The University of York Department of Computer Science

Submitted in part fulfilment for the degree of MEng in Computer Science with Embedded Systems Engineering.

Developing a platform for common evaluation of self-healing in wireless sensor networks

Andrew Durant

DRAFT PROCESSED 26th April 2016

Supervisor: James Harbin

Number of words = 0, as counted by texcount -sum -1 project-report.tex.
This includes the body of the report only.

Abstract

Wireless sensor networks are used in a wide range of applications, and in recent times this is only expanding. Because they are small, low-power devices, and the common network connectivity uses multi-hop routing, network drop-outs and partitioning of devices is a common problem. According to Tong et al. [1] self-healing by means of mobile nodes still remains a greatly unstudied area. However, more recently developments have been made in diverse ways.

There are two main approaches to network and device management for combating and fixing these issues: the distributed approach and the centralised approach. Each of these are effective for different types of applications. The applications that distributed approaches are suited to tend to be those where manual intervention is more difficult once a network is deployed. For this reason self-healing and autonomy within the network is of greater importance and this project focuses on these applications. A variety of algorithms from both approaches are presented to show this, then particular attention is given to Recovery through Inward Motion (RIM).

The wide variety of different implementations of self-healing share some performance metrics, but are often measured in different ways and using differing scenarios. This makes it difficult to compare and evaluate self-healing algorithms for use within a practical system. Because of this a common platform for comparison between approaches to self-healing is desired. This project aims to develop such a platform and to evaluate it using existing approaches to self-healing, again using RIM as an example. Finally concluding with suggestions for future developments in the field of self-healing wireless sensor networks and further development of the simulation platform.

Acknowledgements

I would like the thank James Harbin for his supervision and guidance throughout this project which has been invaluable; from questions leading to all kinds of interesting discussions to generally improving my academic rigour. I am also thankful for my family and friends who have supported me, given me new perspectives, and pushed me to clarify and improve my explanations and examples.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Interest in embedded systems and their applications is ever growing in our society. In recent years the cost of developing small systems on chip and running software on embedded computing platforms has plummeted. This has particularly driven the so called 'Internet of Things'; filling the world with Internet connected devices to analyse, report on, and control any system in any location from any other remote location. For many of these devices, alongside providing convenience to their users, the goal is collecting data from the environment. This data may be used to provide insight into how we interact with the world, to improve and guide the development of infrastructure, or for many other applications.

Wireless Sensor Networks (WSNs) are one specific area of this domain with a wide variety of applications for data gathering within a region of interest. They provide an independent system that can be deployed in any location, from public buildings to forests to disaster zones. One consequence of this is that the networks need to be highly adaptable, and are often designed for bespoke situations. If there are changes to their environment the networks need to be able to cope and continue their mission critical objectives. Because wireless sensor networks communicate through multi-hop routing, if links in the communication fail the network becomes dysfunctional and data will be lost from any network partition that is separate from the sink or root node.

To combat this, self-healing processes have been developed for wireless sensor networks. Often the deployment of nodes is done into inaccessible locations, so returning to manually adapt the network is not possible or desirable. One of the key advantages of wireless sensor networks is their low cost; often to the extent of disposability. A large proportion of this cost is the deployment process. If increasing the development cost a relatively small amount to include self-healing can provide a longer lasting, more reliable network that needs no further intervention once deployed, the cost is negligible compared to the advantages.

When developing wireless sensor networks a variety of self-healing algorithms are available for use, however the differences in their own

evaluation lead to difficulties in comparing and analysing them for use in a particular application.

1.1 Project Aims

This project aims to investigate self-healing in wireless sensor networks. Research will investigate the current developments in this field leading to a need for a common evaluation tool for approaches to self-healing.

The goal of this project is to produce a wireless sensor network simulation platform as a reference architecture for evaluating self-healing algorithms. The applications of this range from academic investigation during the development of new algorithms or the improvement of existing algorithms, to practical evaluation of candidate systems prior to development and deployment. The platform will be able to simulate and visualise any algorithm implementation on a realistic network.

The project does not aim to produce a novel self-healing algorithm, but to independently re-implement algorithms from the literature. Through doing so critical analysis can be performed and flaws that were not discovered in their initial reporting can be shown. This evaluation will allow proposals for what features are critical for general and specific applications, and suggestions for areas in which it would be useful to focus future research and algorithm development.

1.2 Report Structure

The body of this report consists of 5 chapters that each describe the tasks within the project.

In chapter 2 the existing literature and work in self-healing wireless sensor networks is reviewed to gain an in-depth understanding of the current state of research, the problems being solved, and how those solutions are evaluated. Each of these topics are categorised in their own section.

Analysis of the problem and the scope of the project is described in chapter 3; this chapter sets out the objectives that this project is aiming to achieve.

The design and implementation of the project is detailed in chapter 4, describing the decisions made and the reasoning for them, and the specifics of how the system has been built.

Chapter 5 evaluates the systems produced in chapter 4 against the metrics discussed from chapter 2. The chapter also evaluates the platform itself.

The report is concluded in chapter 6 with a review of the results produced and the conclusions that can be made from them. The chapter returns to the objectives defined in chapter 3 and reviews their status. The report is then finished with suggestions for future work in this area, and further development of the platform.

1.3 Statement of Ethics

The intended application of this project is academic. However, there are potential applications for the methods and algorithms discussed for monitoring or affecting humans, animals or the environment. This project does not include such examples and carries no malicious intent.

The project does not involve any human participants. No sensitive information, that could have a negative impact on any person or organisation, has been gathered during the project.

2 Literature Review

In order to consider self-healing in real-world applications of wireless sensor networks an understanding of the possible applications is required. A survey of approaches to self-healing is then presented along with metrics that can be used to evaluate and compare them. This is followed with a review of evaluation techniques and real-world factors for wireless sensor networks.

2.1 WSN applications

The possible applications for wireless sensor networks are diverse and far reaching. The small self-powered devices that can gather data from a wide area, to detect events and communicate information back to a base station either processed or for processing, lend themselves to a large number of applications. Each of these applications utilise the WSN in different ways and thereby bring new challenges to the developer of the systems. Due to this the design of the network is highly dependent on the application domain. Just some of the fields in which WSNs have been used are environmental monitoring, warfare, agriculture, surveillance, medical care, education and micro-surgery [2–7].

Habitat monitoring using WSNs has been taking place on an island of ducks where ordinary monitoring techniques cause disturbance to the animals with problems such as increasing mortality and eventual abandonment of the habitat. The use of WSNs that can be deployed prior to seasonal habitation enables constant monitoring over the time without the need to disturb the inhabitants. Mainwaring et al. [8] detail the requirements of the network and how it has been developed. Longevity of the devices, remote management, and inconspicuous operation are required to ensure the WSN is reliable and the data is able to be collected. The locations used for WSNs are often extreme, particularly for habitat monitoring. Similar requirements to the above are generated by Biagioni and Bridges [9] for monitoring endangered plants. Most importantly however for collecting scientific data is ensuring the system is reliable

and the user can have confidence in the data. This is achieved through conventional means of high capacity batteries and using sleep cycles on the nodes, alongside specialised routing algorithms, specialised sensors, and discreet enclosures to prevent tampering.

A very different approach was taken by Juang et al. [10] in developing ZebraNet. Rather than statically positioned nodes, the devices are attached to the herd of zebra being monitored with collars. The nature of this system is entirely mobile, without even a fixed base station to report to. Because of this the nodes pass data between them when within range forming a distributed collection and storage network. The mobile base station can then be deployed occasionally to collect the dataset, only needing to meet a few devices to obtain all of the data gathered by the network. This network is inherently ad-hoc where nodes may or may not come into contact with one another frequently, and there is no fixed or permanent base station to report to. Again longevity is an important requirement as deploying or re-deploying the sensors requires capture of the animals, so the systems must run for at least a year with no intervention.

Environmental monitoring on volcanoes has been investigated by Werner-Allen et al. [11] where the use of WSNs enables data collection over a wider area for a longer amount of time compared to using traditional equipment. WSN nodes can be deployed easily and left unattended to monitor the volcano area over a period of several weeks or months. However traditional equipment is bulky and heavy, and therefore difficult to deploy, limiting the area of research. The WSN transmitting the collected data back to the base station also removes the need to frequently return to the sensor devices to retrieve data. This is a clear example of the advantages of using wireless sensor networks. However a number of potential issues are raised. The nodes are deployed at the limits of their range to the nearest neighbour: this allows as large an area as possible to be covered, however if a single node fails all other nodes that were dependent upon it will be unreachable. In the scale of this network, where the maximum hop length from the base station is 6 this may seem insignificant, however that is 37.5% of the network nodes.

2.2 Approaches to self-healing in WSNs

2.2.1 Centralised

The main way centralised approaches are used in WSNs is in the configuration of the network structure to minimise energy use and ensure good coverage of the area of interest [12–15]. Centralised evaluation of the network and area of interest prior to deployment ensures the connectivity and coverage requirements of the application are met.

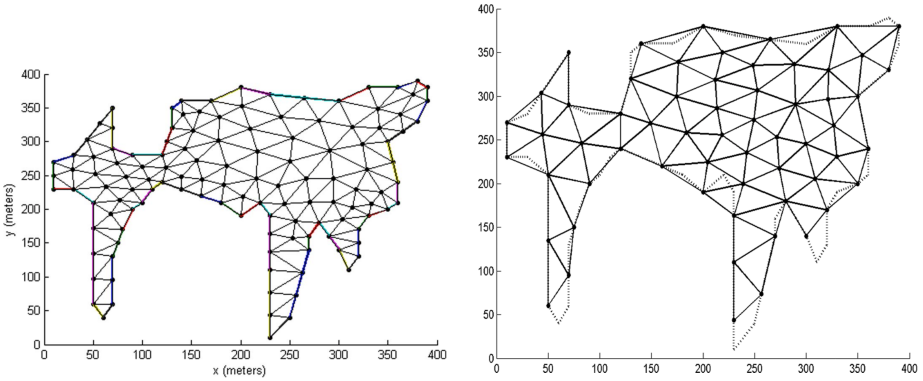
> Coverage requires that every location in the sensing field is monitored by at least one sensor. Connectivity requires that the network is not partitioned in terms of nodes' communication capability. Note that coverage is affected by sensors' sensitivity, while connectivity is influenced by sensors' communication ranges. [14]

These vary by application for instance, detection and localisation of events require multiple nodes covering smaller areas (a dense coverage) whereas other applications may only need one node within its transmission range (a sparse coverage). By pre-planning to match the deployment to the application needs the network will be much more efficient in power use and accurate in data collection.

Networks with a high degree of connectivity are more resilient to node failures. Wang et al. [12] investigate the relationship between coverage and connectivity to produce an integrated solution that is able to maintain both requirements whilst reducing energy consumption. The degree of connectivity is described as K-connected, where any $K-1$ nodes may fail without losing network connectivity. So the higher the connectivity ($> K$) the more nodes are able to fail without disrupting the rest of the functioning network.

Connectivity is important as simply achieving area coverage would result in nodes being placed too thinly in the event of failures, either requiring a large movement in the network or simply not being able to recover. The main goal of pre-planned centralised approaches is to create a deployment that is resilient without further intervention.

Due to the differences in applications a number of application specific methods have been proposed in addition to those previously mentioned [16–18]. All of these provide for different application requirements in different ways. However a single parametrised method that can be applied to many differing applications has been developed by Derr and



(a) Mesh generation with NETGEN, maximum edge length of 40; 126 nodes.

(b) NETGEN mesh simplification with INRCDTS algorithm; 60 nodes.

Figure 2.1: Mesh generation and simplification. Reproduced from Derr and Manic [15].

