# Reliable Many-to-Many Routing in Wireless Sensor Networks using Ant Colony Optimisation

by

**Jasmine Grosso**

**Thesis**

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

**Doctor of Philosophy**

**Department of Computer Science**

October 2021

# Contents

# List of Tables

# List of Figures

# Acknowledgments

Firstly, I would like to thank my supervisor, Arshad Jhumka, for his support and guidance in pursuing this work, without whom it would not be possible.
I would also like to thank Matthew Bradbury, who gave excellent advice and help throughout the course of this work.
Thank you to my fellow PhD students, Liam, Melissa, and Katherine, for making times spend in the office (when we were allowed in an office) a whole lot more fun.
I would also like to thank all of those involved in the maintenance of the Flux and the other resources in the Department of Computer Science.
Finally I would like to thank my boyfriend Chris for his love and support, I wouldn't have made it through without him.

# Declarations

This thesis is the authors own work. The thesis has not been submitted for a degree at another university.

## 1 Publications

Parts of this thesis have been previously published by the author in the following:

[48] J. Grosso, A. Jhumka, and M. Bradbury. Reliable many-to-many routing in wireless sensor networks using ant colony optimisation. In *2019 15th European Dependable Computing Conference (EDCC)*, pages 111–118, 2019. doi: 10.1109/EDCC.2019.00030

[49] Jasmine Grosso and Arshad Jhumka. Fault-tolerant ant colony based-routing in many-to-many iot sensor networks. pages 1–10, 11 2021. doi: 10.1109/NCA53618.2021.9685935

## 2 Sponsorships and Grants

This research was funded by the Engineering and Physical Sciences Research Council (EPSRC).

# Abstract

A wireless Sensor Network (WSN) consists of many simple sensor nodes gathering information, such as air temperature or pollution. Nodes have limited energy resources and computational power. Generally, a WSN consists of source nodes that sense data and sink nodes that require data to be delivered to them; nodes communicate wirelessly to deliver data between them. Reliability is a concern as, due to energy constraints and adverse environments, it is expected that nodes will become faulty. Thus, it is essential to create fault-tolerant routing protocols that can recover from faults and deliver sensed data efficiently. Often studied are networks with a single sink. However, as applications become increasingly sophisticated, WSNs with multiple sources and multiple sinks become increasingly prevalent but the problem is much less studied.

Unfortunately, current solutions for such networks are heuristics based on specific network properties, such as number of sources and sinks. It is beneficial to develop efficient (fault-tolerant) routing protocols, independent of network architecture. As such, the use of meta heuristics are advocated. Presented is a solution for efficient many-to-many routing using the meta heuristic Ant Colony Optimisation (ACO). The contributions are: (i) a distributed ACO-based many-many routing protocol, (ii) using the novel concept of *beacon ants*, a fault-tolerant ACO-based routing protocol for many-many WSNs and (iii) demonstrations of how the same framework can be used to generate a routing protocol based on minimum Steiner tree. Results show that, generally, few message packets are sent, so nodes deplete energy slower, leading to longer network lifetimes. The protocol is scalable, becoming more efficient with increasing nodes as routes are proportionally shorter compared to network size. The fault-tolerant variant is shown to recover from failures while remaining efficient, and successful at continuously delivering data. The ACO-based framework is used to create Steiner Trees in WSNs, an NP-hard problem with many potential applications. The ACO concept provides the basis for a framework that enables the generation of efficient routing protocols that can solve numerous problems without changing the ACO concept. Results show the protocols are scalable, efficient, and can successfully deliver data in numerous different topologies.

# Acronyms

|  |  |
|---|---|
| **ABC** | Artificial Bee Colony. |
| **ACO** | Ant Colony Optimisation. |
| **ACO-QoSR** | ACO-based quality-of-service routing. |
| **ACS** | Ant Colony System. |
| **CCP** | Coverage Configuration Protocol. |
| **CTP** | Collection Tree Protocol. |
| **DD** | Directed Diffusion. |
| **EEABR** | Energy-Efficient Ant-based Routing Algorithm. |
| **ETX** | Expected Tranmissions. |
| **GAF** | Geographical Adaptive Fidelity. |
| **GEAR** | Geographic and Energy Aware Routing. |
| **GPS** | Global Positioning System. |
| **GRAB** | GRAdient Broadcast. |
| **GSO** | Glowworm Swarm Optimisation. |
| **HVAC** | Heating, Ventilating, and Air Conditioning. |
| **k-LCA** | K-restricted Loss-Contracting Algorithm. |
| **LEACH** | Low Energy Adaptive Clustering Hierarchy. |

**MANET** Mobile Ad Hoc Networks.

**MCFA** Minimum Cost Forwarding Algorithm.

**MECN** Minimum Energy Communication Network.

**MMAS** MAX-MIN Ant System.

**MMSPEED** Multipath Multispeed Protocol.

**MRMS** Multipath Routing in large scale sensor networks with Multiple Sink nodes.

**MUSTER** Multisource Multisink Trees for Energy-Efficient Routing.

**PEGASIS** Power-efficient GAthering in Sensor Information Systems.

**PSO** Particle Swarm Optimisation.

**Q-MST** Quadratic Minimum Spanning Tree.

**QoS** Quality of Service.

**REAR** Reliable Energy Aware Routing.

**ReInForm** Reliable Information Forwarding.

**SMECN** Small Minimum Energy Communication Network.

**SPIN** Sensor Protocols for Information via Negotiation.

**STP** Steiner Tree Problem.

**TTDD** Two-Tier Data Dissemination.

**WSN** Wireless Sensor Networks.

# Chapter 1

# Introduction

## 1.1 Wireless Sensor Networks

Wireless Sensor Networks (WSN) are networks formed of multiple sensor nodes placed or scattered in an area in order to measure environmental factors, such as temperature, sound, or pollution levels [4][5]. The sensor nodes vary in characteristics depending on the application and goals of the WSN, however they usually share a number of common characteristics. Though they vary in size, sensor nodes are often relatively small, and usually are inexpensive in order to allow networks of large numbers of nodes [4]. Sensor nodes will have limited energy resources, often being powered by ordinary batteries, and usually will not be able to harvest energy from the environment (though some sensor nodes do have this ability and this forms another branch of research [111]) hence power available to them is limited [4]. Computational resources are also constrained, with sensor nodes often requiring specialised minimal operating systems like Contiki OS [35] or Riot [9]. Sensor nodes in a network are able to communicate wirelessly over limited distances using a radio transceiver [10], and much research has been carried out in routing for these communications.

WSN have many advantages; their simplicity, flexibility and robustness allow them to be placed in environments where traditional networks are not possible. Often wireless sensor networks are placed in dangerous or difficult to reach locations and left to run without human intervention. Additionally they are often a cheap solution to many varied problems due to their low equipment and maintenance costs. However, they come with a number of limitations, particularly arising from their limited energy resources and computational power, requiring novel, efficient solutions for routing messages[4] [5].

Sensor nodes measure and monitor a large range of environmental conditions, including temperature, air quality, sound, and humidity [4][5]. WSN setups may include several types of nodes [4][7]:

- Source nodes, which are the sensing nodes that detect an aspect of the environment. Often these form the majority of the sensor nodes in a network. They will usually then pass on this information to another node in the network using wireless communication.

- Sink nodes, towards which this sensing data needs to be sent towards. The sink node may be an actuator, that could then act on the data that has been sensed. For instance, if a decrease in temperature is detected, an actuator may be to increase heating to a room [3]. A sink node could also be, or be in communication with, a computer with higher processing power to in some way deal with the data further. For example, the data may be passed to a different network, or processing may be performed with higher computational power. These types of sink nodes are often referred to as a base station [5].

- Relay nodes pass information from source to sink nodes using a radio broadcast medium. A node may be some combination of all three of these types of nodes in a sensor network, acting as both a source and a relay node at the same time, or a sink and a relay node. [77].

Wireless Sensor Networks have a number of advantages over traditional wired networks. As the sensor nodes themselves are generally simple, small, and limited in computing resources, wireless sensor networks can be inexpensive to produce and set up. A WSN requires less precision in the placement of nodes, not requiring it to be close to the object or phenomenon that is being monitored. Sometimes sensor nodes are placed randomly, which is an advantage in dangerous or inaccessible areas. Additionally, a wireless sensor network can be created quickly without need for detailed topology plans. They are flexible for a number of environments as the topologies do not always need to be planned in advance [4] [5].

