnection of
Figure 7.4: No AG is in range of the BS

the CDS are able to detect the violation of T2, the concurrent violation of T3 prevents them from reporting this to the BS to initiate reclustering procedures as in Figure 7.4. Figure 7.4 shows that only the AG which was within range of the BS has now moved out of range (the AG with the dotted circle) and, as a consequence, the CDS can no longer function.

In Figure 7.4, the keep alive mechanism allows the neighbours of the AG to detect the disconnection. However, since the neighbours are not within reach of the BS they are unable to report the disconnection to the BS. Hence, it is proposed that in mobile WSNs, target condition T3 be modified as below:

- M3. At least two AGs be one hop away from the BS.

This will ensure that a successful report of a disconnection of the CDS can be
communicated to the BS and reclustering procedures can be initiated as shown in Figure 7.5.

In order to ensure that a broken CDS is detected, each AG sends ‘alive’ beacon messages periodically to the neighbouring AGs. An AG then compares the messages received from other AGs within its neighbourhood and determines if any AG is missing. A missing AG is reported to the BS and once an AG has been reported missing three times to the BS, the BS initiates reclustering.

7.2.6 Mobility of Member Nodes

We now consider mobility of the member nodes. A node that is within the range of one cluster may as a result of its mobility, move out of the range of
the cluster and move into the range of one or more clusters. This change in the membership of clusters needs to be detected and can be done using simple keep alive mechanisms as outlined earlier. To account for mobility it is required that all nodes in the network send out periodic beacons to allow neighbours to detect their connectedness within a cluster. The periodic beaconing can also serve the purpose of allowing a member node that is moving out of the range of a cluster to learn about other clusters that it can join. To ensure that the target conditions are maintained as much as possible, it is proposed that nodes include in their beacons hop count information with respect to distance from the AG. This will allow a member node receiving beacons from multiple clusters to join the cluster that gives it the least number of hops to an AG. It is easily observed that node mobility will make the maintenance of target condition T1 almost impossible to achieve during network operation. Hence, it is proposed that in mobile WSNs, target condition T1 be relaxed during network operation to allow member nodes to join clusters even if AGs are more than 2 hops away.

When a node joins a cluster, the one hop neighbour table of its parent node (i.e., a node that it is directly connected to) is updated and the information of the new member node is propagated to the AG. It is sufficient to do this and not update all neighbour tables in the entire cluster as it is only the AG that requires accurate knowledge of the member nodes during network operation. When reclustering is required, the neighbour tables can be updated as part of the neighbour discovery phase of the reclustering algorithm for each node in the network. Hence mobility does not impact on the reclustering process. However, we do assume that the mobility of the nodes does not result in a change of the neighbour tables during
Based on the above discussion the target conditions for a mobile WSN are:

- M1. Each node be on a hopping path to at least one AG,
- M2. The AG set be connected enabling transmission of data along the AGs to the BS, and
- M3. At least two AGs be one hop away from the BS.

As nodes are mobile, these target conditions will be violated during network operation. However, periodic reclustering can be done to ensure that target conditions are maintained. The mobility of nodes generates this additional overhead in terms of maintaining the target structure.
If all the nodes are involved in determining whether an AG has moved out of the CDS, the additional messaging will significantly increase network collisions and be detrimental to WSN function. Therefore, we limit this determination to AGs. Figure 7.3 describes the method.

7.3 Mobility Impact

7.3.1 Clustering and Re-Clustering

A major part of network resiliency deals with recovery after accidents or attacks. In [LDB10], we introduced protocols to reprogram compromised nodes and to re-cluster when routes between nodes and to the BS are broken or may become broken.

By ensuring that the BS is always in range of at least one member of the CDS, network operation is maintained while the recovery procedures of reclustering and node reprogramming are carried out. Our experiments show that the proposed approach is robust enough to minimise the impact of both AG and member node mobility on the recovery process.

In using the NS-2 platform, we are unable to capture node data relevant to the reprogramming and authentication protocols, whose times were measured on hardware in [LDBS09]. We therefore focus on the reclustering component of the recovery operation. The authenticated reclustering protocol used is in [LBD09]. In each case, a node with ID n contains secret Sn (shared with the BS), R
represents a random value but in practice is the local time obtained from the LocalTime.get() command in TinyOS, and M represents a message to reprogram or recluster or a request that another node be reprogrammed. All messages transmitted include node ID of both sender and receiver, including that of the BS. A hash function check achieves authentication of a message. The two hash functions SHA-1 [oST02] and Rabin [Rab79] were chosen and were both used in each protocol to compare their performance.

7.3.2 Impact of Mobility on Re-Clustering, Reprogramming and Authentication Procedures

The reprogramming and authentication procedures are not affected by node mobility since both reprogramming and authentication are accomplished directly between the BS and the AG and member nodes. This requires that the broadcast range of the BS be such that it can cover the entire network area. The required broadcast range can be derived from the beacon ranges of the individual nodes in the network. Since any node sending a beacon needs at least one other node to be able to receive it, the worst case scenario is when all nodes in the WSN are strung out in one line, each just a beacon range apart. Thus, if \( n \) is the number of nodes in the WSN (excluding the BS), and \( r \) is the beacon range in metres, \( r \times (n-1) \) metres is the largest distance that the nodes can be distributed across while maintaining the ability to execute network protocols. This implies that the BS range \( R \) must also cover this distance, and, allowing for some overlap or malfunction, we enlarge the requirement to \( R \geq r \times n \).
We have thus defined a network area within which nodes in the WSN are assumed to move relative to the range of the BS. This is a reasonable assumption, as a mobile network cannot move in an unrestrained fashion while still maintaining protocol operation.

As long as all nodes are within the broadcast range of the BS and are connected to an AG through one or more member nodes, reprogramming and authentication can be carried out using similar techniques as in a stationary WSN. This condition could be further relaxed to require that only the CDS set be within the range of the BS and routing all BS to member nodes communication through the CDS set of AGs. However, since the CDS set changes regularly, this would have a major impact on the choice of new AGs in reclustering mode, and is not recommended.

The protocol descriptions for the mobile WSN are presented in the next section. We note that the confirmation step in these procedures required the node to be able to send a confirmation message to the BS. Hence, there is the requirement that a reprogrammed node be able to send a message to the BS which it can through its AG. However, a compromised AG can drop this confirmation. Hence, in terms of recovery, accurate detection is imperative to ensure that if an AG is compromised, the AG is reprogrammed first, followed by a reclustering of the WSN which is then followed by reprogramming of any compromised nodes.

The reclustering algorithm can be affected by the mobility of nodes in that the coverage set calculated at the end of the neighbour discovery phase can be inaccurate as a result of node mobility during the reclustering process. It will be infeasible to ensure that the coverage sets are accurate when nodes exhibit individual mobility characteristics. However, as long as the relative mobility between
nodes is such that it does not cause changes in the coverage set, the calculation of
the CDS of AGs is expected to be accurate. As noted earlier, since nodes exhibit
group mobility characteristics, we expect the relative mobility between nodes to
be such that any changes in the coverage set do not have a great impact on the
calculation of the CDS of AGs.

It is also possible that M3 is violated only two AGs connected to the BS are
simultaneously disconnected from the BS. We note that the scheme is robust
enough to handle this situation with the added requirement that the BS monitors
the 'keep alive' messages from the AGs. Since the keep alive interval is relatively
small, a BS will be able to detect a disconnected CDS almost immediately and
then re-cluster.

To summarize the points of difference between the static and mobile cases, we
list the key features here:

**Static**

- 1- and 2-hop neighbour information is gathered only once after a re-cluster.
- A compromised AG generates reclustering.

**Mobile**

- 1-and 2-hop neighbour information is gathered regularly between re-clusters.
- Nodes keep track of hopping path distances from AGs (when available).
- The BS send keepalive messages regularly to its neighbour AGs.
• Loss of contact between the BS and any AG node generates reclustering.

• **Theorem 1**: under the proposed mobility support protocol, M1, M2 and M3 hold.

• **Proof**: When reclustering takes place, each node updates its 1- and 2-hop neighbour tables. Wu [Wu03] proved that a set of nodes satisfying two conditions listed in Section 3.2.2.3 forms a CDS; that is, the set of AGs constructed forms a communication path of 1-hop neighbours and every node is either an AG or a 1-hop neighbour of an AG. In this case, both M1 and M2 are satisfied. Because we have allocated the BS the highest metric in connection with Equation 7.1, the BS is part of the CDS and so one hop away from some AG node. Thus, M3 is satisfied.