Manic [15]. This allows the specification of the number of nodes and the distance between the nodes which defines the desired coverage and connectivity respectively. The algorithm first constructs a triangle based mesh for the given area of interest, for example with NETGEN (figure 2.1a); however this creates more nodes than desired. To reduce the number of nodes Derr and Manic create the mesh simplification algorithm INRCDTS which removes nodes and smooths the mesh until the required number of nodes from the desired network parameters is reached (figure 2.1b). To ensure that both coverage and connectivity requirements are met the algorithm assumes that the communication range (R_C) is greater than $\sqrt{3}$ of the sensing range (R_S) i.e. $R_C \geq \sqrt{3}R_S$. Based upon this assumption the node's spacing is determined by $\sqrt{3}R_S$ which ensures coverage and therefore connectivity for the network.

Most centralised approaches use a priori information to specify exact deployment in the area of interest. However Qu and Georgakopoulos [19] have developed a post deployment particle swarm optimization (PSO) algorithm that is computed centrally to provide optimal coverage and reduce energy consumption. A centralised algorithm is able to understand the entire network, available devices, and area, and therefore produce the optimal distribution of nodes. The nodes are on a mobile platform that is able to move within the area. Reduction of energy consumption is done by dynamically reducing the range of sensing and

communication on each device to the minimum needed once the locations have been determined. Qu and Georgakopoulos do not address node failures, as they assume all nodes are working throughout the time of use. The distance measurement is also assumed to be for a point-to-point movement with no consideration for difficulties navigating the terrain. The energy considerations take account of the mobility and sensing of the node, but the other components' power usage is ignored as it is not increased by the algorithm. The centralised calculations for deployment take place on a device with unlimited power. The significance of the algorithm is dependent upon the relative energy consumption of sensing and movement. If sensing requires more energy then the algorithm can provide significant improvements, however when movement has a higher energy cost the saving is negligible.

No consideration is made for the energy cost of transmitting additional messages back to the root node: this is a concern as if all messages must be relayed back, the nodes near the top of the tree will have a significantly larger number of messages to receive and transmit, reducing their lifetime dramatically. This is partly mitigated by the algorithm only running at the deployment stage rather than throughout, however the nodes closest to the root will still be depleted first, and therefore cut off communication in the network. Provision should be made for this, but Qu and Georgakopoulos leave it unmentioned.

2.2.2 Distributed

In Abbasi et al. [20]'s seminal paper DARA (a Distributed Actor Recovery Algorithm) is introduced. This develops a localised scheme to restore a network of mobile nodes when it has become partitioned due to a node failure. The distributed nature of the algorithm avoids involving every single node, only requiring the local nodes to respond. The algorithm selects a node to replace a failed node and coordinates the local nodes to relocate to rebuild the network. As nodes relocate they may cause other partitioning, but this is dealt with in the same way forming a chain reaction until the entire network is reconnected. The self-healing process requires no supervision or centralised control. The use of cascaded movement reduces the overall movement of nodes as instead of entire blocks moving to recover only a few nodes are needed. This contributes to the main goal of minimising the total distance moved by the network nodes. However, DARA produces particularly bad results when used with cyclic network topologies. When the node move-

ment is cascaded the entire cycle of nodes will be moved, when only one may need to be for the optimal solution. To combat this a variant is produced that sends flooding messages to check connectivity before moving. This increases message overhead, but can significantly reduce node movement which tends to be more power hungry.

PADRA (Partition Detection and Recovery Algorithm) [21] also aims to restore connectivity to a local area after a node has failed, whilst minimising the total distance travelled, and without external supervision or involvement. However unlike DARA it provides the functionality to detect the cut-vertices that are the cause of partitioning. PADRA determines potential cut vertices in advance by forming a connected dominating set of the whole network. The dominating set is also used for selection of recovery nodes as the dominating node moves to recover the network. Through this PADRA improves significantly upon DARA towards total travel distance of the optimal cascade model.

RIM (Recovery through Inward Motion) [22] improves upon DARA and PADRA which each need 2-hop knowledge of the network to only requiring 1-hop knowledge on each node. This significantly reduces the network overhead for maintaining the required knowledge of the network topology, however the simplicity means that the distance travelled by the nodes is greater in larger networks for RIM than for DARA or PADRA. Calculating and transmitting a lot of detailed information is often considered too much overhead for low-power, low-complexity systems, particularly if the network can change often or easily. Depending on the application for the WSN the communication overhead to maintain the topology data could be justified over the generally more efficient algorithm.

Another flaw in RIM is that it assumes and requires only one node failure at a time. Whilst this may be the case, failures in WSNs are most commonly battery depletion, which is likely to occur at similar times across the network, or random failure due to environmental conditions, which could happen at any time to one or multiple nodes.

SFRA (Simultaneous Failures Recovery Approach) [23] is designed specifically to deal with multiple simultaneous failures to combat the multiple failure problem in RIM. Network trees are built from the root node, with local cluster-head nodes, this introduces a fair amount of network overhead compared with RIM. The number of updates needed to send is reduced by waiting for all child node messages before propagating back up the tree.

Distributed approaches become much more efficient as the network

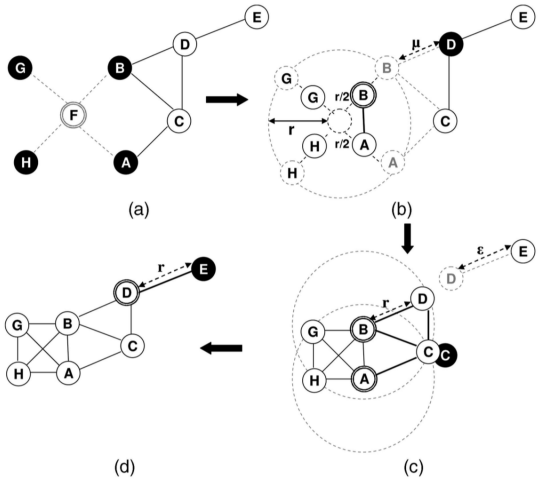
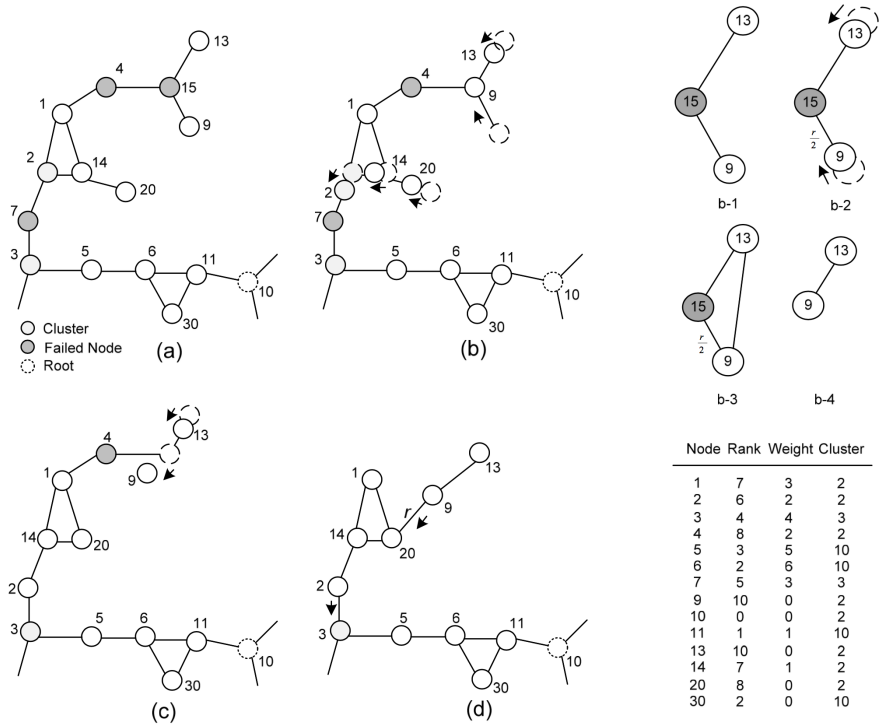


Figure 2.2: (a)–(d) An example for how RIM restoration process; each shaded node moves based on the position of its neighbours, denoted in double-lined circles. Reproduced from Younis and Lee [22].

scales because they are only concerned with the local nodes that are directly reachable. This allows networks built upon distributed algorithms to become much larger, and cover much wider areas as the overhead increase from additional nodes is minimal compared with the exponential increase in complexity from the centralised approaches.

A number of biologically-inspired approaches exist utilising animal and insect coordination and inspired methods of load balancing. Caliskanelli [24] suggests the application of bee pheromone based load balancing to node distribution. The main thesis applies pheromone signalling (PS) to node redundancy enabling and disabling nodes within a dense area to preserve their life when they are unneeded. By deciding upon a cluster head for an area all other nodes move to a dormant state but coverage is maintained. If an active cluster head fails then a new cluster head will be assigned. Each cluster head emits a pheromone signal that prevents other nodes from becoming the head. PS is then applied to robotic agents to increase service availability though guidance to specific locations. A pheromone signal denotes coverage of a particular area; in load balancing if an area does not have a strong enough signal a node will be activated to provide coverage and emit pheromone to show it. For robot



Node	Rank	Weight	Cluster
1	7	3	2
2	6	2	2
3	4	4	3
4	8	2	2
5	3	5	10
6	2	6	10
7	5	3	3
9	10	0	2
10	0	0	2
11	1	1	10
13	10	0	2
14	7	1	2
20	8	0	2
30	2	0	10

Figure 2.3: Detailed example to illustrate how SFRA algorithm restores connectivity after multiple nodes fail. Reproduced from Alfadhly et al. [23].

guidance the agents move towards areas that are lacking in pheromone signal as these will be areas in which nodes have depleted their battery or failed.

Another approach based upon the behaviour of ants in a colony has been introduced by Wang [25]. The system separates mobile nodes from the normal sensor nodes that are static. The network functions as normal on the static nodes. However if there is a need for relocation of a node, due to failure or any other cause, the mobile nodes can pick up the static nodes and travel with them to another location redeploying them there. This reduces the power and cost overheads of adding mobility to all of the nodes. The mobile nodes can carry a larger battery or travel to a charging station when not actively moving nodes.

A similar approach has been taken by Xu et al. [26] for mobile nodes with high power capacity to travel between static sensor nodes, however the mobile nodes here are for recharging the static nodes. In most cases the reason for node failure is depletion of the battery: if this is the case, being able to recharge the sensor nodes in place increases the theoretical length of network use indefinitely.

2.3 Metrics for evaluating self-healing WSNs

The most common metric used for evaluating recovery approaches is the distance travelled [27]. The total distance travelled is ideally minimised as excessive movement for recovery will have high power consumption. Another use of distance is average movement distance per node failure which allows comparison between multi-node and single-node recovery algorithms. Measurement of distance is trivial for simulated environments and therefore gives a good approximation of the power usage.

Another common measurement is the number of messages transmitted with regard to recovery. This shows the overhead produced by the recovery algorithm [20]. As movement and radio use have the highest power usage, measuring and minimising them will produce the system with the lowest power consumption which is the ultimate objective for most WSN implementers.

Measuring the number of nodes that have to be relocated shows the spread of power consumption across the network [22]. A larger spread means that the time until the first node fails will be increased as the load has been balanced across the nodes. This is obviously countered if nodes are moving unnecessarily. This metric can then be fed back

into the design to decide on the number of mobile nodes required in the network. If the number can be reduced then savings can be made.

As discussed previously, connectivity and coverage are both important to WSNs. The average node degree is a good metric for connectivity as it shows the average number of direct neighbours for each node. Coverage can be measured by the sum area of the sensing radii [28]. Higher degrees of connectivity allow for re-routing and improve network reliability without the addition of movement, however it may be desired to reduce the degree of connectivity and thereby the density of the network so that the fewer nodes are needed and costs are lowered.