## 1.2 Data Routing in WSN

Sensor nodes communicate with each other wirelessly using a radio [10]. This communication is used to route sensed data from source nodes to sink nodes. In some scenarios, routing protocols assume that all source nodes are able to communicate with the sink or base station in a single message, known as one-hop or single-hop communication. Other networks will require what is known as "multihop communication" where sensor nodes can communicate with their closest neighbours only. This could be by necessity due to radio range, and so requiring several messages or hops to send data to the sink [5]. Multihop communication allows a much greater range of possible application areas, as the network no longer requires every node to globally communicate with every other node in the network. Multihop routing problems are a large area of research in the field of wireless sensor networks, as many problems arise in minimising the number of messages sent in order to reduce energy used in sending packets, and so increasing the lifetime of the network [109] [1].

Sensor networks may be placed in a great variety of environments, with varied topologies each with different limitations. This leads to the necessity for flexible routing protocols that may be used in many different scenarios. For example, in some networks, the sensor nodes may be mobile, adding an additional layer of complexity in developing protocols [8]. A major form of variation between sensor network setups is whether the WSN is centralised or distributed. Distributed networks mean that each individual sensor node is unaware of the overall network topology, and can only communicated locally with its immediately reachable neighbours. Centralised networks operate with overall knowledge of the network, either with each node being aware of global topology, or a single base station having this knowledge and directing routing accordingly. Each type of network presents different challenges in terms of developing routing protocols [133]. For instance, centralised networks tend to lead to more efficient routes being formed, however this is at the expense of additional computational complexity of algorithms, as well as the tendency of suffering more from node failures. Distributed networks may have less efficient overall routes, however are more robust and scaleable.

WSN consist of both source nodes that sense data about the environment, such as temperature or sound, and sink nodes that need to receive that data. Simple WSN may consist of a single source directing the data that it senses to a single sink, a situation with limited application areas. WSN with multiple sources needing to send data to a single sink (many-to-one) is a research area that is well explored, and often assumes many sensing nodes reporting data to a single base station for further data processing. One-to-Many networks are less common, and so fewer routing protocols have been developed for this scenario, however there are some applications where, for example, multiple actuators would need to respond to the same environmental change detected by a single source. Example topologies of one-to-one, many-to-one, and one-to-many wireless sensor networks are illustrated in figure 1.1.

WSN with both multiple sources and multiple sinks (many-to-many) are a infrequently studied topology type, despite their potential in a number of different application areas as WSN become more sophisticated. Many-to-Many networks have a number of advantages, as they are able to fully utilise a network of sensor nodes. Often a sensor network will have a large proportion of nodes with the ability to sense and act as sources, which may also potentially

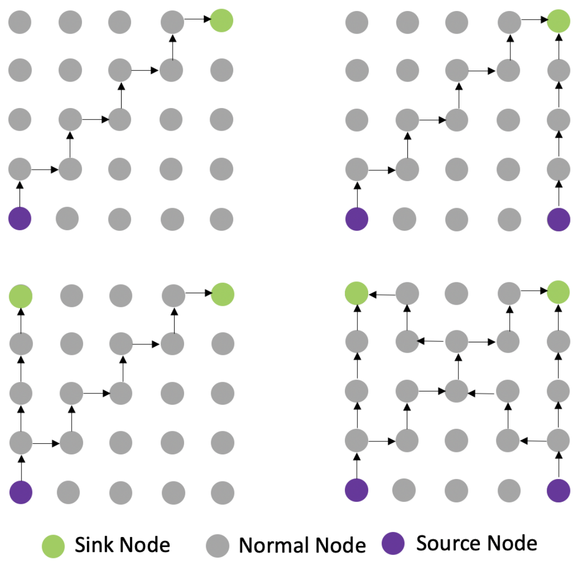


Figure 1.1: Examples of a one-to-one, many-to-one, one-to-many, and many-to-many Wireless Sensor Networks

need to send messages to different sinks in the network. For instance, actuator networks which require data to many actuators from multiple regions are a potential application. Many-to-Many networks also increase the ability for fault tolerance, as additional sinks could introduce alternative strategies and flexibility for dealing with node failures. Generally a many-to-many network allows greater flexibility in the application of a WSN, allowing sensed data from multiple sources to be delivered else were, for either further computation or immediate action by the network.

Though there are a large number of exciting opportunities, it is challenging to develop routing protocols for many-to-many networks. More sinks may require each node to store more information about the network in memory, which in a resource limited sensor node could be problematic. This often means that routing protocols with the requirement for each node to have global knowledge of the network an unsustainable starting condition, leading to the necessity of distributed protocols. It may be hard to scale routing protocols for larger networks, for instance tree based protocols may create a tree routed at the sink, but with multiple sinks this may become too complex as multiple trees would be required. Existing solutions for many-to-many routing are often not scaleable, either requiring a centralised view of the network, or requiring large scale recalculation of routes for each sink. Many existing solutions consist of routing protocols developed for many-to-one networks simply repeated for each sink, an inefficient way of developing routes. Creating an efficient route that minimises the number of messages sent between nodes, a desired characteristic as it will deplete the battery less and increase the lifetime of the nodes, is challenging, as increasing the number of sources and sinks in many routing protocol will increase the number of messages required.

Many routing protocols developed for WSN require a specific set of network conditions to be successful. For instance, many required each node to have some level of global knowledge of the network, which limits the scaleability. Protocols are also often designed for topologies with a single sink, limiting applications. There is a necessity for routing protocols that do not require such specific conditions in order to operate successfully, while also being scaleable and efficient in delivering data. Existing solutions often make use of heuristics dependent on the problem, however meta heuristics independent of problem starting conditions may form a more effective solution.

In addition to efficient routing under normal conditions, it is often required that wireless sensor networks are to be resilient to faulty sensor nodes and harsh environments. The nature of the sensor node devices are such that they often run out of energy, or may fail in other ways due to their low cost. Additionally, the networks are often placed in dangerous areas where they may be destroyed [7]. An advantage of the WSN is that they are placed in these dangerous areas or hard to reach places to minimise human interaction, though this means that the nodes are not easily fixed or replaced when they fail. As such, it is necessary to build fault tolerant routing protocols to deal with node failures. As the sensors are inexpensive, a large amount of redundancy can be built into the network in case of sensor failure or destruction. The flexibility in topology also adds to this resilience [7][131], as routes could be recalculated. However, there is still a need for fault tolerant protocols that are able to efficiently recover from faults when they occur. Such protocols may be challenging in distributed networks, as there is no way of determining which nodes in the network have failed due to the lack of global knowledge. Many existing solutions rerun the initial route finding protocol when node failures occur, an inefficient solution. For this reason it is imperative to develop scaleable fault tolerant protocols that deal with faults efficiently.

Routing problems in WSN often take the form of NP-complete problems, for instance, routing can be considered to be a variant of the vehicle routing problem, an NP-hard problem [70], or a variant of the Steiner Tree problem [57]. Traditional route finding algorithms are often inefficient for such problems, and heuristic based solutions will be problem specific, and often greedy, which is not efficient for WSN. Meta heuristics present many potential solutions for routing in WSN, as they are adaptable to the situation.

WSN have many advantages in their flexibility, however present challenges in the necessity for fault tolerance and distributed routing. As applications become more complicated and sophisticated, the necessity for many-to-many routing protocols becomes more obvious, however little work has been carried out in this area. Existing work will often use heuristics that require specific starting conditions, and so meta heuristics present an elegant way of generating routing protocols with less dependence on starting conditions the specific problems. These challenges and problem areas form the basis of this work, and addressing these issues will allow WSN to be deployed more successfully in many application areas.

## 1.3 Application Areas

Wireless Sensor Networks can be used in a large number of varied environments to monitor a wide range of conditions, due to their flexibility and resilience to problems. This section explores some of these application areas, and examples of previous works that use them in these areas. This indicates the advantages of this type of network, but also illustrates some of the challenges in developing protocols.

### 1.3.1 Environmental Monitoring

Environmental monitoring is the observation of various environmental variables, such as temperature, humidity, light, and air pressure [94]. Often, these variables need to be measured over a large region and therefore may require a large number of sensors [92]. Environment monitoring applications include detection and monitoring of natural disasters such as earthquakes or floods, monitoring of agriculture, and weather forecasting [92].