While the network moves, paths between nodes may change. When it gathers new table information, a node may recognize that it no longer has an AG in its 1- or 2-hop neighbour table. It continues to transmit however, and its messages are carried along any existing path to an AG and hence to the BS. The BS fails to receive messages from a node if that node is a member node that can no longer connect to the CDS or if that node is an AG which has become disconnected from the CDS. In either of these cases, the BS recognizes the situation and initiates reclustering in order to establish a new CDS. Hence, M1, M2 and M3 hold.
7.3.3 The Impact of Mobility on Recovery Procedures

In this section, we present an analysis on the impact of mobility on the recovery procedures developed in the first few chapters. We assume a single, secure BS along with nodes capable of operating as either member nodes or AGs. The WSN is clustered into groups of nodes each monitored by an AG. The AG stores identifying information about those nodes in its cluster. All messages transmitted in the WSN identify the message source. Member nodes gather data which is then sent to AGs. AGs both gather data and aggregate collected data before sending it to the BS. The BS analyses and stores data and keeps logs of this process.

Thus, the goals of the network are to:

- Collect data of several pre-specified types from member nodes and AGs.
- Collate this data via the AGs which then transmit to the BS.
- The BS analyses and stores data, logs network functions and activities and transmits this information regularly to a location beyond the network.

In our architecture, the following components are used to assist in achieving recovery:

- Several algorithms for logical clustering of the network, each playing an important role under certain specified conditions.
- All nodes able to take on the role of member nodes or AGs depending on the clustering algorithm applied.
• A decision-making process in the network which correctly identifies those nodes which have been compromised.

• A decision-making process in the network which determines what action to take with compromised nodes.

• A reprogramming mechanism enabling the network to reprogram compromised nodes.

### 7.4 Mobile WSN Protocol Description

#### 7.4.1 Mobile Reprogramming Protocol from the BS to a Compromised Node

- BS XORs the secret $S_n$, the reprogramming message $M$ and the local time $R$ to obtain $S_n \oplus M \oplus R = m$.

- BS computes $h(m) = c$.

- BS transmits $c$, $M$ and $R$ through the CDS of the AGs which are then received by $n$.

- Node $n$ re-computes $h(S_n \oplus M \oplus R)$ in its tamper-proof section, and checks that it is $c$.

- If the check is 'true' AND time $R$ has not been used in an earlier reprogramming request, the node initiates reprogramming procedures.
• The node confirms that it has reprogrammed to the BS by computing \( h(S_n \otimes M) = c' \) and sending \( c' \) to the BS through the CDS formed by the AGs.

7.4.2 Mobile Reprogramming Protocol from an AG to the BS on Behalf of a Compromised Node

• The AG retrieves the ID \( n \) of the node to be reprogrammed.
• The AG computes \( h(n \otimes SA \otimes M \otimes R) = c \) in its tamper-proof section.
• The AG transmits \( c, n, M \) and \( R \) to the BS through the CDS of AGs.
• The BS retrieves \( SA \), re-computes the hash and checks if it is \( c \).
• If the check is ’true’ AND the time \( R \) has not been used in a reprogramming request earlier, BS initiates Protocol 7.4.1.

7.4.3 Mobile Re-Clustering Protocol from the BS to the Network

• The BS retrieves the secrets \( S_1, Sc \) of each node and for each node ID \( n \), and computes \( h(S_n \otimes M \otimes R) = cn \).
• The BS then transmits \( cn \) with \( M \) and \( R \) to the corresponding node \( n \) through the CDS of AGs.
• Each node \( n \) computes \( h(S_n \otimes M \otimes R) \) in its tamper-proof area to confirm that the message is authentic.
If there is a match AND if no such message with time R has been used in an earlier reclustering request, n accepts this as a valid reclustering message.

Once all nodes have received and verified such a message, reclustering commences.

The proposed approach to handle mobility and its impact on the recovery procedure is simple and with the introduction of the keep alive mechanism, can ensure that the target goals of the recovery process are achieved. We recall that the target goals are:

- R1. Restore the network functions to a target level established by the BS.
- R2. Maintain the network for as long as possible.

By ensuring that the CDS is connected as far as possible, network operation is maintained while the recovery procedures of reclustering and node reprogramming are carried out. The proposed approach is robust enough to minimise the impact of both AG and member node mobility on the recovery process.

7.5 Experiments and Evaluation

In this section, we describe our experimental results. In previous work, our recovery algorithms were implemented on TelosB motes and we summarize that implementation briefly. We then describe in detail the simulation implementation.
7.5.1 TelosB Experiments

The TelosB testbed involved a sensor network of up to 20 nodes deployed within one building. Initial tests showed that the range of these devices varied from 25m to 40m in indoor and outdoor environments respectively. The network was deployed with a flat structure including one BS node and multiple sensors connected through multihop paths to the BS. Once clustering was completed, the network was organised into a hierarchical architecture with a BS, AGs and member nodes.

We tested our recovery protocols on TinyOS Version 2.1 using the Crossbow TelosB research mote as our experimental testbed platform as described in [LDBS09], [LBD09] and [LDB10]. More than 2000 lines of code were written to implement the reclustering. The program was installed on the TelosB motes and the reclustering protocol was triggered by the BS sending a ’start’ command. This solved the time synchronization issue and ensured the nodes all started at the same time.

We tested using 10, 15, 20 stationary motes and then the same number of mobile motes, in order to demonstrate the time differences between the two cases. In our experiments, up to 20 sensor nodes were located on a regular grid of 10 (5x2), 15 (5x3), and 20 (5x4) uniformly at 2 metres apart in both dimensional directions in a rectangular area of 10 metres by 8 metres. Each node had a radio power range of 2.5 meters and so, in the initial set-up, was within range of several nodes in the grid. In the mobile case, we moved the sensors using the waypoint model discussed in the previous section, with at most 10% of the nodes in motion at any time, with each node having an opportunity to move. Since there were no complex routing requirements, broadcasting was used to transmit the messages and
no routing table was maintained during the reclustering process. Two neighbour tables were created to store 1-hop and 2-hop neighbour list.

In the mobile case, motes were mounted on top of remote-controlled toy car models. As the code remained constant in both situations, there was no difference in RAM or ROM size between the stationary and mobile scenarios. Different motes were chosen to start moving in each experiment. When in motion, nodes moved towards waypoints at an average velocity of 0.5 metres per sec. When a waypoint was reached, the mote stopped for a short interval of between 0.5 and 3 secs in order to determine its next direction before moving on to the next randomly chosen waypoint. Mote direction was based on the grid pattern. Motes moved towards nearest waypoints up to 2 metres away. This allowed up to 4 directions of movement for a given mote and randomness was introduced by blind choice of one of 4 counters indicating the next move. If the move chosen was not possible (as for a mote on a side of the grid for instance), it was discarded and a second choice made.

For reclustering, we applied the random waypoint model and ran twenty tests. Several nodes were in motion at any time when authentication was used but all nodes were left stationary when the reclustering process started in order to ensure a correct neighbour table. The results can be found in [LDB10].

For Protocols 7.4.1 and 7.4.2, we used a rectangular area of 10 metres by 8 metres with, as an initial set-up, 20 sensor nodes distributed in a grid pattern inside it uniformly at 2 metres apart in both dimensional directions. We moved the sensors using the waypoint model discussed previously. Waypoints were established in a uniform grid pattern along and within the boundary of the area at 2 metre
intervals in both dimensional directions.

Each node had a radio power range of 2.5 meters and so, in the initial set-up, was within range of several nodes in the grid. Nodes were moved by sitting them on top of remote-controlled toy car models. Due to frequency interference, we were able to move at most, two cars at any given time. When in motion, nodes moved towards waypoints at an average velocity of 0.5 metres per sec.

Testing this set-up for each protocol twenty times resulted in an average CDS size of 4 AGs. The smallest CDS achieved was 3; the largest was 6.