Most distribution / relocation algorithms require knowledge of the node location. Investigation into this is outside the scope of this paper, however there are many methods of obtaining location from GPS receivers to self-localisation algorithms that are well summarised by Hu and Evans [29] and Mao, Fidan, and Anderson [30].

2.4 Review of evaluation techniques for WSNs

Developments in wireless sensor networks can only feasibly take place in simulated environments. The cost and time of full or even scaled deployments is far too high to be practical. Reliability is a key issue, and needs to be maximised prior to deployment as any difficulties will only be multiplied in a real-world setting and debugging these is a greatly increased challenge. The use of simulation for development and testing prior to deployment is therefore paramount. A large number of simulation environments exist: from generic network modellers to specific WSN frameworks, even simulators that are developed for a specific deployment. [31, 32]

Simulation environments must provide accurate models of real-world factors to be reliable and useful for development and testing. Often the more generic simulators provide less of these features, leaving the exercise to the system developer.

When testing self-healing networks node failures must be induced, to do this a reliability function to control the failure rate of devices is needed. Lee et al. [33] analyse the failures of nodes, which are predominantly due to energy depletion. Another common failure mode is the irregularity of radio transmissions which are often modelled, at worst, perfectly or at best very simply. Zhou et al. [34] investigate radio performance and integrity modelling and how to apply real-world data to

the simulated environment to achieve greater accuracy. Implementing and using these models becomes a very large challenge. As with any simulation environment more data can always be added to provide accuracy, but at the same time there is a risk of over-fitting a particular situation reducing the overall accuracy again.

VisualSense [35], part of the PtolemyII modelling software, provides a robust framework for building WSNs and creating modular system components. This modularity allows the complex modelling of energy consumption, radio transmission and reception, sensing, and data processing to be broken down into smaller components and connected together. Rosello et al. [36] show how this modular framework can be developed. In this way each component can be developed individually and iteratively improved to provide a better overall simulation model.

One challenge of discrete event simulation as provided in VisualSense is modelling wireless channel collisions. Each event in the system happens instantaneously, so collisions are not obvious unless the timing behaviour of transmissions is explicitly included in the model. As noted by Rosello et al. the different system states have different power draws and the energy will be depleted by the actions of the system.

2.5 Conclusions

The key failure modes seen in the literature are single-node, multi-node and transient failures. Single-node failures are covered extensively, but approaches to them have mostly been developed and improved to combat multi-node failures due to the commonality of environment changes and other factors affecting regions of nodes. DARA, PADRA and RIM target single node failures stating that they cannot provide recovery from multi-node failures but without showing their behaviour for that case. SFRA and the bio-inspired approaches are designed with multi-node failures in mind as their likelihood is relatively high. The nature and timing of failures is relatively unpredictable, power depletion aside. Any node or group of nodes could fail at any one time due to a change in the environment, particularly when the deployment is in an unstable or dangerous area. Handling of multi-node failures therefore is important for network recovery to be achievable. One untested area is that of transient failures and how the algorithms respond to 'failed' nodes that return to the network.

Almost all algorithm evaluation in the literature is done by simulation.

Whilst simulation can provide realistic data there are often assumptions in the model or perfect representations of, for example, radio transmission. The simulation frameworks are also often bespoke to the algorithm under test, making direct comparisons between the differing approaches tricky. Real world environments are a lot more difficult to work in, and slow down development, but without providing any comparison from the simulations to real world deployments it is hard to evaluate the robustness of the simulation platforms. The impact of other errors in the system are also not shown. Whilst this is treated as out-of-scope for the research being performed, in practice errors in systems do happen. If the goal of each of these algorithms is to provide a network architecture that is able to recover from failures, then testing with failures should take place. Many approaches require accurate location data for each node, which leads to the question: without this, or with a degraded localisation, how would they perform and would they be able to achieve anything?

3 Problem Analysis

This chapter specifies what the project aims to achieve, and the requirements and objectives to be reached for the project to be considered a success.

3.1 Problem Description

This project aims to practically and critically evaluate approaches to self-healing in wireless sensor networks. A reference platform for implementing and testing self-healing algorithms will be designed and implemented; this platform will allow comparisons to be made between different approaches within a single environment thereby providing a fair contrast between the algorithms themselves.

An example algorithm will be presented and critical analysis of its design and the evaluation presented by its designers will be conducted. This will provide a structure for obtaining and presenting results from the simulation platform for further algorithm comparison. The scenarios presented will be based upon those found to be common in the literature.

A simulation rather than hardware platform is chosen for a number of reasons. As presented in section 2.4: hardware deployments are difficult and costly and extracting the data required for the evaluation metrics becomes a challenge within itself; particularly without affecting the performance of the rest of the system. With simulation all metrics can be gathered without side effects, and with considered design it can accurately emulate real-world environments. Finally the size of deployment is a large factor, simulation can relatively easily run large networks that would not be feasible to create during the evaluation stage of system development. The evaluator may not have access to much if any hardware, whereas anyone can run a simulation.

Experiments on hardware did take place in the preliminary stages of investigation, however the above issues were encountered and the time required to implement a general enough platform for comparison with

the simulation platform was too great to be feasible within this project.

The scope of the platform is to accurately simulate communication and power use of the nodes as these are the primary factors affecting longevity of the network and recovery from failure. It is assumed that the nodes have an awareness of their own location, and that could be provided in a number of different ways that are outside the scope of this project.

The project limits the choice of algorithms to those of a distributed nature as the architecture of centralised approaches are very different due to the problems each approach aims to solve. To compare the results of the scenarios produced from the different architectures would be an interesting topic, but is too large for this project.

The platform itself needs to be structured in such a way that it can be easily modified and reused, with the ability to plug-in a variety of self-healing algorithms without need for modification of the overall system. Because of this it should provide interfaces to all the data of a node that can be made available. The structure of the platform should be such that individual parts can be re-implemented and swapped out to provide a more accurate or a faster running model depending on the needs of the system user.

The implementation should be based upon real hardware, and systems found in the literature, however adaptations may be made to achieve the aims of this project.

3.2 Objectives

This section presents the key and optional objectives for the project. The key objectives should all be achieved to result in a usable research output from the project. And the optional objectives extend the research and provide additional confidence in the platform for use with other algorithms and in other scenarios.

3.2.1 Key Objectives

The project aims to:

1. develop a simulation platform for wireless sensor networks, providing wireless communication, power management / monitoring, and movement control for the nodes.

2. implement a self-healing algorithm within the platform and run it in a failure scenario.
3. gather metrics from the scenario simulation.
4. use the metrics to evaluate both the platform and the algorithm and to compare the algorithm with its original results.

3.2.2 Optional Objectives

Additionally the project will attempt to:

1. run further scenarios such as multi-node failure and unreliable data channels.
2. implement further algorithms and compare between those run on the platform.

4 Design and Implementation

In this chapter the system design and architecture are described including the decisions made for their development. Firstly the platform architecture is developed drawing upon practices seen in the literature review, then the implementation of self-healing algorithms within the platform is described. Finally scenarios for evaluation are explained to show how the algorithms can be analysed and critiqued.

4.1 Platform Architecture

To build the simulation platform the PtolemyII VisualSense framework [35] was used as it provides a full simulation package with the flexibility and detail to simulate anything that can be modelled. The actor based simulation enables fast prototyping of architectures that can then be developed into optimised Java components. I also had prior experience with PtolemyII so it required less familiarisation time.

Many other simulation platforms are available with different advantages and disadvantages as seen in section 2.4. TOSSIM [37] from TinyOS by Levis et al. [38] was considered during initial investigation when a simultaneous hardware and simulation platform was being considered as this would enable a single code base for development. However the difficulties of hardware development mentioned previously, alongside the lack of existing structures within TOSSIM resulted in it being rejected. TOSSIM is designed as a low level simulation for TinyOS's execution "rather than simulating the real world", and real world simulation is one of the goals of this project.

PtolemyII provides a large number of existing components and an easy to use visualisation so that the state and topology of the network can be observed throughout simulation, this has led to interesting discoveries which will be discussed later.

The architecture itself is based upon Rosello et al. [36] using finite state machines (FSMs) for each key component: Communication, Motion, Power, Processing, and Sensing. This modular approach "allows

dividing and encapsulating the functionalities included in the node" which is very important for constructing a general use platform. It also allows individual modules to be worked on and improved individually, they can be changed and swapped out without affecting the rest of the system. VisualSense FSMs can be constructed in layers, where each state may have a state machine or system model of its own. This layering allows highly complex state machines to be implemented much more simply and for different functionality to easily be modelled within different states. This architecture design is based upon real hardware which, whilst different from the hardware within this project, shows the same goal of an accurate hardware simulation model.

Using the actor model entirely can lead to inefficient systems as it constructs a very large number of objects that must intercommunicate to perform even the simplest tasks. PtolemyII alleviates this through the ability to create your own actors purely in Java code, this allows more efficient processing within single components. The initial design of most components within the platform will be using the VisualSense actors, but this can then be developed into single Java actors providing the desired modularity is retained.

The platform with 5 nodes is shown in figure 4.1 which has been annotated to show the multi-hop connection topology from parent to child nodes. Node 1, the root node, is the sink node where data would be exported from the network to the external computer logging data. All data messages are passed back to here via multi-hop routing. It can be clearly seen here how node failure would cause partitioning, if node 2 was to fail then nodes 4 and 5 would no longer be connected to the root node as can be seen by their communication range radii. This topology was chosen as it provides the shortest path to the root from any node.

Figure 4.2 is the internal structure of a node showing its parameters and the interconnections between modules. The modules themselves are modal actors with internal finite state machines as described above. Within these further composites are formed by connecting other actors together, each module is expanded upon in the following subsections. In the figures of the finite state machines bold outlined states are the initial state, green states have internal refinements, and double outlined states are final. The guard conditions shown on the transitions control state changes. Transitions to the dead and failed states are immediate, not waiting for any internal computation to complete.

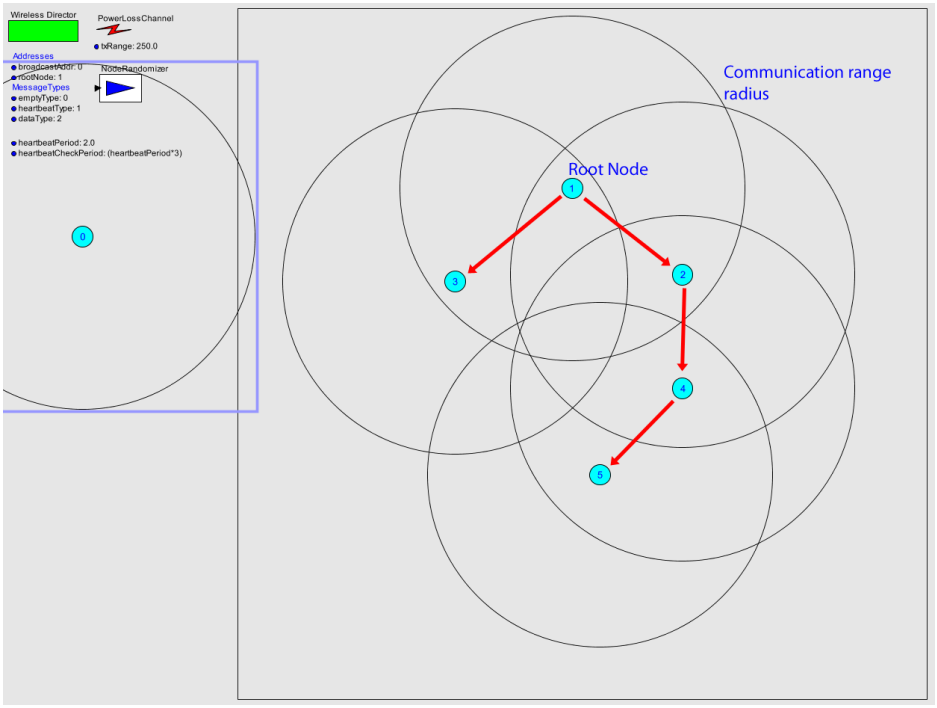


Figure 4.1: Overview of the platform annotated with the communication topology between nodes. Node 0 is the class actor for each of the instances and can be ignored.