For many environmental monitoring applications, a network must be reliable in variable conditions, such as harsh or difficult to predict weather, as well as robust in the case of total node failure or changing topologies, caused by, for instance, a drop in connectivity. Flexibility in the network is an advantage when used for environmental monitoring, as often the environments themselves change, or the requirements of monitoring changes. A level of autonomy of the network is also useful, as a reducing the need to interact with the network lessens the chance of humans changing the environment being monitored accidentally, in addition to reducing danger to people. [94]

Wireless Sensor Networks are particularly suited in the field of environmental monitoring, due to the fact that they do not need frequent human intervention to continue running. The sensor nodes are battery powered and many protocols aim to allow the longest possible network lifetime by reducing energy usage.

They allow real time monitoring of the environment, without risking human maintenance affecting the environment being monitored [94]. Additionally, they may be spread out over a large area to monitor large regions, a feature that is often impractical for more traditional devices. Generally, sensor networks can be designed to be fault tolerant, and so will be better able to survive harsh conditions. Again, the low cost of the sensor nodes is an advantage in this scenario.

An example use of wireless sensor networks for environmental monitoring can be seen in the work presented in [120], where a wireless sensor network is used to monitor redwood trees in California. As the trees are so large, they experience a variation of weather conditions from top to bottom, with changes in humidity, light, and temperature moving around the tree. Previous solutions were unwieldy, involving a researcher climbing the tree to place environmental monitoring equipment to the top of the tree, which is then attached by cable to a battery at the bottom of the tree. This setup was not able to monitor the changes in climate for the tree in any real detail. A WSN would present many advantages in this situation, as no cables are required to connect them to batteries or each other. They should allow higher resolution data on the climate factors around the tree, and also do not require a person to go to the sensors to collect the data gathered. In this study, a sensor network was setup that would sample all the sensors every 5 minutes, with sensors placed at various places within the tree. This study showed that despite some issues with installation, the network was able to gather data to validate biological theories that previous methods were inadequate to enable. There were some issues in installation as well as problems in inadequate fault detection and tolerance within the network, which shows the importance of such factors in designing a WSN. It was found that a WSN is able to provide dense spatial and temporal monitoring in real world environments.

Other examples of environmental monitoring include:

- [50] An automated irrigation system for efficient water usage in irrigating crops. Sensors are placed nearby plants in order to monitor soil moisture levels and temperature, triggering actuators for watering. It was found that this lead to 90% water savings. This is an example where a multiple sink, multiple source network may be advantageous, as multiple source sensors could be used to sent information to sinks associated with irrigation systems throughout the crops.

- [125] A WSN was deployed in an active volcano, collecting data via multihop to an observatory. Compared to traditional approaches, the sensors were smaller and had lower energy consumption, requiring far smaller batteries and less human interaction in a potential dangerous environment.

- [11] SensorScope is an environmental monitoring system, in this case deployed in a glacier in Switzerland for various goals including creating precise maps of an area where environmental conditions make traditional wired network inadequate for gathering the detail required. Such measurements are useful in flood monitoring and prediction.

### 1.3.2 Structural Monitoring

In addition to more natural environments, WSNs are also used in the monitoring of buildings and urban areas. The advantages of a WSN in this setting include the low cost of the network, the ability to place sensors in locations that may be too dangerous or otherwise inaccessible for people to access frequently, such as in ceilings, and the lack of wires makes the network both safer for people using the building in terms of trip hazards, as well as being more aesthetically pleasing [61]. Common uses of WSN in buildings include monitoring lighting, heating, ventilating, and air conditioning (HVAC) [61].

In [66], Kintner-Meyer et al. used wireless sensor networks for HVAC in commercial buildings. Wiring can cost being 20%-80% of a HVAC system [66], and so using a wireless sensor network can represent a significant amount of savings. Two implementations are carried out, with one setup using the network in a large high rise commercial building to monitor temperature and reduce energy usage, and in the other setup a smaller building with rooftop units. Though there were some uncertainties in the cost analysis of the systems, it was found that the wireless sensor networks represented a decrease in cost, as well as being easier to be extended than wired networks. HVAC is a scenario where many-to-many networks could be implemented with success, as having the heating and air conditioning respond to conditions throughout a building could improve efficiency.

Wireless sensor network is used to monitor the structural integrity of both historic and modern buildings. In case of historic buildings, such as [16], WSNs are prefered over a traditional wired network as the smaller sensor nodes can be deployed with less visual impact to the artwork and architecture of these buildings without need for access to power outlets. In this case, nodes were placed over four floors and throughout the course of monitoring the tower was accessible to the public. The building itself posed challenges in the fact that the stone walls were very thick. The goal of the network was to alert for potential structural issues with the building. The network had a low loss rate (loss rate indicating the reliability of delivery of data), however a single faulty node led to a spike in the loss of packets. This indicates that a protocol that can deal with faulty nodes would be advantageous in such real-world implementations of WSN. For structural monitoring, it is necessary to identify structural damage as early as possible, to prevent further damage and maintain safety standards, with particular importance placed on the monitoring of structures such as bridges in locations prone to seismic activity. As such, wireless sensor networks are increasingly used in such scenarios, due to their robustness, flexibility, and low cost. In [79], wireless sensors were installed in the Alamosa Canyon Bridge in New Mexico and found the network to be reliable and accurate in comparison to a wired network when measuring forced vibrations of the bridge.

A drawback of the wireless network was the limited energy resources of the sensors, showing a need to conserve energy usage in order to maximise network lifetime, or perhaps make use of energy harvesting. However in general wireless sensor networks show great potential in this area as the technology develops.

### 1.3.3 Habitat Monitoring

Wireless Sensor Networks have been extensively applied in the field of habitat monitoring [81]. WSNs in particular are useful in this area, due to the fact that they can usually be left alone to run without human intervention. The small size of the sensors mean that they are less invasive in an environment than a traditional wired network, allowing accurate monitoring of the area without influencing the habitat. The habitats being monitored may be harsh, meaning that repeated in person studies could lead to dangerous situations for the people carrying them out. A WSN would also reduce the cost of monitoring, with lower setup and maintenance costs. Additionally, with a WSN, sensor nodes may be placed in previously inaccessible areas, increasing the range of habitats that can be studied [81].

In [81], a case study is described where the habitats of ducks are monitored in the Great Duck Island in Maine, USA. A wireless sensor network is put in place on several small islands in order to monitor the ducks, including the nesting habits of breeding pairs. 32 sensor nodes are placed on the duck island, including 9 underground. Readings indicate that recorded information matches previously seen patterns in humidity, showing the accuracy of the system.

Wireless Sensor Networks have been implemented in [90] in order to monitor seabirds on Skomer Island, a UK nature reserve. Monitoring seabirds in such an environment enables researchers to track the overall ecological health of the island. Prior to the implementation of a wireless sensor networks, birds were tracked using GPS loggers, which needed frequent replacement of batteries as well as manual downloading of data. A WSN was setup with a single sink to monitor the activities of the birds as well as conditions in their burrows. The reliability of measurements of the network was found to be good, and additionally it was found that deploying the network did not lead to a change in the behaviour of the birds. Problems were found in that the configuration of the network needed to be changed with changing conditions in the fields. This suggests that protocols that can adapt to changes in the environment, for instance a change in node location due to environmental conditions, would be advantageous.

### 1.3.4 Industrial Monitoring

Wireless Sensor Networks have a place in industrial monitoring of equipment and production. For instance, machinery condition-based maintenance involves continuously monitoring the state of equipment in real time, performing maintenance only when necessary. This method of maintenance optimises resources, ensuring costs and downtime is minimal. Wireless Sensor Networks are particularly suited to this task as due to their small size, they are able to be placed within small spaces in machinery that would be impractical or completely inaccessible for a traditional wired network, such as moveable parts. As WSNs are often distributed, the sensor nodes could be easily moved around the machinery where the need arises. WSNs are quick to install and can usually be left unattended, reducing the cost involved in setting up and maintaining the network [119].

In [119], a wireless sensor network is used to monitor a Heating and Air-conditioning Plant. In this case, the set up time was indeed minimal, taking only 30 minutes. The network was successfully used in the real time monitoring of equipment.