For Protocol 7.4.3, we have three different set-ups for using a set of 10, 15 and 20
CHAPTER 7. MOBILITY SUPPORT ON RECOVERY

Figure 7.9: 15-node grid topology for the experiments of Protocol 7.4.3

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Execution Time for Authentication (secs)</th>
<th>Execution Time for Reclustering (secs)</th>
<th>Total Program Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stationary WSNs</td>
<td>Mobile WSNs</td>
<td>Stationary WSNs</td>
</tr>
<tr>
<td>10 Nodes</td>
<td>58.13</td>
<td>63.23</td>
<td>87.67</td>
</tr>
<tr>
<td>15 Nodes</td>
<td>79.20</td>
<td>85.37</td>
<td>87.79</td>
</tr>
<tr>
<td>20 Nodes</td>
<td>93.46</td>
<td>99.58</td>
<td>87.21</td>
</tr>
</tbody>
</table>

Table 7.2: Comparative performance of Rabin authentication and reclustering from BS to network
TelosB nodes. As described for Protocols 7.4.1 and 7.4.2, we used a rectangular area of 10 metres by 4 metres for 10 nodes, and 10 metres for 15 node networks respectively. 18 and 24 waypoints were marked as shown in Figure 7.8 and Figure 7.9.

In each run of the experiments, we started with different motes on the grid as our mobile nodes. The mobile nodes moved over all the waypoints at least one time on the grid. When the waypoint was reached, the node stopped and stayed for a randomly chosen time between 0 seconds and 3 seconds. Then it randomly chose another waypoint and began to move again.

We expected additional resource costs in the mobile case compared to the static case. Indeed, the additional percentage performance for the mobile case over the stationary case was 8.77 with 10 nodes, 7.79 with 15 nodes and 6.55 with 20 nodes as shown in Table 7.2. For 10 nodes an average, CDS had a size of 5 AGs; the smallest CDS achieved was 4 and the largest was 5. For 15 nodes an average CDS was the size of 6 AGs; the smallest CDS achieved was 5 and the largest was 7. For 20 nodes an average CDS was the size of 6 AGs; the smallest CDS achieved was 5 and the largest was 8.

All experiments related to mobility produced results slightly slower than in the stationary case. The cost of reprogramming mobile motes was no more than an additional 25% additional to that of reprogramming static motes. The percentage increase relative to reclustering was at most 9% [LDB10]. All in all, the results in the mobile case remain acceptably close to those in the stationary situation.
7.5.2 Simulation Experiments

To extend the physical results, we ran our modified EMPR Algorithm on the NS-2 simulation platform [NS-], the most popular network simulator, in order to test the performance of the reclustering methods with respect to fast recovery from security attacks, while at the same time maintaining reasonable battery energy levels.

We run every experiment for at least 30 times to ensure the reliability and consistency of the design of our protocols.

In this algorithm, the choice of AG is based on the value of $M(v)$, which is a function of the energy level of the node and of the number of 1-hop neighbours of the node. We assessed the performance of our method by measuring and comparing the following performance metrics both in physical and simulated environments:

- Average battery energy level of all WSN AGs.
- Average battery energy level of all WSN member nodes.

If our algorithm was working well, we expected that the AGs chosen would have a higher value of $M(v)$ than those of their member nodes.

In addition, we recorded the number of AGs elected by the algorithm. The expected number of AGs for a given network varies with the number of nodes in the network as shown through simulations in [Wu03], and we expected that this variation was represented by linear growth.
In our simulation tests, waypoints were established in a uniform grid pattern along and within the boundary of the area at 30 metre intervals in both dimensional directions in general, waypoints might be distributed in any uniform configuration across the sensor domain. The random waypoint mobility model was generated by executing the 'setdest' (set destination) command in the NS-2 environment. The mobile nodes were set to move at varied speeds in each experimental scenario. Nodes moved according to the random waypoint model with a speed of 2-3 m/s, 3-5 m/s or 5-10 m/s and a stop period of 2 sec or 5 sec. The destination and direction of nodes were chosen randomly and independently of other nodes. The nodes started moving after AGs were elected and stopped when the simulation was finished. We deployed the nodes in a grid formation within a 700 x 700 metre area with the mobile network size ranging from 100 nodes to 1000 nodes (deployed in 10 nodes x 10 nodes, 10 nodes x 20 nodes, 15 nodes x 20 nodes, 20 nodes x 20 nodes, 20 nodes x 25 nodes, 24 nodes x 25 nodes, 25 nodes x 28 nodes, 20 nodes x 40 nodes, 30 nodes x 30 nodes and 25 nodes x 40 nodes respectively). Each node had a radio power range of 40 meters and so, in the initial set-up, was within range of several nodes in the grid. In each deployment, the simulation experiment started when the nodes ran the protocols and ended within 30 minutes of the simulation time. Reclustering was triggered when the CDS was broken and the BS initiated the reclustering after it received confirmation that the CDS was broken.

[CM09], compared several routing protocols for mobile networks using parameters such as speed of node movement and length of stopping times. They indicated that the standard flooding protocol does particularly bad in terms of packet loss.
as the speed of the nodes increase. This would have an adverse impact on the ability of the WSN to elect a CDS. We were therefore interested in seeing how CDS selection was affected by speed in our protocol as well as by the percentage of nodes which were mobile at any time.

Figure 7.10: Number of AGs chosen: The static case and the mobile cases with 20% of the nodes mobile with a 2 sec pause at waypoints

Figure 7.10 and Figure 7.11 show the average number of AGs selected in a sensor network with 20% of nodes mobile and 2 secs pause time at waypoints and 5 secs pause time at waypoints respectively. Network size varies from 100 to 1000 nodes and speeds vary through 0m/sec (the static case), 2-3 m/sec, 3-5 m/sec and 5-10m/sec. Each case was tested 20 times and the average taken. While in both graphs the static case produces marginally more AGs than any of the mobile cases, there is no significant difference between the results from the different speeds or the different stopping times at waypoints.

Figure 7.12 and Figure 7.13 show the average number of AGs selected in a sensor
Figure 7.11: Number of AGs chosen: The static case and the mobile cases with 20% of the nodes mobile with a 5 sec pause at waypoints

network with 40% of nodes mobile and 2 secs pause time at waypoints and 5 secs pause time at waypoints respectively. Network size varies from 100 to 1000 nodes and speeds vary through 0m/sec (the static case), 2-3 m/sec, 3-5 m/sec and 5-10m/sec. Each case was tested 20 times and the average taken. Again, while in both graphs the static case produces marginally more AGs than any of the mobile cases, there is no significant difference between the results from the different speeds or the different stopping times at waypoints. These results compare well with the conclusions of [CM09].

Finally, to test the protocols when a majority of nodes are mobile, we tested the case with 80% of the network in motion. Figure 7.14 and Figure 7.15 show the average number of AGs selected in a sensor network with 80% of nodes mobile and 2 secs pause time at waypoints and 5 secs pause time at waypoints respectively. Network size varies from 100 to 1000 nodes and speeds vary through 0m/sec (the static case), 2-3 m/sec, 3-5 m/sec and 5-10m/sec. Each case was
CHAPTER 7. MOBILITY SUPPORT ON RECOVERY

Figure 7.12: Number of AGs chosen: The static case and the mobile cases with 40% of the nodes mobile with a 2 sec pause at waypoints.

Figure 7.13: Number of AGs chosen: The static case and the mobile cases with 40% of the nodes mobile with a 5 sec pause at waypoints.

tested 20 times and the average taken. Again, while in both graphs the static case produces marginally more AGs than any of the mobile cases, there is no significant difference between the results from the different speeds or the different stopping
Figure 7.14: Number of AGs chosen: The static case and the mobile cases with 80% of the nodes mobile with a 2 sec pause at waypoints.

The 80% mobility case is close to a worst case scenario in any deployment, such as the taxi scenario described in Section 7.2. At any given time, some of the taxis would be stationary with drivers waiting for passengers or taking a rest break.

In mobile deployments, the task of isolating compromised nodes by localizing to a specific cluster is challenging. To effectively do so in mobile deployments, efficient mobility support that can enable fast recovery is essential. An important requirement for this in our approach is CDS maintenance. It therefore, is imperative that the performance of the reclustering algorithm is not negatively impacted by node mobility and the speed of node movement.