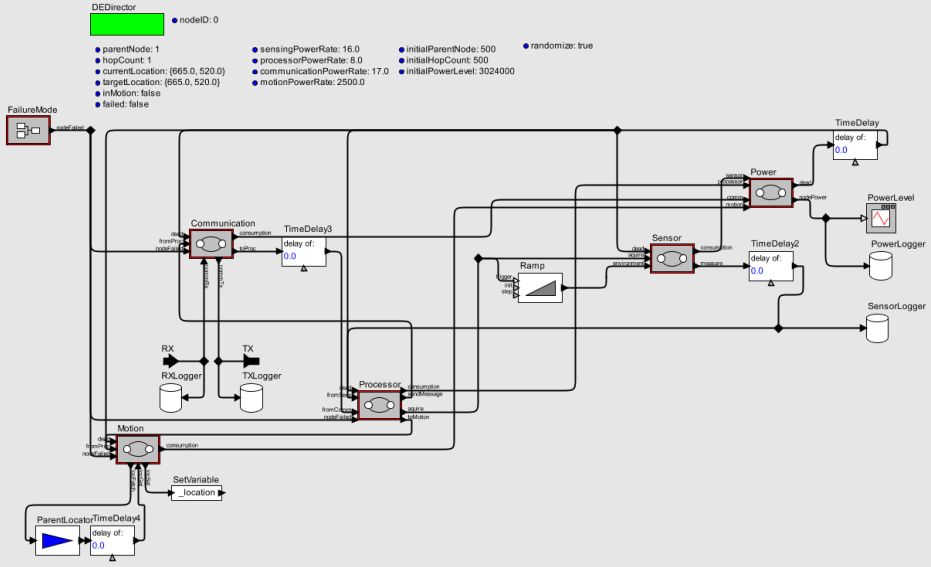


Figure 4.2: The internals of an individual node showing links between the modules.

4.1.1 Communication

The communication module interfaces with the outside world. It is linked directly to the wireless outer ports of the node. It controls what messages are received by the processor and assembles messages from the processor into transmissions into the network. The initial implementation of this module is relatively simplistic, rejecting messages that are not sent to it and sending messages either as broadcast or to the parent node. Further development should allow messages to be sent to any destination and more in-depth modelling of reception and transmission patterns for wireless antennae and communication. However, data messages need only reach the root node for logging in most applications so the decision was made to implement this simple routing structure.

The power consumption model is also ideal and therefore unrealistic, as it should have to turn the radio on to send and receive messages, and off to sleep and conserve power. Currently the communication module is able to receive messages at any time, and outputs the standard consumption rate on message transmission and reception. The state machine should be used to signify transmission, reception and sleep state with the relevant power consumption at a constant level over time

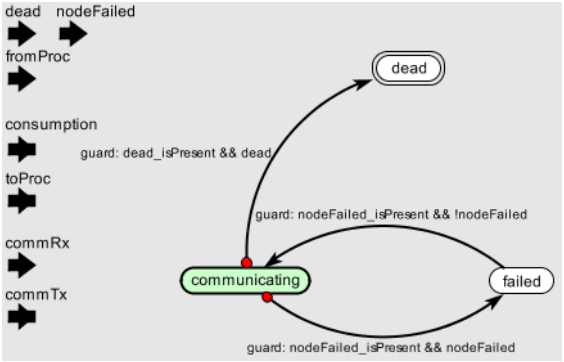


Figure 4.3: Finite State Machine of the Communication Module

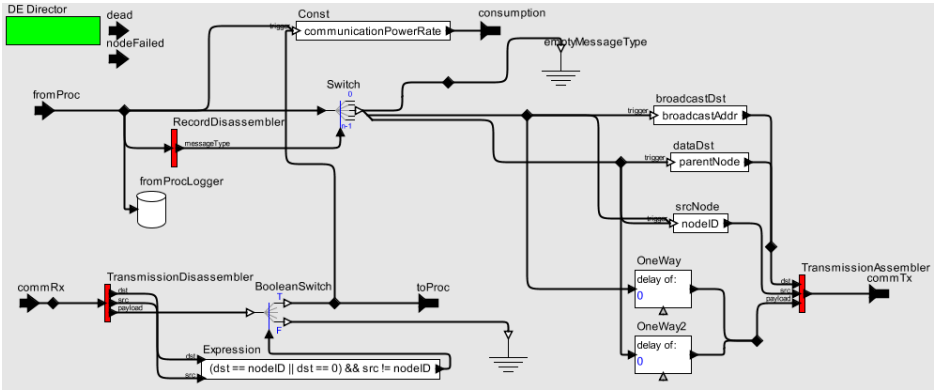


Figure 4.4: Internals of the Communication Module

within each state. A common schedule for this is to activate once per minute with a 50ms activation time, and listening / transmission time depending upon the message size required. Modelling communication in this way would drastically change the power consumption simulation to have a much greater realism.

Further, the communication channel model is overly simplistic. VisualSense, without additional implementation, does not model transmission collisions as all inter-actor communication takes place instantaneously. To model this the transmission time would need to be calculated from the message size and any transmissions that took place simultaneously would need to be dropped by the receiver due to the channel conflict. Communication range is modelled in the channel, but conflicts, power dissipation, and external interference are not.

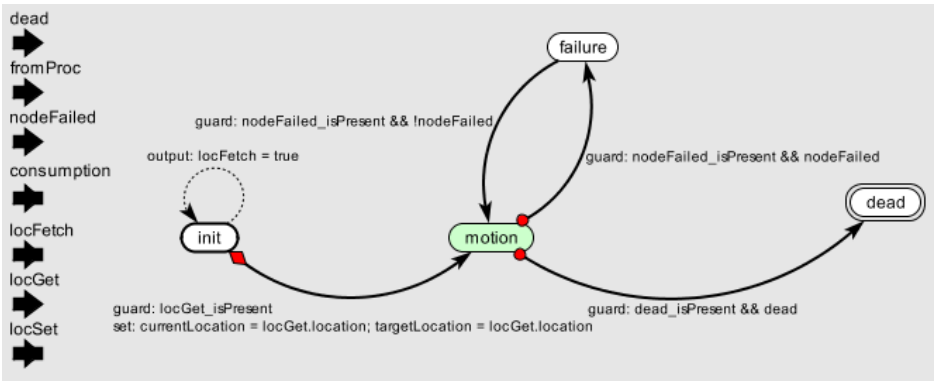


Figure 4.5: Finite State Machine of the Motion Module

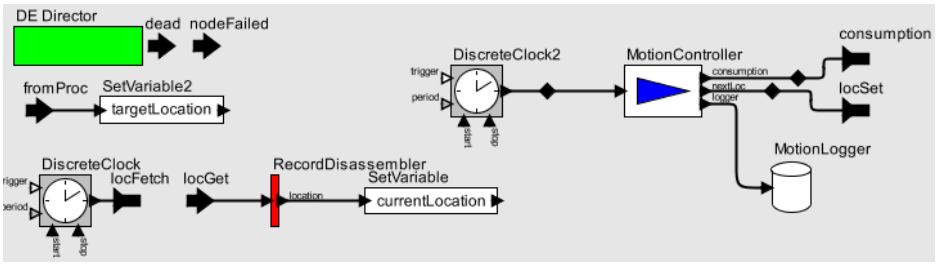


Figure 4.6: Internals of the Motion Module

4.1.2 Motion

The motion module provides localisation and movement for the node. As localisation is outside of the scope of this project it is simply assumed that the node is aware of its location by some means that has been abstracted away from. VisualSense provides a location parameter for each actor that can be read and set, so this is used by the module. The current location is periodically updated and checked against a target location which can be set. When the target is set to a location that the node is not currently at the module causes the node to move towards the target over time; this is done by incrementing/decrementing the location parameter of the node towards the target value. By changing location by small amounts over time the speed of the node motion can be controlled by the module instead of simply jumping to the target which is unrealistic.

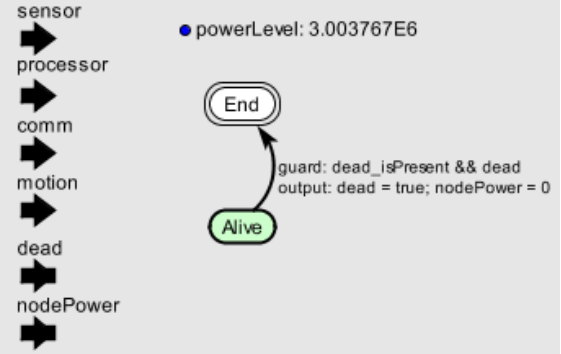


Figure 4.7: Finite State Machine of the Power Module

Node Module	Energy Consumption Rate (mA)
Communication	17.0
Sensing	16.0
Processing data	8.0
Motion	2500

Table 4.1: Energy consumption rates of the various node modules

4.1.3 Power

The power module takes an input from each of the other modules as they work and reduces its stored power level by their consumption over time until the energy level reaches 0. At this point the battery is depleted and it sends out a 'Dead' signal. All of the modules have alive states and a single dead state that is the final state. At any point if a 'Dead' signal is received the modules should transition immediately to the dead state which ceases all computation of that module. The power level and consumption rates of each component are customisable on each node. They are set in my tests using the Memsic IRIS node data-sheet [39]. These consumption rates are shown in table 4.1.

4.1.4 Processing

The processing module is the heart of the wireless sensor node. It controls and communicates with every other module. Initially the architecture for this component was designed using actors within VisualSense, this allowed fast prototyping and debugging, however as the full be-

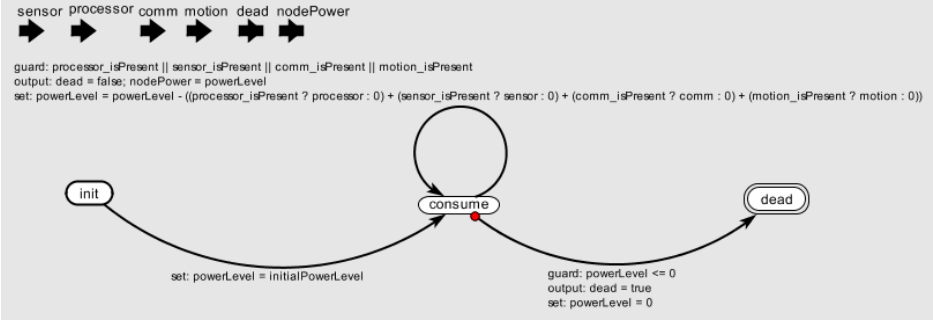


Figure 4.8: Internals of the Power Module

haviour of the model was developed the complexity of the system increased and actor implementation became unwieldy. Larger parts of connected actors were needing to be duplicated to accommodate all the required functionality, so the actor development was stopped and the processor was moved to Java. This actor model however provided the basis for the architecture of the processor code, and showed what elements were required for the platform and what needed to be implemented by individual algorithms.

Moving to Java allowed much more complex behaviour to be defined simply and multiple actions / function reuse could take place without the problems of looping connections that introduce the need for 'TimeDelay: 0' actors to act as one way connections. The partially complete actor model can be seen in Figure 4.9.