## 1.4 Contributions

The main contributions of this work focuses on wireless sensor networks with multiple sources and multiple sinks, or many-to-many WSN. This is an area of research that is little studied, with solutions often involving the inefficient calculation of separate routes for each sink, despite the usefulness of a network that is able to sent data to multiple locations.

Current solutions often make use of heuristics for routing problems, requiring a specific set of starting conditions. Meta heuristics form the basis of the work here, as they do not require a particular set of starting conditions for the network but are adaptable to different scenarios. They are able generate efficient solutions of NP hard problems that are raised by the constrained conditions of energy limited sensor nodes in a many-to-many WSN.

A framework based on the Ant Colony Optimisation [29] meta heuristic is presented, from which many routing protocols can be developed from applying to a number of scenarios and problems relating to many-to-many WSN. The ACO based routing protocol does not need a certain set of conditions to be successful, instead the input parameters of the framework can be changed to suit the topology and the environment the network is placed in. The same concepts of the protocol are used with some extensions or variations in order to be applied to different topologies, making the framework adaptable to a number of different scenarios. A summary of the contributions are explored in this section.

### 1.4.1 Many-to-Many Routing in Wireless Sensor Networks using ACO

Many-to-Many routing in WSN is the problem of ensuring that sensing data from multiple sources $s_1, \dots, s_k$ is delivered to multiple sinks $\Delta_1, \dots, \Delta_l$, while sending a minimal number of packets with a minimal number of nodes. In this way, fewer messages will be sent overall and so network lifetime is increased as less energy is expended. To solve this problem, a distributed ACO based routing protocol has been developed that utilises the concept of a shared route, or backbone, of nodes that aggregates messages from many sources before directing messages towards many sinks. This backbone has the advantage of scaleability, meaning that overall fewer messages are sent due to the effects of aggregation.

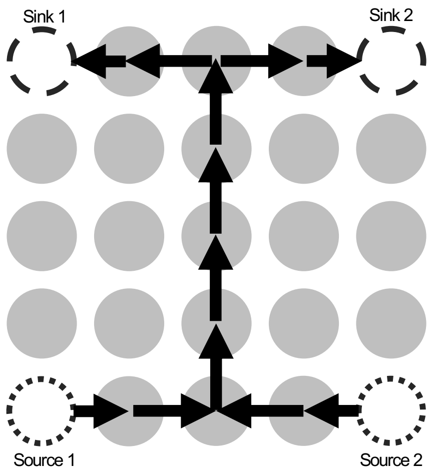


Figure 1.2: Example of a route from sources to sinks using a shared backbone.

The concept of the shared backbone of nodes is illustrated in figure 1.2. The main advantage of this backbone is to reduce the overall number of nodes involved in routing, and also to reduce the number of messages sent. This increases network lifetime, and also leads to a more scaleable protocol that is still efficient as network size increases. This backbone also has the advantage of allowing the aggregation of data, potentially reducing the size of the payload to be sent. To form a backbone, multiple messages originating from different sources receive on the same node. From this node, a single aggregated message will be forwarded for as long as possible, before splitting into multiple messages again to be forwarded to multiple sinks. The backbone concept can be related to the concept of a backbone mesh in Mesh networks, formed of many routers. However, this is quite different to what is proposed here, as in WSN energy concerns are much greater and form much of the challenge in routing. Additionally, there is greater homogeneity of nodes in WSN, with any node possibly forming the backbone, not just routers as in mesh networks.

The protocol that has been developed is distributed, which is often a necessity in large networks where nodes cannot hold information about the whole network, or where there is no power base station that can have a centralised view of the network. Outside of necessity, a distributed protocol is very advantageous in WSN as it simplifies the amount of information each individual node needs to deal with, with each node only requiring knowledge of localised information about its neighbours. This has the effect of making the protocol more scaleable to increased network size. Additionally, when only local information is necessary to make routing choices, the protocol is more resistant to changes in the network, allowing greater flexibility.

## 1.5 Fault Tolerant Many-to-Many Routing in Wireless Sensor Networks

The ACO meta heuristic is extended to create a Fault Tolerance routing protocol for many-to-many WSN, specifically the scenario where nodes fail during the running of the protocol. The ACO meta heuristic based routing protocol is still able to efficiently generate routes in a many-to-many WSN efficiently while dealing with faulty nodes.

The fault tolerant version of the ACO based routing protocol aims to solve the problem of given multiple sources $s_1, \dots, s_k$ and multiple sinks $\Delta_1, \dots, \Delta_l$, a path should be found between them such that the protocol is tolerant to an arbitrary number of node failures. WSN often are placed in inaccessible environments where humans are not able to access and repair nodes frequently, however node failures are still likely to occur through energy depletion or other adverse conditions. This makes maintaining successful operation of routing from sources to sinks without outside intervention important. This means that it is necessary to develop a routing protocol that is able to recover from faults in the network while still being able to route from multiple sources to multiple sinks. A fault tolerant variant of the ACO based protocol is developed to solve this problem.

The many-to-many ACO routing protocol is adapted in the fault tolerant variant, however an additional type of ant is introduced to detect faults. Beacon ants are sent from every node that track which neighbours may be faulty. ACO can be adaptive to node failures, as only local decisions are made by each ant. This means no network wide updates are required, or wholesale recalculation of routes that take the node failure in account. Instead, only the local decision of whether to travel to a node is required to be considered. As ACO minimises the number of nodes involved, often failed nodes are not on the route at all, leading to resistance to many node failures. To recover from faults that are on the main route followed by the ants, targeted evaporation of the pheromone trail is used to encourage ants to choose different nodes that have not been detected as faulty. Pheromone trail is evaporated more on nodes that are more likely to have failed.

## 1.6 Generating Steiner trees in Wireless Sensor Networks

The final contribution is an exploration of the ability of the ACO meta heuristic framework to create Steiner Trees in a many-to-many WSN, with varied number of sources and sinks. A Steiner Tree is a popular and useful structure to form in a network due its ability to efficiently connect sensor nodes, and using the ACO meta heuristic these can be build efficiently in a distributed fashion, only changing the inputs to deal with different input problems. This investigation also shows the ability of the ACO based framework to be applied to different problems, in this case making some changes to the start up phase of the algorithm and how it decides when to split and combine into a backbone.

The terminal nodes of the Steiner tree generated by the protocol will be the sources and sinks in the network, which are connected by any number of additional sensor nodes that act as Steiner points. The use of a shared backbone is also utilised here, as it enables the generation of minimal Steiner Trees in certain topologies. In order to connect an arbitrary number of sources and sinks in a Steiner Tree, the concept of a preliminary pheromone trail in introduced. This attempts to solve the problem by initialising the pheromone trail between nodes in such as way as to encourage the formation of Steiner Trees. A fault tolerant variant that also makes use of beacon ants is also developed to solve the problem of node failures in the network.

## 1.7 Protocol Performance

Initial results show that the ACO meta heuristic based routing protocol for many-to-many WSN is effective in building efficient routes from all sources to all sinks for a wide range of network sizes and scenarios. The protocol is successful in delivering data, having a high delivery ratio of packets sent. The initial protocol developed had a delivery ratio of 94.8% for a network of 121 nodes with two sources and two sinks, compared with a figure of 69.2% for flooding in the same network. Across multiple network sizes ranging from 25 to 169 nodes, the mean delivery ratio was 93.3%. The number of packets sent is minimised, with the protocol becoming more efficient as the network becomes larger. When considering the initial protocol with 121 and two sources and two sinks, there are 31.5% of the nodes involved to route, compared with flooding at 43.7%. The fault tolerant variant of the protocol is able to detect and respond to faults, routing messages around failures in the network to still maintain a high delivery ratio and short routes. With 5% of nodes failures, the Fault Tolerant variant of ACO maintained a delivery ratio of 87.6% in a network with 121 nodes. The Steiner Tree variant is able to build Steiner Trees between a range of sources and sinks in a distributed fashion, and the fault tolerant variant for building Steiner Trees shows some promise.