From the results presented in Figure 7.10 to Figure 7.15, we observe that the mobility support procedures are efficient and ensure that performance of the reclustering algorithm is not degraded as a result of node mobility. We observe
that the number of AGs elected (i.e., clusters established) in the mobile case, compares well with the static case, which is the benchmark. This shows that the impact of node mobility is minimized on the clustering algorithm and it performs as well as in the static case. We also observe that the performance of the reclustering algorithm is not impacted by the speed of node mobility. In low mobility deployments (2m/sec) and in high mobility deployments (10m/sec), the performance of the reclustering algorithm again compares well with the static case once again being the benchmark.

Further, from Figure 7.16 we observe that the metric of the CDS is always higher than that of the full network. This proves that the choice of AGs by the reclustering algorithm proceeds in an energy-efficient fashion. This also has the advantage of reducing the occurrence of CDS breakage as a result of energy exhaustion in AGs, thereby reducing the need for excessive triggering of reclustering procedures.
The results presented prove the scalability of the proposed scheme both in terms of the proportion of mobile nodes in the network and the size of the network. We observe that the proportion of nodes required for the CDS is independent of network size and is approximately 20-25% of nodes deployed. It is also observed that the proportion of mobile nodes has a negligible impact on the size of the CDS.

These observations show that the proposed mobility support procedures are efficient to enable fast recovery in mobile network deployments.

The final experimental work in this section demonstrates the fact that the mobility support infrastructure proposed in this chapter is significantly superior to the basic mobility model without it. In order to demonstrate the impact of our mobility support on the recovery methodology, we ran the same code as described in this section but with conditions M1, M2 and M3 replaced by T1, T2 and T3, thus defaulting to the typical static case. The experiments were established in the same way as with the mobility model, running each size network 20 times for
30 minutes and determining the number of nodes left out of the cluster arrangements. We did this for node speeds of 2-3 m/sec, 3-5 m/sec and 5-10 m/sec, and also for the two different pause times of 2 secs and 5 secs. The results are shown in Figure 7.17 and Figure 7.18.

![Graph showing percentage of nodes included in a cluster in mobile WSN without mobility support and with 2 sec pause](image)

With the mobile support, we achieved 100% coverage of nodes, that is, each node in the network was included in some cluster after each reclustering. The constant line on the top of each figure indicates this situation. As the speed increases, the loss of mobility support results in an increasing number of nodes left outside clusters in the recovery process. In the case where 80% of the network is mobile, the WSN is essentially not functioning and would be considered 'dead'.

With mobility support, a break in the CDS is detected and reclustering is triggered to reconfigure the CDS. Without mobility support, a CDS break goes undetected and hence both network function and the recovery process fail. Thus, we measured the times at which the CDS broke when there was no mobility support. The results are shown in Table 7.3.
Figure 7.18: Percentage of nodes included in a cluster in a mobile WSN without mobility support and with a 5 sec pause

As expected, in each case, an increase in speed produces an earlier CDS breakage. An increase in the percentage of nodes which are mobile produces a similar result. However, increasing the pause time, which gives nodes a greater opportunity for assembling neighbour tables correctly, slows down CDS breakage. In Table 7.3, we ran each experiment 20 times and averaged the results for each case.

7.6 Resource Needed to Implement Our Protocol

In determining the efficiency of our approach, we note that restoring communication connectivity is based only on messaging between nodes and so determining the number of messages in a single re-cluster operation is a suitable measure of the cost to the network.
Table 7.3: Times of first CDS break in mobile WSN without mobility support

To this effect, we state and prove the following.

- **Theorem 2**: the WSN mobility support protocol described in Section 7.2.3 restores connectivity of a broken communication in the WSN with message complexity $O(N)$ where $N$ is the total number of nodes.

- **Proof**: the fact that the mobility support protocol restores connectivity follows from Wu [Wu03] as explained in the proof of Theorem 1. In order to determine the message complexity, we count the number of messages needed to re-cluster as indicated in the protocol:

1. Each node sends one message in order to establish a 1-hop neighbour table ($N$ messages).

2. Each node sends one message in order to establish a 2-hop neighbour table ($N$).
3. Each node sends one message in order to establish a metric table (N).

4. Each node sends one message in order to establish a coverage set table (N).

5. Each potential aggregator sends one message of invitation (between N/2 and N).

6. Each node accepts one offer (N minus the number used in 5). This step is optional.

In total, there are approximately 5N messages, and so the total message complexity is O(N).

- **Corollary**: If N is the total number of nodes in the WSN, then 5N messages is the maximum number needed in order to re-cluster the network.

- **Proof**: this computation was made in the proof of the Theorem.

In our protocol, there is no other type of cost in restoring connectivity. However, as described in Table 7.1, there are also non-messaging costs associated with some protocols. The papers [KAU10], [MYA10] and [IYSH10] share similarities with our work in that their aim is reconnection of a broken communication path in a mobile WSN. In both their work and ours, a metric is integrated into the recovery procedure in order to assist in reducing the cost, in terms of energy used, to the network. The major distinction between our work and theirs is that in all three of these papers, nodes are moved into position to cover a break in communication, whereas our solution is to logically re-organize the network. In comparing cost,
we consider the number of messages sent between nodes as both solutions rely on this. For our work, this is the entire solution; for theirs, we must add the actual cost of node movement.

The paper [MYA10] compares its protocol RIM with those in [KAU10] and demonstrates the superiority of RIM both in terms of number of messages used and number of nodes and total distance moved. The paper [IYSH10] introduces the protocol PCR which the authors then show is superior to RIM in terms of all three of the above-mentioned criteria. In this paper, the authors note that, when a node fails, the RIM protocol moves all the 1Chop neighbours towards it until they become connected. In contrast, PCR only moves non-critical nodes to avoid cascaded relocations; cascaded relocations are invoked only when non-critical nodes are not available. These authors also argue that as the communication range grows, RIM runs the risk of moving most of the WSN towards the failed node and so leaving the network periphery uncovered.

All three papers, ours, [IYSH10] and [MYA10] make use of a keepalive mechanism to detect a failed node. (see Section 7.2.4). The last two papers do not include these messages in the determination of the total message count for re-connectivity; thus, neither do we. Theorem 5 of [MYA10] indicates that, like us, the complexity of message use is $O(N)$ where $N$ is the size of the network.

In Table 7.4, we list the attributes of the AG nodes needed in the three papers in order to accomplish communication re-connection.

Table 7.4 indicates the high cost of using mobility as a solution to broken communication. We point out that in situations where motion is not controllable,
such as nodes falling or moving in a collapsing mine shaft, or in an earthquake, the solutions of [MYA10], [IYSH10] and [AAA09] are untenable. In situations where motion is necessary for the deployment task, such as gathering information as taxis cover the city, or collecting information about migratory patterns where motes are attached to birds, moving the nodes deliberately to fill communication gaps is not an option. Thus, our solution is not only significantly less expensive, but also more useful in a much wider range of situations.

### 7.7 Summary

A stationary WSN under attack can be supported by recovery mechanisms which are built on the regularity and stability of the reporting platform. However, many WSNs are deployed in situations which require their mobility and so lose the underlying stability which assists in maintaining message connectivity. A number of recent papers have dealt with this problem by leveraging the mobility itself and enabling nodes to move into communication holes in order to restore connections; however, to implement this solution, the nodes must be well-resourced and be able to move independently, which is not the case in all WSNs. In addition, there are many situations in which motion is an integral part of the deployment and cannot be re-allocate to form a solution.

In this chapter, we present a mobility support mechanism for mobile WSNs based only on node messaging, which is designed to restore communication connectivity. In order to demonstrate the effectiveness of this mobility support, we simulate a mobile environment with up to 1000 nodes. In this setting, we show that our
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical self-assessment (is the node a cut-vertex in the network topology?)</td>
<td>Critical self-assessment (is the node a cut-vertex in the network topology?)</td>
<td>Critical self-assessment (is the node a cut-vertex in the network topology?)</td>
<td></td>
<td>Has knowledge of all 1- and 2-hop neighbour IDs.</td>
</tr>
<tr>
<td>Needs knowledge of the positions of all AGs in order to do this.</td>
<td>Needs knowledge of the positions of all AGs in order to do this.</td>
<td>Needs knowledge of the positions of all AGs in order to do this.</td>
<td></td>
<td>Only capable of controlled, supported movement.</td>
</tr>
<tr>
<td>Mobility on Demand</td>
<td>Is able to move on demand.</td>
<td>Is able to move on demand.</td>
<td>Is able to move on demand.</td>
<td></td>
</tr>
<tr>
<td>Neighbour Knowledge</td>
<td>Has knowledge of location of all 1- and 2-hop neighbours.</td>
<td>Has knowledge of location of all 1-hop neighbours.</td>
<td>Has knowledge of location of all 1-hop neighbours.</td>
<td>Has knowledge of all 1- and 2-hop neighbour IDs.</td>
</tr>
</tbody>
</table>

Table 7.4: Attributes of the AG nodes in respective papers
mobility support is more suitable in a low-resource environment than current state-of-the-art methods. We also demonstrate that it is as efficient for a mobile WSN as for a static WSN in maintaining communication throughout the network and, consequently, in restoring communication in the WSN.