As the processor controls the behaviour of the nodes it would need to be re-implemented for each algorithm tested. To minimise this the processor class is designed to be extended for each control algorithm, providing all the other necessary components in the base class for reuse. These can always be overridden as needed according to Java's inheritance model.

The processor has three periodic tasks that are controlled by the clocks in the model. Firstly to create heartbeat messages: these are used to communicate the state of nodes throughout the network. The heartbeat is broadcast to all nodes within the communication range — the neighbours of the node. The node information is used to construct a tree network architecture with the parent node being the one with the shortest hop count to the root node. Secondly to get data from the sensing module and produce application messages to send back to the root. And

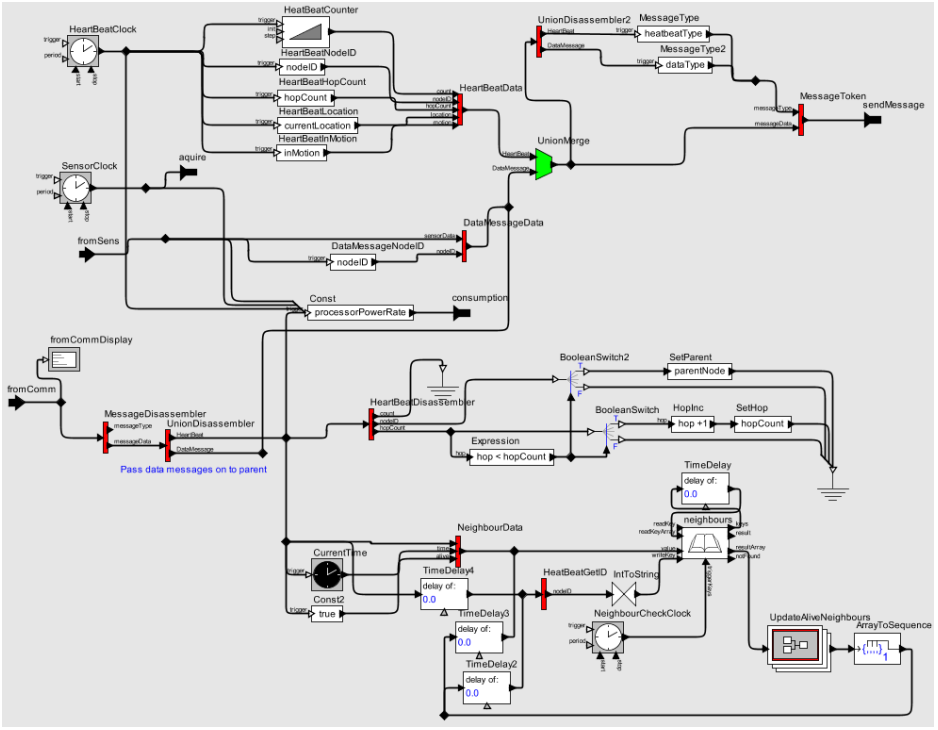


Figure 4.9: The partially complete actor model of the processor before re-implementation in Java.

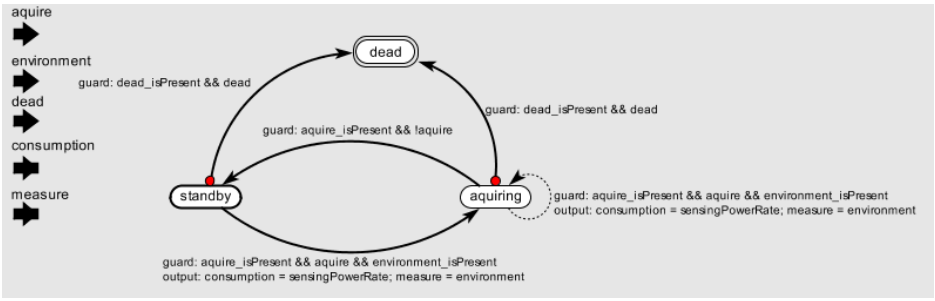


Figure 4.10: Finite State Machine of the Sensor Module

thirdly to check the state of the known neighbours.

When messages are received from other nodes application data is forwarded on to the parent, eventually reaching the root, and heartbeat messages are stored in a neighbour data store. As the heartbeat period across the network is known the time between heartbeats from neighbours can be used to check if they are still alive — the reasoning behind the naming of the heartbeat protocol. This time-out is calculated to be larger than the heartbeat period to reduce the likelihood of transient failures. When the neighbour check is carried out the data store of neighbours is updated with the alive state for each neighbour. The processor is then able to read this information to trigger recovery.

4.1.5 Sensing

The sensing module is the simplest: it outputs an environment measurement when triggered, this models the behaviour of reading from a sensor board and the power consumption required to do so and gives data to send in the periodic application messages.

4.2 Self-Healing Algorithms

The self-healing algorithms under test are implemented within the processing module. The base Java class for processing can be extended for the specific functionality required by each algorithm.

Wireless network recovery is initiated when a node discovers a neighbouring node is no longer communicating. This could be for a variety of reasons as discussed in Chapter 2. How the recovery is dealt with depends upon the particular algorithm used, and it is the algorithm's

responsibility to recognise the reason for non-communication and react accordingly.

To implement a recovery algorithm the periodically called 'neighbour check' function is overridden.

4.2.1 Restoring connectivity through Inward Motion (RIM)

The RIM algorithm was developed by Younis and Lee [22]. It was chosen for implementation because of its simplicity and performance, the algorithm is clearly presented and seemingly performs well in low density networks with an even distribution of power consumption. This seemed worthwhile to put to the test, particularly as RIM has become a common comparison algorithm since its development.

Using the pseudocode the algorithm was implemented to run when the neighbour check found a non-communicating node. The algorithm then reacts based upon whether the node seems to have failed or just moved away. Various geometric functions are required to compute the new location depending on how many other nodes are moving.

The implementation of RIM (algorithm 4.1) can be seen in listing 4.2 which is run periodically for each neighbour of a node. My implementation differs slightly as it does not implement the "do-not-move" behaviour. Therefore if a leaf node fails, the algorithm will send a node to recover that will not return. However this behaviour should not affect the success of recovery due to the cascaded motion of the other nodes, and actually improves the sustained coverage of the network. It is unclear whether the algorithm continues to run: detecting failed nodes, whilst in motion. In my implementation all nodes are periodically checked and if they are detected as failed then the rest of the algorithm is executed. This can cause one recovery action to override another depending on the *I_Already_Reconnected* guard. Whether or not *I_Already_Reconnected* should be reset upon post-recovery stabilisation is also unclear, but it is assumed not due to the design being for single-node recovery only. It would be advantageous to multi-node recovery if it could be reset as it allows cascaded motion to re-occur.

These differences should not affect the evaluation of RIM as the overall recovery process is unchanged. And the assumptions made over unspecified behaviour of the system outside of this algorithm are an integral part of an independent implementation and evaluation. The algorithm documentation should provide everything that is needed for it to be implemented and should deal appropriately with any assump-

Algorithm 4.1 Pseudocode for the RIM algorithm

```
1: if a sensor node J detects a failure of its neighbour F then
2:     NOTIFY_MOVEMENT(J)
3:     J moves towards F until becoming a distance r/2 away
4:     if J is unable to reach any other 1-hop neighbour of F then
5:         J returns to its original location (F is a boundary node) and
   sends a do-not-move message to its neighbours.
6:     end if
7: else if J receives (a) notification message(s) from F then
8:     if I_Already_Reconnected then
9:         Done;
10:    end if
11:    NewPosition ← COMPUTE_NEWPOSITION(J)
12:    if NewPosition ≠ CurrentPosition then
13:        NOTIFY_MOVEMENT(J)
14:        J moves to NewPosition
15:    end if
16: else if J receives a "do-not-move" message from F then
17:    J does not move towards F (no cascade movement)
18: end if
19: I_Already_Reconnected ← TRUE
20: function COMPUTE_NEWPOSITION(J)
21:    NUM_PriorNBR ← Number of notification messages that J has
   received with the lowest rank
22:    if J stays connected with all PriorNBR with least rank then
23:        return CurrentPosition
24:    end if
25:    Location_Senders[] ← New location(s) of (a) sender(s) from
   which J have received (a) notification message(s)
26:    if NUM_PriorNBR = 1 then
27:        return a point r unit(s) away from Location_Sender[0] on the
   direct path to Location_Sender[0]
28:    else
29:        Define a circle of radius r around each of Location_Sender[]
30:        if NUM_PriorNBR = 2 then
31:            return the closest point between two intersection points
32:        else if NUM_PriorNBR > 3 then
33:            return the closest point among intersection points which
   is located inside all other circles
34:        end if
35:    end if
36: end function
37: function NOTIFY_MOVEMENT(J)
38:    Send a message with a rank value increased by 1 to inform all
   neighbours of J about its motion and its new position.
39: end function
```

```
60  if ((currentTime >= nodeUpdateTime + heartbeatCheckPeriod)) {
61      // Higher ranked node has lost connectivity.
62
63      if (nodeInMotion) {
64          // Node has moved away
65          if (alreadyMoved) {
66              return;
67          }
68
69          ArrayToken newPosition = computeNewPosition();
70          ArrayToken currentPosition = (ArrayToken) getVariable("
                currentLocation");
71
72          try {
73              if (!currentPosition.isCloseTo(newPosition, 1.0).
                    booleanValue()) {
74                  toMotion.send(0, newPosition);
75              }
76          } catch (IllegalActionException e) {
77              throw new RuntimeException(e);
78          }
79      } else {
80          // Node has probably died
81          try {
82              neighbours.replace(node, setNeighbourLive(nodeRecord,
                        false));
83              toMotion.send(0, computePositionFromDeadNode(
                        nodeLocation));
84          } catch (IllegalActionException e) {
85              throw new RuntimeException(e);
86          }
87      }
88      alreadyMoved = true;
89  }
```

Listing 4.2: Java implementation of RIM

tions that do need to be made; these should not affect the performance of the algorithm. In this case I would have expected specification over the continued detection of failures during recovery as it does change the behaviour.

4.3 Simulation Scenarios

To test the algorithms and the platform a variety of scenarios are constructed for the network to run in. These scenarios, their goals, and the reasoning for their use are presented in this section.

4.3.1 Single Node Failure

The single node failure is a common and simple scenario to perform. Whether due to energy depletion, hardware failure, obstruction to transmission, or a software fault, an individual node could fail at any time for many reasons. As seen in the literature, this was the first problem scenario self-healing algorithms were designed to solve, therefore it is the base test as to whether or not they are functional and can be classed as self-healing.

The goal of this scenario is to retain / restore connectivity across the network after a single node has failed. For this scenario the simulation was set up with a number of nodes randomly distributed within a 1000x1000 unit area. Once running a node is randomly selected to fail, where it changes to a non-responsive state no longer sending or receiving messages. RIM is designed with this specific scenario in mind and results are presented with varying communication ranges and numbers of deployed nodes. However when re-creating these experiments it was discovered that the random placement combined with lower communication range or a lower deployment number results in a highly partitioned network: a high number of non-connected nodes / clusters that cannot route to the root node.

The minimum range deemed suitable for this area for the number of deployed nodes is presented in table 4.2. As can be seen, these values are much larger parameters than most of the simulations run by Younis and Lee. It would seem that their random deployments are either non-random enough to create a connected network, or their results for low density networks are incredibly sparse. To test this scenario on my platform the values from table 4.2 were used as the network parameters.

Deployed nodes	Minimum range	Mean connected nodes	Std. Deviation
25	250	24	1.54
50	200	47.8	3.34
100	150	99	1
200	100	196.8	3.4

Table 4.2: Minimum ranges for number of randomly deployed nodes in a 1000x1000 unit area.