## 1.8 Organisation

This chapter has introduced wireless sensor networks, their applications areas, and the challenges that arise from their limitations. A number of real world applications have been explored showing the usefulness of WSN. These use cases emphasises the necessity for robust and flexible wireless sensor nodes, with elegant solutions for routing protocols. The rest of this work is organised as follows:

- Chapter Two presents and review of existing literature in the field of WSNs.
- Chapter Three describes the problem statement to be solved and experimental setup followed.
- Chapters Four to Six present the technical contributions.
- Chapter Seven presents some discussion of the work and processes learnt.
- Chapter Eight concludes and discusses future work directions.

# Chapter 2

# Background

Wireless Sensor Networks (WSN), made up of large numbers of simple, low cost, and low power sensor nodes, have the potential to solve problems in a number of applications areas in science and industry. Due to their power constraints and the extreme environments they may be placed in, protocols developed for the devices must conform to a number of constraints. Challenges arise in limited power resources for the nodes, leading to the necessity to reduce the number of messages sent in order to increase network lifetime. Additionally, node failures may be common in a WSN, due to the potential for nodes to run out of energy, or to be destroyed by the environment they are in, and so routing protocols need to deal with node failures when they occur. Many existing routing protocols assume a network of a single source and a single sink, or a network with multiple sources and a single sink. Such solutions may be useful in limited circumstances, however are often not adaptable to different situations. Limited solutions with both multiple sources and multiple sinks exist, however these are not common and generally do not scale well, in many cases requiring a particular starting setup. An efficient solution that does not require specific network conditions can be provided with the use of meta heuristics such as Ant Colony Optimisation. Steiner Trees in WSN are a useful form of providing routing solutions, as they are able to connect together a generic number of nodes with minimal cost, and so the research that has been carried out in this area is also discussed.

A review of these existing technologies for wireless sensor networks is presented in this chapter. The topics covered are broadly; routing in wireless sensor networks, separated into the categories of one-to-one, many-to-one, and many-to-many routing, the use of Ant Colony Optimisation in WSN, fault tolerant protocols for WSNs, including fault tolerant routing, and finally, existing work relating to the formation of Steiner Trees in WSNs is described.

## 2.1 Routing in Wireless Sensor Networks

Routing in Wireless Sensor Networks is a widely studied problem with large numbers of existing solutions. Routing is the process of directing sensing data, for instance environmental information like temperature or pressure, from source sensor nodes towards the sinks or the base station of the network. In smaller WSN, nodes are close enough to the sink or base station such that all nodes can communicate with it directly, this is known as single hop communication. However, more often sensor nodes are only able to communicate with close neighbours, and so to send data to the sink nodes multi-hop communication is required. This is where nodes send messages to neighbouring nodes they can communicate with until the sink is reached. Routing is the process of finding multi-hop paths from sources to sinks [109].

A number of challenges present themselves when developing routing protocols for WSN. In general, a large amount of energy that is expended by sensor nodes originates from communication using a radio [37]. As a consequence of this, the goal of many routing protocols is to send as few messages as possible, in order to reduce energy expended to increase network lifetime. As wireless sensor networks are usually more constrained in power than typical wired networks, energy efficient routing protocols is of particular value [1]. In addition to the low power resources, sensor nodes also have low computational resources, and so complicated routing protocols are often not possible. Other problems arise

from the need for distributed protocols. Many protocols assume a centralised approach is possible, for instance if all nodes have a global knowledge of topology or if a base station has a centralised view of the network. However, often in real world scenarios this is not possible to implement and so it is necessary to develop distributed protocols where each node has no global knowledge of the rest of the network. Though routing protocols have some similarity to classical path finding algorithms, creating efficient paths in a distributed network with no overall knowledge of topology makes these algorithms difficult to implement, so a large research area is finding efficient routing protocols under these conditions. Routing protocols should be scaleable, remaining efficient in networks with hundreds of sensor nodes. Solutions for routing should also have a level of robustness, as due to the nature of wireless sensor networks and the locations they are often installed, failures of transmission will occur often. It is generally desireable for routing protocols to have a level of robustness and reliability when failures occur. [109] [1] [5] [83] [2]
When classifying protocols, it is often helpful to categorise based on the topology of the network they are used in. The number of sources and sinks is an important feature, and allows the categorisation of how routes are formed. These types of routing are:

- *One-to-One*: Routing from a single source to a single sink, this is the most simplistic form of routing. This is often not scaleable, as most networks will have multiple nodes sensing information acting as sources, and as such, the protocol will simply be repeated for each source node. This will lead to more messages than necessary being sent.

- *Many-to-One*: This type of routing will direct many messages from multiple sources to a single sink. This type of setup may be useful where the network has a single base station that deals with all messages, however this scenario is not applicable to every WSN. Often hierarchical protocols are considered to be many-to-one, as many sensor nodes report to a single cluster head.

- *One-to-Many*: This network topology is less common, and as such is not well investigated.
- *Many-to-Many*: Routing from multiple sources to multiple sinks is the most flexible of WSN routing, as it allows a large range of network setups and conditions.

It should be noted that there may be overlap between the categories, as for instance, a one-to-one protocol could be repeated in the same network for a different source to also deliver messages to the sink. This could be considered many-to-one, however as the protocol in this scenario is naively repeated it will likely be inefficient and not scaleable. For the sake of this review, a routing protocol that can be repeated in order to deliver from multiple sources to one or multiple sinks will still be considered to be one-to-one, as each instance of the protocol is only performing a single routing path.

### 2.1.1 One-to-One Routing

This section investigates existing one-to-one solutions for routing in wireless sensor networks. These are often the simplest and earliest protocols, simple to implement but often inefficient and not scaleable.

A classic and simplistic form of routing in WSN is flooding [54]. Flooding consists of nodes, on receiving a message, broadcasting that message to all of its neighbours repeatedly until the destination node is reached. The technique is simple, does not require expensive setup processes, or expensive updates in the case of topology changes like node failures. However, three main issues arise with this technique. 'Implosion' is where nodes receive duplicate messages where multiple neighbouring nodes broadcast the same message. 'Overlap' is where multiple nodes sense the same data and each broadcasts to its neighbour, leading to more duplicated messages. Additionally there is the problem of 'resource blindness', where energy resources of nodes are not taken into account [54][2]. Flooding is generally not very scaleable and will end up sending a lot of messages, becoming inefficient. Flooding could also be used in a many-to-one or a many-to-many manner by simply repeating the flood for each node, however this would lead to an incredibly large number of messages sent, potentially leading to many collisions and is not practical. Additionally the routing protocol is not creating a route between multiple sources and multiple sinks, but instead repeating the same routing protocol multiple times.

Another classic routing technique is gossiping [53]. This technique is similar to flooding, however instead of broadcasting too all neighbours a single neighbour is randomly selected to forward a message to. This avoids the inefficiency involved in flooding, however gossiping may take a long time to deliver messages to the desired destination.

Sensor Protocols for Information via Negotiation (SPIN) [54] is a set of protocols that improves upon flooding as a routing technique. SPIN is based on the concept of data-centric routing, where nodes broadcast an advertisement message of the type of data it has access to. A sink may respond to this broadcast, initiating the process of routing from the sensing node. To improve upon flooding, SPIN introduces three types of message; ADV, REQ, and DATA. A node will first send a ADV message that contains meta data about the message. If a neighbour would like to receive the message, it sends back a REQ message, and the original node will then send the DATA message containing the sensed data. SPIN has the advantage that only relevant data is sent between nodes; there are fewer problems with implosions or overlap. There is also some level of resource awareness, as SPIN-2 has nodes only participating in the three stage message sending protocol if it has sufficient energy to do so. SPIN is limited in that to deliver multiple messages to multiple sinks, it will need to developed multiple different routes throughout the network. Additionally there is no guarantee that a particular path will exist towards a sink, as nodes may not request the data.

### 2.1.2 One-to-Many Routing

Flooding, as mentioned in the previous section, could also be used in a One-to-Many manner. If a source floods the entire network, it will likely be able to deliver to multiple sinks from a single source. This, though likely inefficient, is a simple way of delivering from one source to multiple sinks.