In comparing our work with that of authors who have used mobility as the solution for re-connecting a broken network, we argue that that solution has several drawbacks that make it impractical in many situations. In addition, we argue that our solution is significantly less expensive than theirs and so more appropriate in supporting low resourced WSN.

The next chapter summarizes the body of work presented in Chapters 3 to 7, and also provides a comparison with the work of other researchers on recovery in sensor networks. A new recovery standard is also proposed in Chapter 8 for future reference.
The limits of my language mean the limits of my world.

—Ludwig Wittgenstein
Chapter 8

Recovery Standards

8.1 Background

In this chapter we provide general recovery standards which can certainly be used in the sensor networks.

8.2 System Assumptions - A Review

As described in previous chapters, we assume a single, secure BS along with nodes capable of operating as either member nodes or AGs. The requirements for enabling a fast recovery scenario are:

- The BS is capable of determining when a node needs reprogramming.
- The BS is capable of reprogramming specific nodes in the WSN in an au-
The BS is capable of determining when the WSN needs to re-cluster.

- The BS is capable of initiating authenticated reclustering.

- AGs are capable of determining when a member node needs reprogramming.

- AGs are capable of initiating an authenticated member node reprogramming request to the BS.

The WSN is initially clustered by the BS into groups of nodes each monitored by an aggregator as shown in Figure 3.1.

We chose to use MicaZ and TelosB sensor nodes for implementation as both are supported by open source TinyOS [WC06]. Both are IEEE 802.15.4 compliant, a standard in implementing security protocols on motes, while the former is also ZigBee compliant. 802.15.4 defines the physical and MAC layers, whereas ZigBee defines the network and application layers. Long battery life, low cost, small footprint and mesh networking in supporting communication between large numbers of devices have driven the specifications for 802.15.4 and ZigBee.

As shown in Table 8.1, the MicaZ processor has a 4 MHz 8-bit Atmel ATMEGA103 CPU which consumes 5.5 mA (at 3 volts) when active, and about 0.06 mA when sleeping. The radio is a 916 MHz low-power radio from RFM, delivering up to 40 Kbps bandwidth on a single shared channel with a range of up to a few dozen meters. The RFM radio consumes 4.8 mA (at 3 volts) in receive mode, up to 12 mA in transmit mode, and 5A in sleep mode.
### Table 8.1: Specification comparison of MicaZ and TelosB

<table>
<thead>
<tr>
<th></th>
<th>MicaZ</th>
<th>TelosB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>4 MHz 8-bit</td>
<td>8 MHz</td>
</tr>
<tr>
<td></td>
<td>Atmel ATMEGA103</td>
<td>TI MSP430</td>
</tr>
<tr>
<td>Radio</td>
<td>916 MHz</td>
<td>2400 MHz to 2483 MHz</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>40 Kbps</td>
<td>250 Kbps</td>
</tr>
<tr>
<td>CPU Energy Consumption in Active Mode</td>
<td>5.5 mA</td>
<td>1.8 mA</td>
</tr>
<tr>
<td>CPU Energy Consumption in Sleep Mode</td>
<td>0.06 mA</td>
<td>5.1 uA</td>
</tr>
<tr>
<td>RFM Energy Consumption in Receive Mode</td>
<td>4.8 mA</td>
<td>4.8 mA</td>
</tr>
<tr>
<td>RFM Energy Consumption in Transmit Mode</td>
<td>12 mA</td>
<td>12 mA</td>
</tr>
<tr>
<td>RFM Energy Consumption in Sleep Mode</td>
<td>5 uA</td>
<td>5 uA</td>
</tr>
</tbody>
</table>
TelosB has an 8 MHz TI MSP430 microcontroller processor with 16 KB of instruction memory. The CPU consumes 1.8 mA (at 3 volts) when active, and 5.1uA power when sleeping. The radio is a 2400 MHz to 2483 MHz globally compatible ISM band, delivering up to 250 Kbps of bandwidth on a single shared channel and with a range of up to a few dozen meters. The RFM radio consumes 4.8 mA (at 3 volts) in receive mode, up to 12 mA in transmit mode, and 5A in sleep mode.

The main parameters of both motes can be seen in Table 6.4. MicaZ only supports 32 bit computation while the 16 bit processor of TelosB supports 64 bit computation.

8.3 Summary of the Needed Protocols - Stationary Case

In [LDBS09] and [LBD09], we presented protocols for reprogramming, reclustering and authentication as part of the recovery process in a WSN. We assume that the WSN is structured hierarchically with component roles as follows:

- Member node - senses and transmits data.
- AG - controls member nodes and senses, collects, aggregates, analyses and transmits data.
- BS - controls the system and collects, analyses, transmits and stores data.
As explained in detail in the research mentioned, to optimize the recovery process, we propose to retain connectivity and maximize flexibility in the network by allowing each node to play the role of either a member node or AG as appropriate under the conditions arising, and also at the same time, for the entire network to efficiently re-organize itself in order to remain connected. Authentication is implemented both in reclustering and reprogramming in order to avoid several typical attacks.

Each node $v$ broadcasts a HELLO message at a random time instant within a fixed time interval. Each message carries the ID of the transmitting node $v$, the IDs of all currently known 1-hop neighbours of $v$, and the metric values $M$ that characterize the capabilities of the node $v$ and its neighbours to act as an AG. The metric $M(v)$ of node $v$ is defined in Equation 8.1 as

$$M(v) = a \frac{e(v)}{e_{\text{max}}} + (1 - a) \frac{d(v)}{d_{\text{max}}}, a \in [0, 1] \quad (8.1)$$

where $e$ is the available battery energy of the node and $e_{\text{max}}$ its maximum value, $d$ is the node degree that should not exceed a pre-defined value $d_{\text{max}}$, and $a$ is a pre-defined weighting parameter. The key parameters $e$ and $d$ are chosen to maximize WSN energy and connectivity respectively.

- **Re-Clustering**

  A node $v$ decides to act as a AG if it has never been compromised, and

  1. it has a larger metric $M(v)$ than all its 1-hop neighbours and has at least two unconnected neighbours, or if
2. it is in the coverage set formed by its neighbour with the largest metric M.

- Reprogramming

Deluge is a reliable data dissemination protocol for large objects, such as program binaries. Together with a boot-loader, Deluge provides a way to reprogram sensor motes in a network. Since Deluge only supports network-wide reprogramming, we modified the dissemination engine of the protocol to individually address sensor nodes. This was done by replacing the AM_BROADCAST_ADDR parameter in the engine with the node ID of the node to be recovered. For reprogramming, we assume that each node in the WSN is within range of the BS, while the converse is not necessarily the case. This is a much stronger assumption than that needed for reclustering.

- Authentication

In each case, a node with ID n contains secret Sn, known only to itself and the BS. R represents a random value (but is in fact the local time obtained from the LocalTime.get() command in TinyOS), M represents a message to reprogram or recluster or a request that another node needs to be reprogrammed. H represents a hash function and we assume that n, M and R are the appropriate size for input to H. All message requests to reprogram or to re-cluster between nodes and nodes, or between the BS and nodes, include the node ID of both sender and receiver and a hash of data including the secrets known only to the sender and BS, and of the current value for R. This ensures the authenticity of the content and its sender, and
prevents resending of legitimate messages at a later time by an attacker.