4.3.2 Multi Node Failure

A multi node failure is as likely to occur as a single failure, if not more so. Yet recovery from such a failure is much more complex. As we have seen, until more recent developments in self-healing, very few algorithms were designed to be able to recover from a multi node failure, specifically discounting it as unlikely or too complex to be within scope for their development. The SFRA algorithm completely restructures the topology of the network to combat this issue, leaving behind many previous assumptions and providing a relatively simple solution to the complex problem.

There are a few variations of this scenario. Firstly and most simply is sequential failing nodes, this could be considered as repeated single node failures within a single simulation run. This should not present any particularly new challenges to systems that can recover from single node failures, however the assumptions made in their development may cause problems for recovery.

Secondly, simultaneous node failure in distinct regions of the network. Again this should present no further challenge than the single node failure if the nodes are separated by a large enough distance. The challenge appears when they are close enough that a node is required in the recovery for both failed nodes. This challenge was beyond algorithms such as DARA and RIM, and simply not answered for them.

Thirdly, regional failure. This is where an entire region of nodes fails simultaneously and is the common consequence from an environmental change, for example, a landslide occurring due to an earthquake that destroys the nodes monitoring that area. In practice it may be unwise to attempt recovery into a lost region if the destructive event is repeated, or has left a region in a state where deployment is not possible, such as lava flow from an erupted volcano.

Again the goal of these scenarios is to recover to a fully connected state with adequate coverage of the area of interest.

4.3.3 Further Scenarios

There are several other scenarios that could be developed for the platform. One particularly useful scenario is using an unreliable communication channel. Doing so would produce a more realistic transmission model and show the resilience to real-world problems in communication for the algorithm under test. All kinds of outside actors can affect radio transmission, from electrical noise generated by nearby equipment to third-party radio transmissions; interference from other sources or even from within the network itself is a major issue in all forms of wireless communication. As discussed in the literature, the radio patterns for transmission and reception vary greatly from the ideal fixed radius circle used in the previous scenarios [34]. Modelling this degradation would be another challenge, but a worthwhile development to improve the simulation platform.

The wireless channel model in PtolemyII provides power loss modelling and probabilistic transmission loss alongside the limited range properties which have been used. Within VisualSense collision detection is possible by providing a message size / transmission time per transmission. Using these to extend the communications module of the nodes could greatly increase the resemblance to real world environments and allow testing in harsh settings during the development of a self-healing algorithm.

Some of the other scenarios that could be implemented are: transient failures, where nodes fail for a period of time then recover. Movement of nodes from outside actors, this could be caused by a landslip which does not cause the nodes to fail but moves them away from their original location. Failure of other system components, for example location data is reported incorrectly.

5 Evaluation

In this chapter the scenarios developed in section 4.3 are evaluated using a variety of metrics to give a quality assessment of the algorithms and to compare them against the original results presented in the literature. This assessment then provides an evaluation of the platform itself and its performance and accuracy can be discussed.

Each scenario is run on each of the different sized networks and repeated to produce an average response for that network set-up. The simulation was run for a fixed amount of time with a random deployment of nodes and random scenario events; this ensured a fair representation between runs. Each run was repeated to give a representative sample of the scenario and check for consistency in the simulation. All simulations were run with the parameters shown in table 5.1, energy consumption rates were shown previously in table 4.1. The failed nodes are chosen from the centre of the deployment region, this makes it unlikely to be a leaf node and gives the best graphical view of the recovery process.

5.1 Evaluation Metrics

In the literature review a number of metrics were discussed; here the chosen metrics for the system are described and the reasoning for their choice is explained.

As previously shown, the predominant measure of performance for any wireless sensor network is the energy consumption. One of the

Simulation Parameter	Value	Units
Heartbeat Period	1	s
Data Message Period	20	s
Simulation Run Time	1000	s
Travel Speed	1	units/s

Table 5.1: Simulation Parameters

main features of wireless sensor networks is their low power demand, so any features introduced that increase this unnecessarily are rejected in development. As with any architecture design the benefits must be weighed against the cost, and energy consumption is a major cost. High energy consumption reduces the lifetime of the network and the amount of data that is able to be collected, and increases the monetary cost of developing and running the network through larger capacity batteries or other changes that must be made due to nodes requiring more power.

Therefore energy consumption is a major concern for developers of self-healing wireless sensor networks. It may be the case that not implementing self-healing would produce a cheaper and longer lasting network, even if nodes need re-deploying manually, due to the cost of the additional power use. Of course the aim in developing such a system is to increase the lifetime of the network and reduce the need for costly re-deployments, but this can only be achieved by keeping the energy consumption down, and potentially reducing it below that of a non-healing network. Both the total consumption and the average consumption can be measured to show if individual nodes are having a particularly high draw, which will often be the case with nodes that have moved compared to those which have not.

The main energy draw in the system is moving the node and any time taken in doing so will likely prevent sensor reading from taking place or at the least data will be lost whilst a node is out of range until it reconnects. Therefore the distance travelled is a useful metric in observing why the energy consumption might be so high. The total distance travelled shows this across the whole network, and the mean distance travelled shows the spread of node travelling around the environment. The goal of this metric is to reduce the amount of motion to only what is required, removing any unnecessary movement. Involving a large number of nodes in recovery (which can be seen using the number of nodes that have moved as a metric) will have different consequences to having few travelling longer distances. Again these differences must be weighed up by system designers and a balance provided between them to achieve an optimal solution.

The degree of connectivity can be measured by the number of neighbours each node is connected to; the mean of these values provides an overall measure for the scenario. This provides a guideline for the density of the network, which affects how many nodes are likely to move when a failure occurs. However a high connectivity should allow rerouting of the network without the need for movement. A good ap-

proach to self-healing should combine both routing and movement for the optimal result.

The platform architecture is designed to deliver application messages to the root node. By measuring the number of these transmitted into the network and those received by the root, a percentage of successful message transmissions can be made. This gives a value of robustness to the recovery. Ideally all messages should reach the root, and if nodes fail the network should be restored fully so there is no partitioning from the root. However this can be affected by all kinds of other factors: the unreliable channels of wireless transmission will automatically reduce this, but despite the external factor this metric can still provide useful information about the success of self-healing.

Finally the mean hop count from the root node gives an idea of the topology of the network: if this is high then the topology will be in its worst case of a long individual line with only one route to the root node. With random deployment this is unlikely to occur.

5.2 Single Node Failure

Running the single node failure scenario with 25 nodes produces relatively consistent results (shown in figure 5.1). RIM is able to restore connectivity and reach a stable state in the network within a minute (of simulated time).

Even following the guideline parameters discussed previously run 1 had a partitioned network from the random placement and hence has a low percentage of data messages received back at the root node. The travel distance and number of nodes moved stay consistent between runs, as these are dependent upon the density of the network which can be seen by comparing to the results of 50 node networks. Comparing against the results produced by Younis and Lee shows that their network must be much more densely packed, as suggested previously, due to the average distance moved being considerably lower. However the trends in the data are consistent with the original results, particularly considering the increase in average travel with high communication ranges.

Energy use is not presented in the original results so cannot be compared. By using the average power usage during the simulation, the lifetime of the network can be estimated to be around 7 hours with a 2000mAh power source. This is incredibly low, however, the actual average consumption rate is likely to be much lower than the average

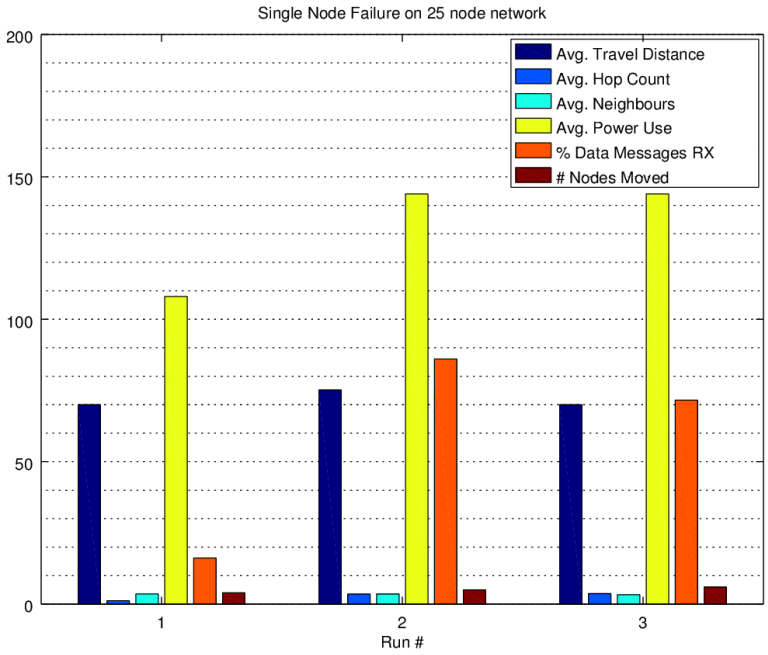


Figure 5.1: Results from three runs of a 25 node network with RIM. Travel Distance is units within the simulation. Power Use is mA. Other values are quantities.

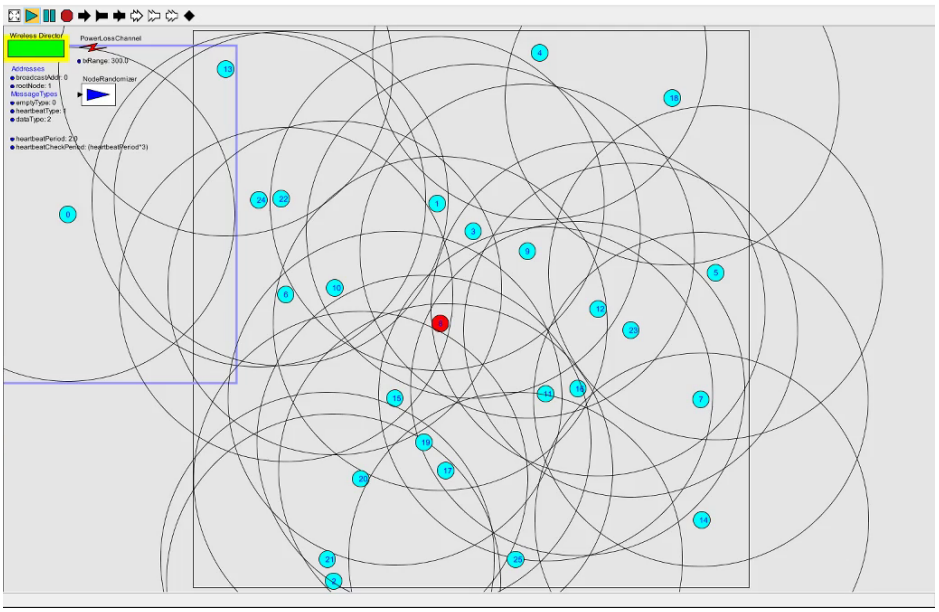


Figure 5.2: 25 node network with failed node 8 (red) prior to recovery.

presented in this short simulation as the majority of the simulation involved movement. This result shows a known flaw in the simulation platform, and a bad assumption that has been made in development. The communication module does not sleep, and takes a pessimistic 1s use of energy for each transmission and reception. This is gravely exaggerated as it is common for radio use to be in cycles of 250ms active per minute: this arrangement would drain a total of 1.8mAh per day for radio use, compared to the 408mAh per day currently simulated.

The average neighbour count is 4 which is a fairly high level of connectivity throughout, measuring this over time during the simulation would allow comparison of connectivity before and after recovery. The nature of RIM is to increase connectivity in its recovery process as it brings nodes closer together. The average hop count is also 4 which correlates with the neighbour count given the network size of 25.