### 2.1.3 Many-to-One Routing

Directed Diffusion (DD) [59] is a data-centric routing protocol that is widely studied and from which many other routing protocols are based on. In Directed Diffusion, a sink node sends out an "interest" too all nodes in the form of attribute-values pairs, which has the effect of requesting data. The interest is propagated throughout the network whilst also initiating "gradients". When a source node has the requested data for a sink, it will send that data to follow the gradient path towards the sink. Several paths may be created from the source to the sink, and so the sink will send further interest messages out in order to reinforce better routes. In this way better paths will be followed often, and worse paths followed less. An example of this process is shown in figure 2.1. Directed Diffusion has the ability to recover from failures in the network by reinforcing a different path, should a failure occur on the one currently being used. An advantage of Directed Diffusion is that all decisions are made locally by the nodes, with no need for maintaining a centralised view of the network topology. As it is an on demand protocol, meaning that the sinks only request data when required, the protocol is able to conserve energy by reducing the number of unnecessary messages sent. This is different to SPIN where all source nodes will advertise the data they have at all times. The method used by Directed Diffusion, where a sink makes a request to the network for a specific piece of data, limits the protocols effectiveness in scenarios where continuous data is required. An example of this kind of application is habitat monitoring, where the same piece of data may be required to be delivered to

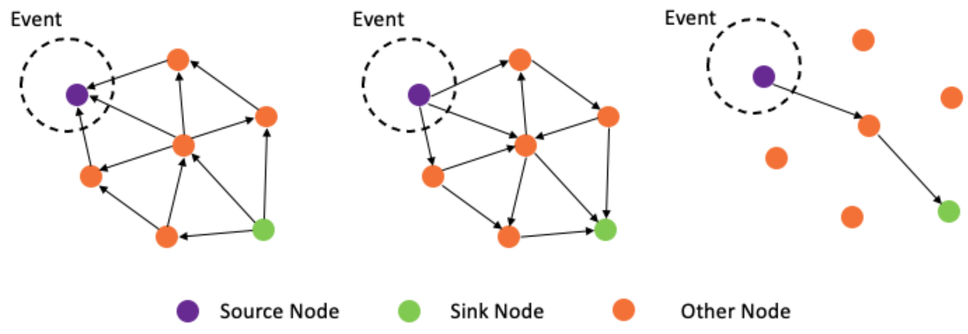


Figure 2.1: Example process of Directed Diffusion

the sinks constantly. Directed Diffusion is able to deliver messages to multiple sinks, however this is through the setup of multiple paths throughout the network leading to each sink, a setup which may lead to more messages sent than necessary if aggregation took place. The refreshing of paths with flooding of interests from the sink could affect scaelability of the protocol, requiring a large amount of messages to be sent to maintain routes. There may also be some inefficiencies as alternate routes must be maintained at a low rate if a different path needs to be used in the case of node failures.

Rumor Routing [13] is a variation on Directed Diffusion [59]. In rumor routing, if a sink requires data, it routes its request to the sensor nodes that have observed that data, as opposed to DD where the sink floods the network with an interest. When a node observes a sensing event, Rumor Routing uses long lived agents that propagate information about the event to nodes, which then store information about the event in tables. When a node then requests data on that event, nodes in the network will be able to know where to forward the request to in the network. Unlike Directed Diffusion, Rumor Routing only maintains one path from source to sink. There is an overhead in the storing of information in local tables on every node, and as well as maintaining agents for each event throughout the network. This could mean that the protocol is suited best to scenarios where the number of events or sources is small, in order to avoid large amounts of overhead.

Energy Aware Routing aims to increase overall network lifetime by storing a set of good, but not necessarily optimal, paths from sources to sink [110]. The paths are found in a similar method to Directed Diffusion, with initial flooding to set up multiple paths and routing tables, however in this case an energy cost is maintained and high cost paths discarded. Routing tables are used to choose the path from source the sink, with each node being assigned a probability based on the cost of travelling to it. Infrequent flooding is required to maintain paths. Storing a number of routes can involve a large amount of data stored on the memory of nodes, in addition to the flooding, limits the scaleability of the protocol.

Gradient Based Routing [108] is another protocol based on the concepts presented in Directed Diffusion with a focus on energy efficiency. The goal of this scheme is to enable the longest possible network lifetime. The protocol starts with flooding interests through the network to set up gradients. The gradients here are based on the difference between the nodes "height" and the height of its neighbour, with the height being the number of hops to the sink. Following this, techniques are applied to reduce energy consumption. "Data Combining Entities" combines data from different nodes, reducing the amount of data needing to be sent in the packet header, increasing efficiency. A series of data spreading schemes are also presented with the goal of distributed traffic among different nodes, balancing the overall load to the network across different paths. Three schemes are described, a stochastic scheme where a random next hop is chosen when the gradient is the same, an energy based scheme that discourages choosing nodes below and energy threshold, and a stream based scheme which discourages choosing nodes that are already in another stream. These techniques for reducing energy consumption are effective at increasing network lifetime overall, however has some limitations in terms of network attributes. For instance, the work assumes that all network nodes are of equal importance, which may not always be the case. Additionally the limitations of Directed Diffusion apply, such as it not being well suited to continuous data delivery.

Hierachical protocols use a clustering based model in order to route from sources to sink. In general, this involves a number of sensors being designated as cluster heads that perform some data aggregation, with other sensors being assigned to send messages to a particular cluster head. It is arguable that this form of routing could be many-to-many, as there are multiple sources and multiple sinks in the form of cluster heads. However, each source will only ever deliver data to a single sink, which may then coordinate with the other cluster heads, sometimes to send aggregated data to a single sink or basestation. This review will generally consider this to be many-to-one communication, as a source node will only ever need to route to one sink, however it could be scenario dependent. In general, this style of routing has good scaleability but at the cost of increased overhead in terms of cluster setup and updates [83].

Low Energy Adaptive Clustering Hierarchy (LEACH) is a highly referenced protocol that makes use of the clustering method [51]. In LEACH, clusters are formed based on signal strength, and local cluster heads are selected within each cluster to forward messages to the sink. No global knowledge of the network is required for the protocol, however the overhead involved in repeatedly selecting cluster heads may be high. The cluster heads will aggregate the data it receives from the other nodes in the cluster before forwarding to the sink, in order to reduce energy consumption by sending multiple messages. Cluster heads are routinely rotated such that energy consumption of nodes is balanced. LEACH is dependent on each node in a cluster being within a one hop distance to the cluster head, making it unsuitable for sensor networks where this is not possible, for instance those over a large area.

Power-efficient GAthering in Sensor Information Systems (PEGASIS) is a protocol whereby nodes organise themselves into chains [74]. Each chain will have a cluster head node assigned, and this node will forward messages to the sink. The nodes on the chain will communicate with one neighbour towards the cluster head in order to minimise energy useage. PEGASIS relies on each node having global knowledge of the network to form the chain, with the furthest node from the sink starting the chain before nodes are added in a greedy manner to the chain. During operation, data is aggregated along the chain as it is gathered before being sent to the next node along in the chain. The nodes take turns to broadcast to the sink in order to reduce energy consumption. In case of node failures, the chain will reconstruct around the failed node. PEGASIS represents an improvement on LEACH in terms of energy savings, with nodes running out of energy later in compared with LEACH. The protocol is limited in its requirement that each node has global knowledge of the network, and also requires all nodes along the chain to send a message in each round, which may not be necessary. Nodes further from the cluster head will lead to a longer delay in sending messages to sink, as it must travel along the entire chain. There is also the requirement for all nodes to be within a one hop distance to the sink, which may not be possible in many scenarios.