For comparison purposes, we chose two hash functions in implementing the authentication protocol. SHA-1 was chosen as it is used in many government standards. Our second choice was the Rabin encryption system [Rab79] adapted for use as a hash function as proposed by Shamir in [Sha08a]. While Shamir suggests an adaptation of Rabin’s scheme to what he calls SQUASH, our implementation is constrained by TinyOS requirements and therefore, we use smaller values than those proposed to ensure the security of SQUASH. Such smaller values do not warrant use of the improved SQUASH computations, and so we use Rabin’s scheme as proposed in [Rab79]. The details can be found in [LBD09].

We make the assumption that keys are stored in a tamper resistant location and that an attacker is unable to retrieve them.

8.4 The Integrated Recovery Protocol - Modified EMPR and Reprogramming with Authentication

Given a WSN deployed in an environment in which the protocols of Chapter 7 are activated, the following procedures are followed as needed:

1. The BS determines, from information collected from the network, that reprogramming is required and initiates an authenticated reprogramming pro-
2. The BS determines, from information collected from the network, that reclustering is required and initiates an authenticated reclustering protocol.

3. The BS determines that the data being received from the network has deteriorated below a pre-determined quality target and that neither reprogramming nor reclustering improves this target. The BS shuts down operation.

8.5 Implementation Issues

In this section, we describe our integrated recovery protocol and the technical issues involved with the integration. This discussion should assist other implementers of WSNs with some of their decisions.

- Re-Clustering

We found that the hardware of our two platforms had an impact on program size after compilation. For the MicaZ platform, the reclustering algorithm requires about 126KB of RAM making it impossible to run due to the 4KB RAM limitation. However, TelosB consumes only 9.87KB after successful compilation of reclustering and so can comfortably accommodate the reclustering algorithm within the 10KB of RAM available.

- Reprogramming
As mentioned in Chapter 4, we needed to modify Deluge to reprogram individual sensors. This was done easily by replacing the AM_BROADCAST_ADDR parameter in the engine with the node ID of the node to be recovered. This allowed us to implement the reprogramming protocol effectively.

- Authentication

In order to transmit an authentication message using the IEEE 802.15.4 MAC protocol, 64 bit data is required to be fragmented on the sender side (the BS) and re-assembled on the receiver side (at the node) since a payload unit is only 8 bits. Use of either 64-bit SHA-1 code or 64-bit Rabin code consistently resulted in error responses.

We have discussed this problem with developers from the TinyOS community and believe that it could be occurring due to a bug in the GCC compiler or could be due to some unidentified hardware constraints. The problem disappeared on both platforms when we reduced our transmitted data size to 32 bit Rabin authentication.

- Integration.

We encountered several problems during the integration phase. Firstly, Deluge reprogramming methods use the original CC2420 driver from TinyOS Official, while our reclustering algorithm was designed based on a MAC protocol which uses a modified CC2420 chip driver in order to incorporate it into the IEEE 802.15.4 standard. In developing TinyOS, the driver set was not designed to be compatible with the IEEE 802.15.4 standard and because Deluge reprogramming was also designed based on this driver
set, Deluge is not compatible with IEEE 802.15.4, and therefore, also not compatible with our reclustering algorithm.

In addition, Deluge requires large volumes of RAM and ROM on both our platforms; at least 1.13KB RAM and 31.95KB of ROM on MicaZ; 1.31KB of RAM and 37.40KB of ROM on TelosB.

There are several possible solutions to these problems, including development of a new CC2420 chip driver set based on the IEEE 802.15.4 protocol in the first instance, and revising Deluge to significantly reduce the memory requirements in the second instance. Both of these solutions require resources, time and people and are beyond our project scope at the moment and therefore, they are for future projects.

We also mention here two issues that arise from the hardware that had an impact on our protocols.

• Clock skewing.

In implementing the reclustering algorithm, we encountered time latency due to the limited processing capacity on both platforms and also due to network traffic collisions. We were unable to directly deal with time latencies caused by the processing capacity, but covered this problem by allowing larger time intervals than might otherwise be used when dealing with traffic collisions. This process is described under the heading ‘collision resistance’ below.

• Collision resistance.

Traffic collisions occur when a mote attempts to receive more than one
message simultaneously. The Medium Access protocol (MAC) supported by
the underlying TinyOS operating on both platforms, includes the standard
CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) to manage
the network traffic in an optimal way and so reduce the packet loss rate as
much as possible.

CSMA/CA is the channel access mechanism used by most wireless LANs,
specifying how the node uses the medium, for example when to listen, when
to transmit etc.

In addition to employing the MAC with CMSA/CA, we added a method
similar to node beaconing used in [KCA07]. To improve collision avoidance
and allow for clock skew caused by the hardware, we set a small random
variation for the time interval to prevent multiple nodes from broadcasting
messages simultaneously.

Some of the implementation issues which arose were due to hardware ca-
pacity, while some were due to the associated standards embedded in their
design: TinyOS, MAC, Zigbee and IEEE 802.15.4.

Table 8.2 summarizes the capacity of the two platforms to deal with the
protocol components.

- Standards integration.

There is a lack of support in the motes themselves for proper integration
with standards defined by MACs, Zigbee and IEEE 802.15.4, thus making
integration across some platforms difficult, if not impossible. Dealing with
this is still an open question in the research community, though [KCA07]
### Table 8.2: Capabilities of the two platforms

('Yes' means the algorithm is fully supported by a platform; 'No' means the algorithm is not supported by the platform.)

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>TelosB</th>
<th>MicaZ</th>
<th>Reasons for 'No'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deluge Reprogramming</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Authentication protocols</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 8bit SHA-1</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 8bit Rabin</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 16bit SHA-1</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 16bit Rabin</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 32bit SHA-1</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 32bit Rabin</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Standalone 64bit SHA-1</td>
<td>Yes</td>
<td>No</td>
<td>Limited computing capacity</td>
</tr>
<tr>
<td>Standalone 64bit Rabin</td>
<td>Yes</td>
<td>No</td>
<td>Limited computing capacity</td>
</tr>
<tr>
<td>Reclustering</td>
<td>Yes</td>
<td>No</td>
<td>Limited RAM size</td>
</tr>
</tbody>
</table>
| Deluge Reprogramming + SHA-1
authentication | Yes    | Yes   |                                   |
| Deluge Reprogramming + Rabin
authentication | Yes    | Yes   |                                   |
| Reclustering + Rabin
authentication | Yes    | No    | Limited RAM size                  |
| Deluge Reprogramming + reclustering | No     | No    | Different sets of CC2420 chip
driver                                      |
| Deluge Reprogramming + authentication + reclustering | No | No | Limited RAM size and different sets of CC2420 chip driver |

8.6 Comparison with Other Work

In this section, we compare our implementation with those of other researchers who have developed recovery protocols for a similar situation. The difficulties
CHAPTER 8. RECOVERY STANDARDS

encountered in implementation and the work-arounds developed are a particular focus, as these are integral to our recommendations. There are three recent papers which have implemented protocols on low resource motes.

[LL07] implements 27 Crossbow MicaZ motes in a coal mine to detect collapsed areas and to maintain system integrity, especially in case motes are moved or destroyed in a collapse. In [WBLS07], the authors implement data delivery mechanisms on MicaZ motes to determine performance limitations in obtaining 'accurate modeling of wireless signal propagation behaviour and erroneous device behaviour'. Finally, the paper [KCA07] discusses the use of Zigbee together with IEEE 802.15.4 standards, and highlights they do not support time synchronization to support clusters. They propose a work-around for this, and perform testing on MicaZ motes. We describe the work in each of these papers in more detail below, commenting particularly on the implementation problems they had and how they worked around them.

The paper [LL07] implements 27 Crossbow MicaZ motes in a coal mine to detect collapsed areas and to maintain system integrity, especially in case motes are moved or destroyed in a collapse. Thus, they needed to allow for frequent re-verification of neighbour nodes. As the neighbour table method we use is extremely expensive in such a situation, they replace this by a beacon method whereby nodes regularly report their existence to the network. In addition, since a collapse initiates a flurry of correspondence within the network, the authors must deal with a serious collision message problem; in fact flooding through the network can occur. In order to reduce collisions, they employ two methodologies: randomized forward latency to hold back messages from the less important areas
of the WSN after a collapse and data aggregation is the second method applied, with C motes at the edge of a collapse collecting messages from the area and aggregating the data in them into a single report message to the BS. Data aggregation is a major component of our own methodology as the AGs of clusters perform this regularly.