Graphical visualisation of the simulation shows some interesting behaviour. Certain arrangements of nodes when a failure occurs can lead to oscillations in location rapidly depleting the energy as the node moves back and forth. This does not seem specific to the implementation of RIM presented here, but is inherent to the algorithm (algorithm 4.1). This occurs when a node (A) becomes just out of range from two other

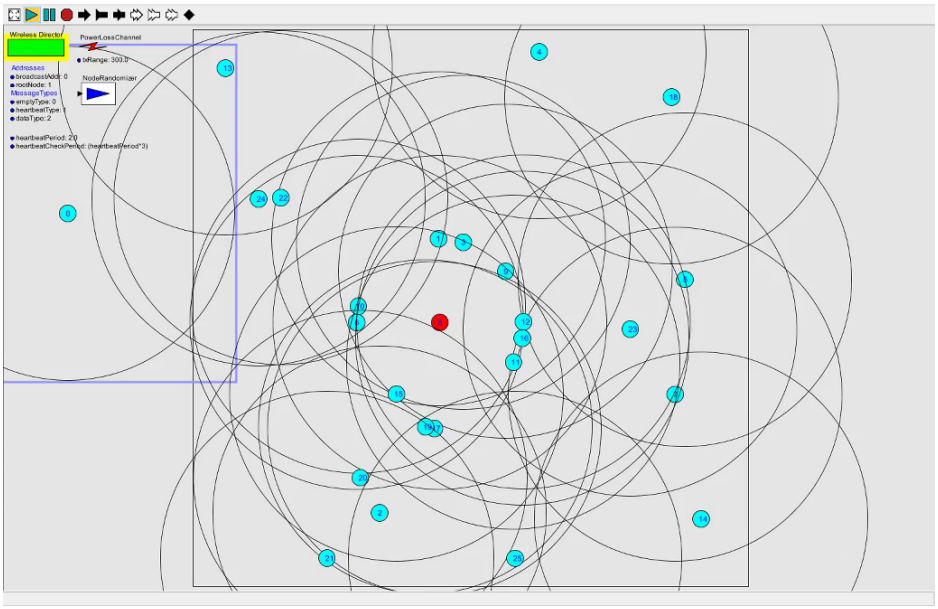
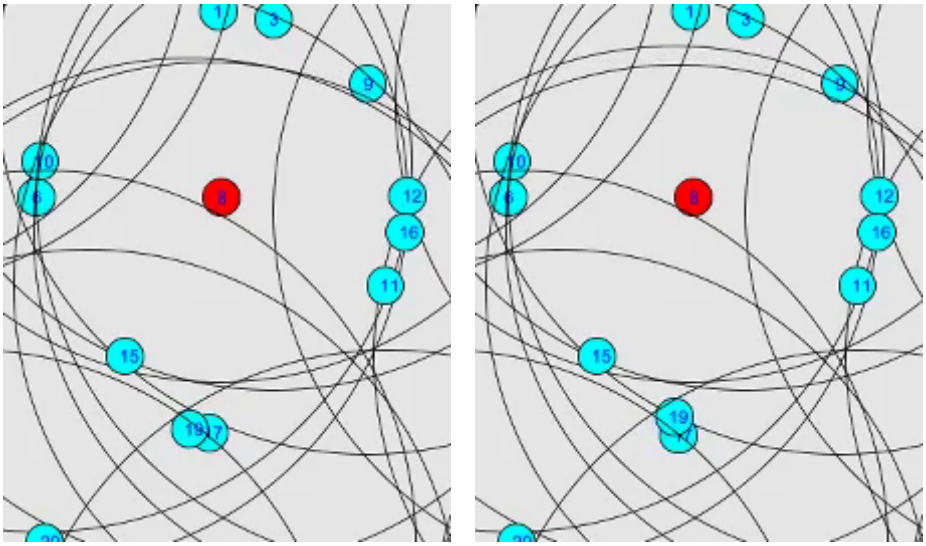


Figure 5.3: 25 node network with failed node 8 (red) post recovery.

nodes (*B* and *C*) that have previously moved and set *I_Already_Reconnected* true. When the *A* notifies its intention to move *B* and *C* do not respond in chained motion because of lines 8–10 in algorithm 4.1, therefore as *A* moves away and loses connection with *B* or *C* it detects them as failed and moves back towards them. When *B* and *C* are either side of *A* this will repeat. As one node comes into range of *A* the other will have become out of range and the cycle continues. Figure 5.4 shows the positions of two nodes (17 and 19) oscillating between nodes 21 and 9 from figure 5.3.

This behaviour was occasionally seen in 25 node networks, but was often the case in 50 and 100 node networks, this can be seen in run 3 on figure 5.5 where the travel distance is dramatically increased. This run did not actually converge but was still oscillating when the simulation ended. The number of nodes moved has increased due to the increase in the average number of neighbours.

Despite having raised some concerns over the original process and evaluation presented for RIM, the results from my implementation seem consistent with theirs, which would suggest the simulation platform is able to provide accurate data. The data itself may show problems in the algorithm, but the similarity shows accuracy in the implementation and



(a) Node oscillation stage 1. (b) Node oscillation stage 2.

Figure 5.4: Node 17 and 19 oscillating post recovery

the platform itself.

The 100 node network shows a similar trend, and all runs finished with nodes still oscillating their location. This seems to be a fairly major flaw in the algorithm as it is hugely wasteful of energy, and I suspect that the behaviour would be exhibited on any network with high density or a high node count. However discovering the flaw shows the strength of visual simulation, as it is obvious to any observer, whereas it would be unlikely to be discovered by purely analysing log file output unless it was specifically looked for. It is likely that the statistics presented on RIM would have been automatically generated from the logs without further analysis for oscillations or for network connectivity as previously mentioned.

The platform itself performs well for networks with lower node counts, however it does slow down significantly for larger networks, table 5.2 shows the comparison of wall-clock run times for each network size tested, runtime increases linearly at just less than x2 for a x2 increase in nodes. At this rate simulating 7 days of lifetime for a 100 node network would take 4.4 days of run time. This is not an infeasible simulation, but it would be desirable to be faster. This however would be the case with more realistic simulation parameters. The current parameters could feas-

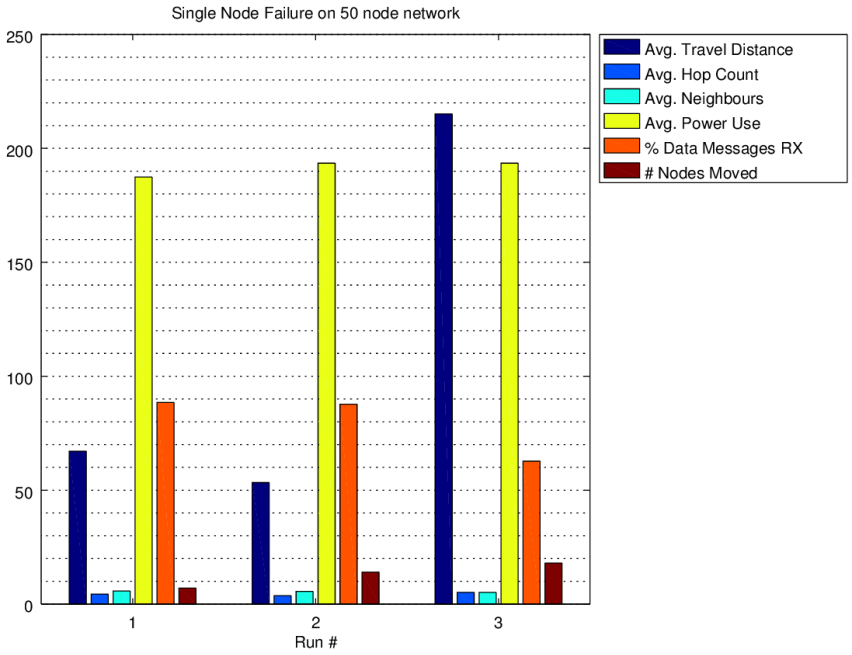


Figure 5.5: Results from three runs of a 50 node network with RIM. Travel Distance is units within the simulation. Power Use is mA. Other values are quantities.

Number of Nodes	Wall clock Runtime
25	2m 40s
50	5m 16s
100	10m 28s

Table 5.2: Comparison of wall-clock run times for 1000s of simulation time.

Metric	Value	Units
Mean Distance Traveled	280.07	units
Mean Hop Count	1.63	
Mean Neighbours	6.76	
Mean Power Use	542.67	mA
Data Message RX	82.19	%
Nodes Moved	24	
Total Distance Traveled	6721.8	units
Total Messages Sent	14845	
Total Power Use	13512124	mA

Table 5.3: Results from 25 node network with multi-node scenario: Three nodes fail sequentially.

ibly equate 1s of simulation with 1 min of 'reality' resulting in the run times in table 5.2 being for 1000 mins (or 16 hours 40 mins). For this to be the case the parameters should be adjusted, but in actuality they are probably closer to 1 min of simulation than 1s.

5.3 Multi Node Failure

Multi-node failure was briefly investigated with RIM producing the results shown in table 5.3. The first multi-node scenario of sequential failing was run for a total of three failures. RIM coped surprisingly well despite being solely for single node failures, the network is fully recovered and reconnected as can be seen in figure 5.6, however four nodes are in a state of oscillation which can be seen in the large distance and energy use values.

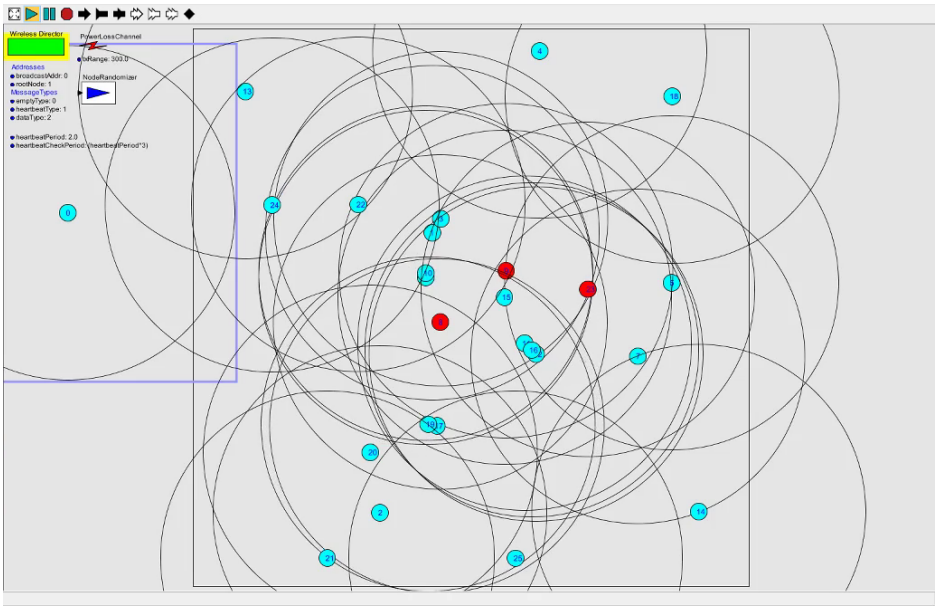


Figure 5.6: Network state post recovery from three sequentially failed nodes. Nodes 6, 10, 17 and 19 are in a state of oscillation.

5.4 Conclusion

Overall RIM seems functional, but the caveats with oscillation, and the specification against multi-node recovery would lead to problems in a real-world deployment. The oscillation issues could be solved with minor adaptations to the algorithm. But it is inherently designed for single-node failure recovery and the unspecified behaviour exhibited in such situations would need fundamental changes to solve. Performing these changes would result in an entirely different algorithm. This development can be seen in SFRA [23] which takes the lessons learned from RIM into an entirely new algorithm specifically designed for simultaneous failures.