Location based routing protocols rely upon the locations of each sensor node being known, for instance to calculate distance between nodes. Locations may be provided using Global Positioning System (GPS), or through exchanging information with neighbours [109]. This requirement will limit the potential applications such networks can be used in, as location information may not be possible. However, this does enable efficient routing in the sensor network. An example of a location based routing protocol is Minimum Energy Communication Network (MECN) [105], which makes use of a low power GPS to generate an optimal spanning tree routed at the sink containing paths between source and sink that consume the least power, which is referred to as the minimum power topology of the network. The distributed protocol involves each node broadcasting its location information in order to build a graph of communication links between nodes such that the graph represents the minimal power consumption to the sink. The protocol then uses the distributed Bellman-Ford

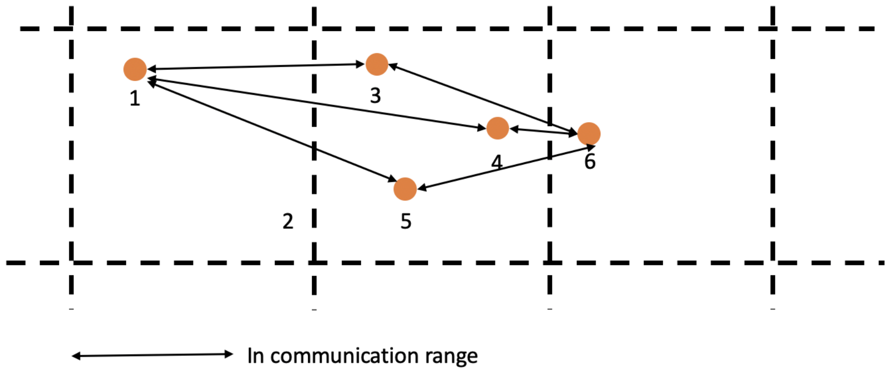


Figure 2.2: Example grid formed by GAF

shortest path algorithm [80] with power consumption as the cost metric to find the most efficient path to the sink. The protocol assumes every node in the network can communicate with each other, which may not always be possible. MECN is then extended to SMECN (Small Minimum Energy Communication Network) [72], which is able to create a smaller minimum power topology and is able to deal with obstacles in the network, however with higher overhead.

An example of an energy aware location based routing protocol is Geographical Adaptive Fidelity (GAF) [127]. GAF generates a virtual grid of nodes based on location data and aims to reduce energy consumption by keeping a number of nodes in the same grid square asleep at all times, and alternating which node in a square is awake. Using their location awareness, nodes will associate themselves with a virtual grid, and then collaborate within the grid zones to elect a master node that is awake and communicates with the sink. Other nodes in the grid zone will then sleep in order to save energy. The protocol may be limited in that each node in a grid is considered to be equivalent in routing costs, which may not be the case. An example of the type of grid formed by GAF is shown in figure 2.2, where nodes 3, 4, and 5 are equivalent in which nodes they can communicate with. This means that two of the three can be asleep while maintaining the same connectivity.

Geographic and Energy Aware Routing (GEAR) [132] is a location aware routing protocol that is also based on the concepts introduced in Directed Diffusion. Interests are disseminated throughout the network directed towards a targeted location in the network rather than flooding the whole network as in Directed Diffusion. Each node is assumed to know its location with GPS, in addition to its energy level, as well as its neighbours location and energy level. Using this knowledge, interests are directed through the network by nodes working out if a neighbour is closer to the target location than itself. If all neighbours are further away, a node is chosen to minimise estimated cost. Once the interest reaches the targeted region, data is diffused through the network using either recursive geographic forwarding (recursively flooding regions) or restricted flooding.

Another common method of routing involves setting up trees in the network. The Collection Tree Protocol (CTP) builds trees routed at the sink, where data from a source travels up the tree to reach that sink [46]. CTP uses a cost metric called expected transmissions (ETX), where ETX is the expected number of transmissions required to send a message to the root of the tree. ETX is used by nodes to select the next node to send a message to upwards through the tree, with each node selecting the next node based on lowest ETX. ETX as a metric has advantages in that is should lead to short routes with smaller number of packets sent, but also comes with some limitations. To maintain accurate ETX figures, nodes will periodically broadcast its ETX value to is neighbours, an expensive process. There are also likely scenarios where it is not possible for nodes to accurately report ETX, as this could reasonably require some level of network wide knowledge. CTP could be used to route to multiple sinks by repeating the algorithm to construct multiple trees rooted at each sink. This however may lead to each node having to store a large amount of routing information, as well as inefficient routes when a source needs to send the same data to multiple sinks. Additionally, a node will not choose a particular sink to travel to, just its closest sink, which may limit potential applications.

### 2.1.4 Many-to-Many Routing

Wireless Sensor Networks with multiple sinks have a number of advantages. Often, the nodes surrounding a sink will use energy more quickly than other nodes in the networks, as they will be required to route to that sink. Multiple sinks may lead to some load balancing network wide, with more nodes taking the strain to routing to sinks. Also, multiple sinks are useful in applications such as actuator networks, or in very large networks where it is helpful to have many sinks covering the area. In terms of fault tolerance where it is often helpful to have multiple sinks in case of sink failure, or sensor node failures that lead to partitioning of the network. For this reason, many-to-many-routing is an advantageous area to investigate, though solutions are not as common as many-to-route routing.

An example of a hierarchical routing protocol that can be considered to be many-to-many is Two-Tier Data Dissemination (TTDD) [78], an approach that aims to deliver data to multiple mobile sinks. Source nodes build a grid of nodes preemptively, with dissemination points being set as the nodes at the crossing points of the grid, and nodes closest to these points being assigned dissemination nodes using recursively data announcements. The process of creating the grid does not rely on each node having global topology knowledge only localised information. Each source will build different grids, and so may have different sets of dissemination nodes associated with their grids, which has the potential to lead to overlapping grids. When a sink requires data, it will flood a query in a grid cell sizes area until it reaches a dissemination node, which then forwards the message via dissemination nodes until it reaches a source. This method has the advantage of only needing to flood a small area, as well as the potential to aggregate multiple queries within the cell. On receiving a query, the sources forwards data via the dissemination nodes back through the network to the sink. If there are multiple sinks, it is possible to forward this message to all of them. TTDD relies on the sensor nodes being location aware, i.e. they know their own location.

Multisource Multisink Trees for Energy-Efficient Routing (MUSTER) is a distributed many-to-many routing protocol that merges independently built trees from source to sink [89]. The result of this process is a shared path, or backbone, between the merged trees. This backbone will split at the last possible node in order to send packets to both sinks. The trees are built with a “flooding-and-reverse-path” scheme, MUSTER deploys a load balancing mechanism that relocates this backbone in order to reduce overall energy consumption and increase network lifetime. MUSTER is distributed and used in many-to-many networks with multiple sources and multiple sinks. A requirement of this protocol is periodic network wide broadcasts to refresh the trees that are built to account for topology changes, which will increase the over all number of packets sent and lead to decreased lifetime, as well as reducing the scaleability of the protocol.

In [95], the problem of sink placement in a cluster based network is investigated. Three problems are considered; the problem of Best Sink Locations where number of sinks is known and it is required to find where to place them, the problem of minimising the number of sinks for a given operational period, and the problem of minimising the number of sinks whilst increasing network lifetime.

Multipath Routing in large scale sensor networks with Multiple Sink nodes (MRMS) [18] finds routes in networks with multiple sinks and can be divided in three parts; topology discovery, cluster maintenance, and path switching. Topology discovery is based on the TopDisk algorithm [22], a greedy algorithm for finding the set cover, and once this has completed the network is divided into clusters, though it is not made clear how this occurs. Cluster heads are updated if the energy of the head node falls below a threshold value by sending a message out to its neighbours, which then report their energy levels to the head so that it can inform a node to be a new head. Path switching occurs when a path to a sink has been used for a long time, leading to a path to another sink being used instead. When clusters decide a new cluster head, it is also decided whether path switching should occur. The protocol may be limited in that there is overhead in the process of cluster maintenance and path switching. Also the protocol does not account for scenarios where multiple sinks require to receive the data simultaneously and only uses multiple sinks as a fault tolerance measure.

## 2.2 Meta Heuristics

Routing protocols for many-to-one topologies are much researched, however many-to-many routing protocols are far less common, despite there many possible applications and advantages. Many algorithms require a specific set of network conditions for an efficient solution for routing, for instance many require global knowledge of the network, or location based methods such as GPS. For this reason, a more generic perspective is required in order to develop a routing protocol that is flexible, and able to route from multiple sources to multiple sinks in a wide variety of situation with little if any modifications. Meta heuristics present great opportunities in this area, as they are adaptable to a number of starting conditions while still being able to efficiently come up with solutions. This section explores some existing solutions that use meta heuristics in WSN, particularly Ant Colony Optimisation [31] [32].