In [WBLS07], the authors implement data delivery mechanisms on MicaZ motes to determine performance limitations in obtaining ‘accurate modeling of wireless signal propagation behaviour and erroneous device behaviour’. The authors point out that changes in the environment can affect WSN performance and so focus on fault tolerant routing. They use acknowledgment packets when finding traffic routes, despite the extra load on the network, arguing that the trade-off in reliability is worth it. However, in compensation, their protocols limit acknowledgments to motes close to the destination of the transmitted packet. They also implement wait-time periods during which a mote listens for a response, but after which responses are ignored. Thus, the number of acknowledgements is kept relatively small. In evaluating their protocols, the authors measure the time required for a packet to traverse a WSN. However, this measurement is affected by clock skew C drift in the internal mote clocks. They compensate for this with linear regression methods. In our work, acknowledgment of packets for the purposes of authentication is necessary. Thus, where authentication is critical, we need all acknowledgments. We dealt with clock skew in our protocols by using time lags in order to allow for receipt of packets.

Finally, the paper [KCA07] discusses the use of Zigbee together with IEEE 802.15.4 standards, and points out that they do not support time synchroniza-
tion for cluster topologies. They propose a work-around for this, based on time
divisioning beaconing, and testing is performed on MicaZ motes supported by
TinyOS. The protocol allocates a packet sending timing schedule to each AG and
its cluster in order to avoid collisions between packets sent within the cluster.
A 'super-frame' schedule is also allocated across the WSN. Their protocol aims
to align itself with the IEEE 802.14.5 protocol specification which, they point
out, does not support a beacon-only approach. Thus, their work heads towards
filling a gap in that standard. As noted in our discussion on integration issues in
the previous section, we also encountered difficulties with existing specifications.
The authors of [KCA07] also identify clock skew as an issue as did the authors
of [WBLS07].

In summary, the issues raised in the above papers are clock skewing and collision
resistance. In addition, to these issues, in our work and that of [KCA07], a
lack of support is identified in the motes themselves for proper integration with
standards defined by MACs, Zigbee and IEEE 802.15.4, thus making integration
across some platforms difficult, if not impossible.

Based on this analysis and comparison, we now provide recommendations for
consideration by researchers who may consider integrating several protocols on
these platforms in the future.

- The use of data aggregation in order to reduce collisions.
- A reduction in the number of acknowledgment packets where possible.
- Integration of the time division beaconing methods of [KCA07] with the
  IEEE specifications for routing.
• Recognition of the difficulties in integrating Zigbee and IEEE 802.15.4 across some platforms.

In particular, and of relevance to the last point, the TinyOS driver set needs to be redesigned so as to incorporate the IEEE 802.15.4 standard.

Table 8.3 summarizes the main implementation issues identified in recent work.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Clock Skewing</th>
<th>Collision Resistance</th>
<th>Standards Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. [LL07]</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wasilewski et al. [WBLS07]</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Koubaa et al. [KCA07]</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Our Approach</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 8.3: Implementation issues identified in recent work

8.7 A Recovery Standard Proposal for WSNs

As a result of the work in this thesis, we are able to propose a standard for the design and implementation of WSNs. We begin with the scope of the standard, along with definitions and general requirements, and then provide the design and implementation details. The types of measurements needed are then addressed followed, finally, by a discussion of how improvements can be identified and implemented.
8.7.1 Scope

This proposed standard applies to a single, secure BS along with sensor nodes capable of operating as either member nodes or AGs. Sensors collect data and transmit it to the BS. The BS can analyse incoming data and, based on it, makes decisions about the state of the nodes and of the network.

The network is assumed to be stationary and low resourced but provides the capability to self-organize (cluster), detect an attack on the nodes, determine which nodes have been compromised (with high probability) and make a decision to reprogram compromised nodes.

The aim of this proposed standard is to effect efficient recovery and continued operation with the objective of maintaining the life of the network for as long as possible.

8.7.2 Terms and Definitions

8.7.3 General Requirements

Any sensor nodes supported by open source TinyOS [WC06], and which are IEEE 802.15.4 compliant, can be used. Sensors which are also ZigBee compliant are recommended.
8.7.4 Design

In order to optimize the recovery process: each node plays the role of either a member node or AG as appropriate under the conditions arising, including energy remaining and number of neighbours.

In order to recover compromised nodes: reprogramming of all or of any subset of the nodes is possible.

In order to recover disconnected communications: reclustering of the WSN is possible.

In order to avoid several typical attacks: authentication is implemented in the recovery protocols.

8.7.5 Implementation

There are three protocols to be implemented:

- Reclustering

  This protocol should be initiated by the BS. The choice of AGs is the main step in reclustering, as the member nodes of an AG’s cluster are then determined by its communication range.

  A node \( v \) provided with a metric \( M(v) \) which measures its energy level and number of neighbours acts as a AG if it has never been compromised, and

  1. it has a larger metric \( M(v) \) than all of its 1-hop neighbours and has at least two unconnected neighbours, or if
2. it is in the coverage set formed by its neighbour with the largest metric M.

- Reprogramming

Modify Deluge in order to reprogram individual sensor nodes by replacing the AM_BROADCAST_ADDR parameter in the engine with the node ID of the node to be recovered. This allows effective implementation of the reprogramming protocol.

It is assumed that the entire WSN is stationary and within range of the BS; however, it is not necessary for the BS to be in range of each node.

- Authentication

Each node shares a common secret with the BS. In communications between nodes and the BS associated with reclustering or reprogramming, authenticated messages, using a combination of ID, the shared secret and a current random number (which may be the network time), are used. Any hash function acceptable as a standard to hash this data as a verification tool. No node knows the secret of any other node.

8.7.6 Measurements

The measures of interest in this standard are:

- Execution time of each protocol described in Chapter 7 separately.

- RAM and ROM needed to perform each protocol separately.
- WSN life-time given a fixed number of nodes and under various attack scenarios.

8.7.7 Improvement

Changes in hardware and improvements in computing speed may affect some of the components of this proposed standard. Future versions of the standard should take these into consideration.

8.8 Summary

In this project, we have presented an integrated recovery technique implemented on TelosB and MicaZ motes in a low resourced, stationary and mobile deployment. We described the issues arising in integrating several separate protocols reclustered, reprogramming and authentication, using Zigbee with IEEE 802.15.4 and the reprogramming module Deluge, and compared our challenges and solutions with those faced by other researchers in this area.

We have also proposed a standard for the implementation of recovery in stationary WSNs.

Based on this analysis and comparison, we now provide recommendations for consideration by researchers who may consider integrating several protocols on these platforms in the future.

- The use of data aggregation in order to reduce collisions.
• A reduction in the number of acknowledgment packets where possible.

• Integration of the time division beaconing methods of [KCA07] with the IEEE specifications for routing.

• Recognition of the difficulties in integrating Zigbee and IEEE 802.15.4 across some platforms.

In particular, attention needs to be given to the fact that the TinyOS driver set needs to be redesigned so as to incorporate the IEEE 802.15.4 standard.

In the next and final chapter, the implications of this work and the directions for future work is presented.

My work consists of two parts: of the one which is here, and of everything which I have not written. And precisely this second part is the important one.

—C. Grant Luckhard
Chapter 9

Conclusions

The research of security in WSNs has grown rapidly in recent years. In this thesis, we presented the problem of how to recover WSNs from attack. This has become a very serious issue to the current network infrastructure. The aim of our research is to investigate principal approaches for effective and efficient recovery mechanisms; to develop general recovery mechanisms which can cope with most attacks targeting on WSNs; and to explore fundamental theoretical frameworks that can support the development and deployment of such systems. As a result, we present a recovery approach that includes a clustering algorithm for low-resource stationary WSNs in conjunction with a reprogramming mechanism to enable efficient recovery from node compromise while maintaining the operation of the network. The recovery approach takes into account the state of individual sensors to ensure energy-sensitive cluster and network organization.

In this chapter, we examine how successful the project has been with respect to
the original requirements given in Chapter 1. It summarizes the main contributions of this thesis on the recovery of WSNs, and draws overall conclusions, and highlights any areas where further work could be carried out.

In Section 9.1 we present the achievements of the thesis. Section 9.2 provides several suggestions for future work.

9.1 Achievements

The contribution of this thesis can be summarized as:

- A recovery mechanism that combines network reclustering and node reprogramming to dynamically re-organize the network into a clustered network architecture that does not include compromised nodes, thus permitting the network to be operational while initiating node reprogramming procedures to recover compromised nodes.