6 Conclusions

To conclude, the objectives set out in section 3.2 are reviewed and the key contributions are summarised. This leads to suggestions for future work in the area. Finally closing remarks are made reviewing the entire process. Although the energy consumption model is less accurate than desired, the simulation platform successfully models self-healing algorithms and is able to provide new insights into the performance of different solutions to the problem.

6.1 Completed Objectives

6.1.1 Key Objectives

1. The simulation platform developed with VisualSense provides a generic model for wireless sensor nodes with the flexibility to extend their basic processing for any self-healing algorithm.

2. The RIM implementation runs on top of the platform and is able to provide network recovery from the single node failure scenario.

3. The platform provides data logs which provide information from the individual nodes which is collated and analysed by Octave, these results are presented in section 5.2.

4. Analysis of the data gathered from running the scenario has led to a number of observations of both RIM and self-healing in general, this is expanded further in section 6.2.

6.1.2 Optional Objectives

1. The specification for several additional scenarios is defined, and the platform implements multi-node failure scenarios, however not all forms were able to be tested.

2. Additional algorithms have not been implemented but support exists within the platform to do so in future with no further modification required.

6.2 Key Contributions

Through the implementation and running of RIM with a graphical simulation several points have been raised that were either not discovered or not described in its original presentation.

Firstly the node location edge cases that cause oscillation of node movement. This is a fairly major flaw as not only does it prevent full recovery of the network, but also is hugely wasteful of the limited energy the nodes have available. Oscillations like this are difficult to detect so protection against them needs to be inherent in the algorithms to prevent them from ever occurring. It would seem from RIM that the main cause of the oscillation is from part the algorithm itself designed to prevent unnecessary movement, which it does in many situations, however a better preventative measure should be made to combat both situations.

Secondly, from the descriptions given of the simulated deployments: random dispersement of nodes in a 1000x1000m area with varying numbers of nodes from 25 to 250 and communication ranges from 25m to 200m, deployments would be produced that are highly partitioned and would therefore not involve many of the nodes in any part of the process. If this is the case it would skew the results making them inaccurate and unrealistic for the reported parameters.

The simulation platform created does provide all that it is designed to do, despite the flaw in energy measurement. All different self-healing algorithms can be implemented on top of it, and the results produced will provide a fair comparison between them. This was the primary goal of the project. And with a relatively small amount of modification the energy consumption for each module can be improved to match real-world implementations. The basic platform is there to be built upon and improved over time, but even now it can facilitate evaluation of self-healing algorithms.

6.3 Future Work

As mentioned previously there are many iterative improvements that could be made to the platform: primarily to the energy consumption of

the communication module. By changing this to follow the behaviour of wake-sleep cycles used commonly in wireless sensor networks, the lifetime of the simulated system would dramatically increase to match those of real-world applications. Secondly in the communication module, applying more detailed properties of wireless channels, particularly transmission collisions, would create a much more realistic model for communications. Using varying power loss factors would also improve realism.

Additional scenarios, such as those described in sections 4.3.2 and 4.3.3, would provide a wider range of tests for the implemented algorithms giving a more in-depth analysis. These would particularly show the robustness of the recovery algorithm in dealing with large scale failures and lossy communication channels, which are fairly common situations.

6.4 Closing Remarks

Throughout this project the simulation platform has not been named, however following the various algorithms researched and other platforms and frameworks used a suitable name should be given to it, therefore it will be named SimPl.

The aim of this project was to develop a system through which self-healing for wireless sensor networks could be evaluated. SimPl achieves this goal providing a platform on which any algorithm can be implemented and evaluated in a variety of scenarios.

It also provides a good introduction to self-healing and the factors that should be considered when designing a robust wireless sensor network. Throughout the research and development a number of issues in the field were brought to my attention alongside the need for a common platform, particularly the lack of comparison and development on real hardware to demonstrate theoretical systems working in practice. These themes were mentioned in the literature review, but too far reaching to be further developed. There is potential for these to be addressed in the future and that SimPl can provide a basis for realistic simulation moving to real-world testing.

Bibliography

[1] B. Tong, Z. Li, G. Wang, and W. Zhang, "On-Demand Node Reclamation and Replacement for Guaranteed Area Coverage in Long-Lived Sensor Networks," *Quality of Service in Heterogeneous Networks*, vol. 22, pp. 148—-166, 2009.

[2] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran, "Design and evaluation of a hybrid sensor network for cane toad monitoring," *ACM Transactions on Sensor Networks*, vol. 5, no. 1, pp. 1—-28, 2009.

[3] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits, "Shooter localization in urban terrain," *Computer*, vol. 37, no. 8, pp. 60—-61, 2004.

[4] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard Computing: Sensor Networks in Agricultural Production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38—-45, 2004.

[5] T. Gao, L. K. Hauenstein, A. Alm, D. Crawford, C. K. Sims, A. Husain, and D. M. White, "Vital signs monitoring and patient tracking over a wireless network," *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 27, no. 1, pp. 66—-73, 2006.

[6] M. B. Srivastava, R. Muntz, and M. Potkonjak, "Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving enviroments," in *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01*. New York, New York, USA: ACM Press, 2001, pp. 132—-138. [Online]. Available: http://portal.acm.org/citation.cfm?doid=381677.381690

[7] J. A. Stankovic, A. D. Wood, and T. He, "Realistic applications for wireless sensor networks," *Theoretical Aspects of Distributed ...*, 2011. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-14849-1{_}25

[8] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proceedings of the 1st {ACM} International Workshop on Wireless Sensor Networks and Applications*, pp. 88—-97, 2002. [Online]. Available: http://doi.acm.org/10.1145/570738.570751

[9] E. S. Biagioni and K. Bridges, "The Application of Remote Sensor Technology To Assist the Recovery of Rare and Endangered Species," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 315—-324, 2002.

[10] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking," in *Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X) - ASPLOS '02*. New York, New York, USA: ACM Press, 2002, pp. 96—-107. [Online]. Available: http://portal.acm.org/citation.cfm?doid=605397.605408

[11] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. April, pp. 18—-25, 2006.

[12] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems - SenSys '03*. New York, New York, USA: ACM Press, 2003, pp. 28—-39. [Online]. Available: http://portal.acm.org/citation.cfm?doid=958491.958496

[13] N. Ding and P. X. Liu, "A Centralized Approach to Energy-Efficient Protocols for Wireless Sensor Networks," *Computer Engineering*, no. July, pp. 1636—-1641, 2005.

[14] Y.-c. Wang, C.-c. Hu, and Y.-c. Tseng, "Efficient Deployment Algorithms for Ensuring Coverage and Connectivity ofWireless Sensor Networks," *First International Conference on Wireless Internet (WICON'05)*, pp. 114—-121, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1509645

[15] K. Derr and M. Manic, "Wireless Sensor Network Configuration — Part I: Mesh Simplification for Centralized Algorithms," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1717—-1727, aug 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6459011

[16] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 3, no. 4, pp. 1380—-1387, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036640500037X

[17] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless Ad-Hoc sensor networks," *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01*, pp. 139—-150, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?doid=381677.381691

[18] S. Meguerdichian and M. Potkonjak, "Low Power 0 / 1 Coverage and Scheduling Techniques in Sensor Networks," Tech. Rep., 2003.

[19] Y. Qu and S. V. Georgakopoulos, "A centralized algorithm for prolonging the lifetime of wireless sensor networks using Particle Swarm Optimization," in *WAMICON 2012 IEEE Wireless & Microwave Technology Conference*. IEEE, apr 2012, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6208432

[20] A. A. Abbasi, K. Akkaya, and M. F. Younis, "A Distributed Connectivity Restoration Algorithm in Wireless Sensor and Actor Networks," *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, pp. 496—-503, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4367880

[21] K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, "Distributed Recovery of Actor Failures in Wireless Sensor and Actor Networks," *2008 IEEE Wireless Communications and Networking Conference*, pp. 2480—-2485, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4489467

[22] M. F. Younis and S. Lee, "A localized algorithm for restoring internode connectivity in networks of moveable sensors," *Computers, IEEE Transactions*, vol. 59, no. 12, pp. 1669—-1682, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=5551121

[23] A. Alfadhly, U. Baroudi, and M. F. Younis, "An effective approach for tolerating simultaneous failures in wireless sensor and actor networks," *Proceedings of the first ACM international workshop on Mission-oriented wireless sensor networking*, pp. 21—-26, 2012.

[24] I. Caliskanelli, "A Bio-inspired Load Balancing Technique for Wireless Sensor Networks," Ph.D. dissertation, University of York, 2014.

[25] Y. Wang, "Localized Ant Colony of Robots for Redeployment in Wireless Sensor Networks," Ph.D. dissertation, 2014.

[26] W. Xu, W. Liang, X. Lin, and G. Mao, "Efficient Scheduling of Multiple Mobile Chargers for Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7362022

[27] M. F. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: A survey," *Computer Networks*, vol. 58, pp. 254—-283, 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128613002879

[28] Y. K. Joshi and M. F. Younis, "Autonomous recovery from multi-node failure in Wireless Sensor Network," *2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 652—-657, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6503187

[29] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*, no. October. New York, New York, USA: ACM Press, 2004, p. 45. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1023720.1023726

[30] G. Mao, B. Fidan, and B. D. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51,

no. 10, pp. 2529–2553, jul 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128606003227

[31] E. Egea-Lopez and J. Vales-Alonso, "Simulation tools for wireless sensor networks," *Summer Simulation Multiconference - SPECTS 2005*, pp. 2–9, 2005. [Online]. Available: http://ait.upct.es/{~}eegea/pub/spects05.pdf

[32] B. Musznicki and P. Zwierzykowski, "Survey of Simulators for Wireless Sensor Networks," *International Journal of Grid and Distributed Computing*, vol. 5, no. 3, pp. 23–50, 2012. [Online]. Available: http://www.researchgate.net/publication/234065225{_}Survey{_}of{_}Simulators{_}for{_}Wireless{_}Sensor{_}Networks/file/79e4150eeb702c9511.pdf

[33] J.-J. Lee, B. Krishnamachari, and C.-C. J. Kuo, "Impact of energy depletion and reliability on wireless sensor network connectivity," in *In Proceedings of the SPIE Defense and Security*, R. M. Rao, S. A. Dianat, and M. D. Zoltowski, Eds., aug 2004, pp. 169–180. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=844812

[34] G. Zhou, T. He, S. Krishnamurthy, and J. a. Stankovic, "Impact of radio irregularity on wireless sensor networks," *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, p. 125, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=990064.990081

[35] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, Y. Zhao, C. Ee, C. Brooks, N. Krishnan, S. Neuendorffer, C. Zhong, and R. Zhou, "Visualsense: Visual modeling for wireless and sensor network systems," University of California, Berkeley, Tech. Rep., 2005.

[36] V. Rosello, J. Portilla, Y. E. Krasteva, and T. Riesgo, "Wireless sensor network modular node modeling and simulation with VisualSense," in *2009 35th Annual Conference of IEEE Industrial Electronics*. IEEE, nov 2009, pp. 2685–2689. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5415249

[37] P. Levis and N. Lee, "Tossim: A simulator for tinyos networks," *UC Berkeley, September*, pp. 1–17, 2003. [Online]. Available: http://www.tinyos.net/dist-1.1.0/snapshot-1.1.11Feb2005cvs/doc/nido.pdf

[38] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, W. Weber, J. Rabaey, and E. Aarts, "TinyOS: An Operating System for Wireless Sensor Networks," *Ambient Intelligence*, pp. 115—-148, 2005. [Online]. Available: http://www.springerlink.com/index/10.1007/b138670

[39] Memsic, "IRIS Sensor Node Module," pp. 1–2, 2011. [Online]. Available: http://www.memsic.com/userfiles/files/datasheets/wsn/iris{_}datasheet.pdf