- Demonstration that the proposed recovery mechanism comprising of network reclustering and node reprogramming can be implemented efficiently on both a sensor network testbed using TelosB motes and NS-2 simulation platform.

- An efficient authenticated reprogramming request protocol from the BS to a compromised node, from an AG to the BS on behalf of a compromised node and from the BS to the network.

- A comparison of the performance of the recovery protocols between the
stationary and mobile WSNs demonstrating that our recovery protocol for mobile WSNs is best possible.

9.2 Future Work

The proposed distributed and decision-making recovery protocols presented in this thesis are the first of their kind and is a methodology by which the sensor networks can assess the levels of damage from attacks and choose relevant recovery techniques to restore the nodes or even the networks. In this section, we discuss future work that we will be undertaking in order to strengthen the proposed methodology for the sensor network recovery. It should be noted that future work is not limited to the areas discussed.

First and foremost, due to the limited resources available to us, we did not test our recovery mechanisms on large scale physical WSNs but on a simulation platform. It would be very interesting to see the test results from the network with hundreds of motes, especially for the reclusteri ng algorithms. Potentially, this kind of research testbed can benefit other researchers by providing a standard data set. More research could be undertaken to compare the effectiveness and efficiency of applying such design to the mission critical areas such as battlefields.

Secondly, our authentication protocols were developed based on TelosB and MicaZ. The poor computation capabilities and the small size of memory determined that no powerful authentication can be performed on those two platforms. Some customized hardware can also be considered in the future to run some better
CHAPTER 9. CONCLUSIONS

authentication functions.

Next, with regard to the assumptions made in Chapter 5, we require the use of a shared secret to authenticate reprogramming and reclustering for the purposes of recovery. This shared secret is also useful for authenticating the legitimacy of sensor data messages from the network, which helps to avoid other types of attacks.

We assume that secrets are stored in a tamper-resistant section of the node and that calculations involving it are executed in this tamper-proof section. While the use of a tamper-proof or secure area for storing secrets and executing computations is a standard solution to key management, it is difficult to design and implement such a secure area on a small device such as a mote. In future work, we would be interested in investigating the possibilities of building such a secure environment in WSN motes.

In our situation, the sensors were permitted to move around a convex area. This assists in ensuring that, at any time, some node is within range of the BS as the radius of communication of the BS is likely to be circular (or spherical). However, in a practical deployment of a mobile WSN, many nodes may not be in range of the BS at all times. There are two possibilities for research here. One refers to a situation in which the range of the BS is limited and nodes are allowed to move out of it; the other refers to a non-convex area where, for instance, stone or metal walls block transmission. Both areas are of interest and models for both need to be developed.

Last but not least, in the real world, WSNs still have to be exposed to many
potential threats, such as weather, animal interference etc. How our recovery protocols perform in extreme conditions could raise more interesting research motivations.

In anything at all, perfection is usually attained not when there is no longer anything to add, but when there is no longer anything to take away.

—Antoine de Saint-Exupéry
Glossary

**AG**  Aggregator Node, 3

**BS**  Base Station, 3

**CDS**  Connected Dominating Set, 25

**IEEE**  Institute of Electrical and Electronics Engineers, 50

**MAC**  Medium Access Control, 44

**MPR**  Multi-point Relays, 9

**RAM**  Random-access Memory, 14

**ROM**  Read-only Memory, 96

**WSN**  Wireless Sensor Network, 1
Appendix A

MPR Protocol

In this appendix, a brief description of MPR protocols is present to show how it operates in sensor networks.

The original MPR is source-dependent; that is, the forward node set is dependent on the source of the broadcast and communication latency. Recently, Adjih, Jacquet, and Viennot [AJV02] proposed a source-independent MPR. The CDS is constructed based on MPR following two simple rules. In this section of the thesis, we enhance the source-independent MPR through a simple modification. [Wu03] proposed an enhanced source independent MPR based on the recently proposed source-independent MPR. The enhancement is done without increasing the complexity of the method. The effectiveness of the enhancement is confirmed through a simulation study on both sparse and dense networks.
A.1 MPR

Let $N(V)$ denote the set of all nodes that are in $V$ or have a neighbour in $V$. $V$ covers $U$ if $U \subset N(V)$. In MPR [QVL02], each node $v$ maintains 2-hop neighbour set $N(N(v))$. Node $v$ selects a small forward node set, $C(v)$, from its 1-hop neighbour set to cover its 2-hop neighbour set; that is, $C(v) \cup v$ is a CDS for $N(N(v))$. $C(v)$ is also called the coverage set for $v$. When $u$ is selected by $v$ as a forward node, $v$ is called the selector of $u$. Note that several selectors may exist for a particular node. A forward node may or may not actually retransmit the message; its actual status is determined by the following MPR rule.

- **MPR rule**: a node re-transmits the message once and only once if the first message received is from a selector.

The collection of nodes that have re-transmitted the message plus the source node form a CDS.

Let $N_1(V) = N(V) - V$ denote the nodes at distance one from $V$ and $N_2(V) = N(N(V)) - N(V)$ denote the nodes at distance two from $V$. A simple greedy algorithm for computing $C(v)$ (initially empty) at $v$ is as follows.

**Algorithm 3 Greedy Algorithm**

- Add $u \in N_1(v)$ to $C(v)$, if there is a node in $N_2(v)$ covered only by $u$.
- Add $u \in N_1(v)$ to $C(v)$, if $u$ covers the largest number of uncovered nodes in $N_2(v)$ that have not been covered.
A.2 Source-independent MPR

The original MPR is source-dependent. Adjih, Jacquet, and Viennot [AJV02] recently proposed a localized algorithm to construct a CDS based on MPR, and it is source-independent. A node belongs to a CDS if

- **Rule 1**: the node has a smaller ID than all its neighbours.
- **Rule 2**: the node is a forward node selected by its neighbour with the smallest ID.

A.3 Enhanced MPR

A.3.1 Enhanced Rule 1

In the following, we propose two extensions to source independent MPR: one on Rule 1 and the other on the greedy algorithm.

- **Enhanced Rule 1**: the node has a smaller ID than all its neighbours and it has two unconnected neighbours.

The Enhanced Rule 1 together with the original Rule 2 will generate a CDS under all cases except complete graphs. Note that when the network is complete, there is no need for CDS, because each source forms a CDS.

Theorem 1: If the given graph is not a complete graph, the set of forward nodes selected by the Enhanced Rule 1 and Rule 2 forms a CDS.
APPENDIX A. MPR PROTOCOL

Proof: It has been shown in [AJV02] that forward nodes selected by Rule 1 and Rule 2 form a CDS. We only need to show that whenever a smallest ID node v within its 1-hop neighbourhood is removed based on the Enhanced Rule 1, the resultant forward nodes still form a CDS.

Because the graph is not a complete graph and all of v’s neighbours are pairwise connected, there must exist a node that is not a neighbour of v. Let w be such a node with the smallest ID. Since v has the smallest ID in its 1-hop neighbourhood, either v or w has the smallest ID in the 1-hop neighbourhood of any neighbour of v. When one neighbour of v, say u, is selected by its smallest ID neighbour v (w) to reach w (v) in MPR, based on Rule 2, u is a forward node and it covers v and all neighbours of v. Therefore, v can be removed.

A.3.2 Enhanced forward node selection

Node u is a free neighbour of v if v is not the smallest ID neighbour of u. In the enhanced forward node selection, we first include all free neighbours, then nodes with higher degrees (i.e., covering more uncovered 2-hop neighbours) are selected and use the node ID to break a tie if needed until N_2(v) is covered.

Algorithm 4 Extended Greedy Algorithm

Add \( u \in N_1(v) \) to \( C(v) \), if there is a node in \( N_2(v) \) covered only by u.
Add \( u \in N_1(v) \) to \( C(v) \), if u covers the largest number of uncovered nodes in \( N_2(v) \) that have not been covered by the current \( C(v) \). Use node ID to break a tie when two nodes cover the same number of uncovered nodes.
Bibliography


[HLBL06] D. Herbert, Y.H. Lu, S. Bagchi, and Z. Li. Detection and repair of software errors in hierarchical sensor networks. *In IEEE Interna-