### 2.2.1 Ant Colony Optimsation

Ant Colony Optimisation (ACO) was originally defined by M. Dorigo, V. Maniezzo, and A. Colorni [31][32]. The heuristic approach can be used to provide near optimal solutions to NP-hard problems, with the advantage of being versatile and robust. The algorithm is based upon the observed behaviours of ants in nature as they travel towards food sources. Ants drop

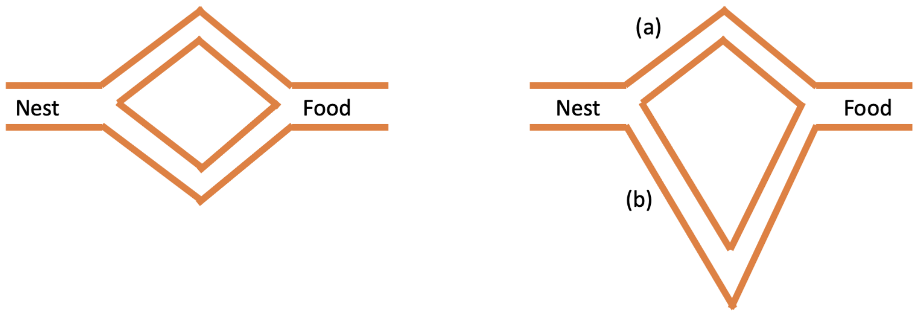


Figure 2.3: Ant bridge experiment

a pheromone trail as they travel, which subsequent ants will follow, and so adding further to the pheromone trail. The pheromone trail will evaporate over time, which means that shorter, more successful routes will eventually be preferred, as the trail on shorter routes receives pheromone on the whole path quicker, enabling the pheromone to build up quicker before it evaporates. Goss et al [47] shows this with an experiment called the double bridge experiment, illustrated in figure2.3. If the bridges where the same length, then over time the ants tended towards a single bridge due to random fluctuations leading to more ants choosing that bridge at a particular point in time. If one bridge is short, in this case bridge a, the ants will converge upon that bridge. This is because the pheromone trail laid by the ants will build up more quickly compared with the longer route, as more ants could complete the route before evaporation completes.

ACO has been applied to many types of problems, such as the travelling salesman problem [33] [30] [116], vehicle routing [41] [102], graph colouring [20], and set covering [71]. In the case of finding solutions to the Travelling Salesman Problem, Ant Colony Optimisation as first presented by Dorigo et al. would be applied by simulating ants moving throughout a graph. The ants move between vertices, and the pheromone trail is represented by a value associated with each edge. Ants will travel between nodes, choosing the next node stochastically favouring nodes with higher pheromone levels and not repeating nodes. Once the ants have finished finding a path, the quality of the solutions found are evaluated and the pheromone values are updated according to the quality of these solutions. This process will be repeated until some end condition is met, iteratively improving upon the solutions found. The ACO meta heuristic is defined by Dorigo et al. in [29] and can be summarised by Algorithm 1, presented in [32].

**Algorithm 1** Description of ACO Metaheuristic

```
1: procedure ACO-METAHEURISTIC
2:     Set parameters and initialise pheromone trails
3:     while termination condition not met do
4:         Construct Ant Solutions
5:         Apply Local Search (optimal)
6:         Update Pheromones
```

The meta heuristic repeatedly carries out three steps:

- **Construct Ant Solutions**: A set of ants generates solutions to a optimisation problem. Each solution starts as an empty partial solution, which is extended at each step by adding a component, for instance adding a vertex in the graph.
- **Apply Local Search**: An optional step that may improve paths found by ants before updating pheromone values.
- **Update Pheromones**: Update pheromone values such that good solutions have increased pheromone and bad solutions have decreased pheromone, usually through pheromone evaporation.

Ant System is the first algorithm that uses the concept of Ant Colony Optimisation presented in literature [31], and is the basis of many future works. The implementation is described in this section, with all equations coming from this seminal work. Let $\tau_{i,j}(t)$ be the intensity of pheromone trail on the edge between node $i$ and node $j$ at time $t$. Every ant in the network will choose the next node to travel to by time $t+1$, referred to as an iteration of the algorithm, or a cycle. After sufficient iterations, in the case of travelling salesman this will be equal to the number of nodes, the ants have completed their solutions and will updated the pheromone trail with

$$\tau_{i,j}(t+n) = \rho \cdot \tau_{i,j}(t) + \Delta\tau_{i,j} \tag{2.1}$$

where $\rho$ is the evaporation coefficient. $\Delta\tau_{i,j}$ is calculated using:

$$\Delta\tau_{i,j} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \tag{2.2}$$

where m is the number of ants and $\Delta\tau_{i,j}^{k}$ is

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if kth ant uses edge (i,j) in its solution between time t and (t + n)} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

where $Q$ is a predetermined constant and $L_k$ is the length of the solution found by the $k$th ant. In the Ant System algorithm, each ant keeps track of the solution found so far using a data structure known as the tabu list. The tabu list is a list of partial solutions, and further solutions may not contain elements of the tabu list. In this case, the tabu list saves the nodes visited so far in order to both avoid repeat visits, which each ant carrying a list of nodes that it has visited, so that no two solutions are the same. It is also used to evaluate the solution found.

At each iteration of the algorithm, a set of $m$ ants choose a node to travel to with probability

$$p_{i,j}^{k}(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^{\alpha} \cdot \eta_{i,j}]^{\beta}}{\sum\limits_{k \in allowed_k} [\tau_{i,j}(t)]^{\alpha} \cdot \eta_{i,j}]^{\beta}} & if j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where $allowed_k$ is the set of nodes not contained within the tabu list of the ant, $\eta$ is the visibility, a value denoted by $1/d_{i,j}$ where $d_{i,j}$ is the distance between node $i$ and node $j$, and $\alpha, \beta$ are weights denoting the important of pheromone trail and visibility.

Ant System has been shown to be both versatile and robust, and has the potential to be applied to a number of different problems. A number of works have been produced based upon the concepts introduced in Ant System.

MAX-MIN Ant System (MMAS) [115] is an ACO based algorithm that is an improvement on Ant System. In MMAS, pheromone updates are limited to only the ant that has performed the best each round. The best ant is often considered to be the ant that has travelled the shortest route. Also, the pheromone amounts have upper and lower limits, usually found empirically.

Another variation of the ACO algorithm is Ant Colony System (ACS) [40], which introduces the concept of the “local pheromone update”. In the ACS algorithm, all ants update the pheromone trail along the edge it just travelled along decreasing the pheromone. This has the effect of encouraging other ants to choose different edges, leading to a more diverse set of solutions found. A pheromone update is also performed at the end of each round, and similarly to MMAS only one ant will perform the update; either the best ant that round or the best ant so far, with the best ant being the ant that has travelled the shortest route.

### ACO in Networks

Though many of the routing algorithms discussed in the previous sections are successful, many of them depend on a particular set of starting conditions in order to achieve good performance. For instance, many are not distributed, and require some level of global knowledge of the network. Often they will only work with a singular sink, and to be applied in a network in multiple sinks they are simply repeated, a likely expensive process. In order to apply to a large range of potential networks, meta heuristics such as ACO are an interesting path of research. Due to its suitability to solving shortest path problems, finding solutions to NP-hard problems, and the potential of developing distributed version of the meta heuristic, ACO has been applied in the context of both traditional wired networks and wireless sensor networks in previous works. Examples of these solutions are presented here.

An early example of the use of ACO in networks is AntNet [24], a distributed protocol for routing in communications networks. The protocol builds routing tables for each node, which are used to direct messages towards the next node in the network. The entries of the routing tables contain probabilistic values used to direct ants, similar to information provided by equation 2.4. Ants are periodically launched from source nodes towards destination nodes, choosing nodes to travel to using the routing tables such that the path is minimal cost. Once at the destination, the ants travel back to the sources in the reverse of the original path updating routing tables, known as backward ants.

AntNet was developed for use in wired networks, however other works have adapted it for use in wireless sensor networks. Zhang et al. extends this work to apply the protocol to wireless networks, as well as introducing three variations upon this adaptation [134]. It was found that the base version of the algorithm was not very successful, as the ants had no idea where the sinks were and relied only on pheromone trail. To deal with this, the first variation on the base protocol is Sensor-Driven Cost-Aware Ant Routing, where each ant is aware of the cost from each node to the source and uses it to choose where to travel to. Flooded Forward Ant Routing is the second variation, and involves close neighbours of the source also launching ants towards the sink. Flooded Piggybacked Ant Routing has the ants also carrying data to the sink, in addition to finding shortest paths. These variations on the AntNet routing protocol do not take into account multiple sinks.

In [15], Camilo et al. apply ACO to wireless sensor networks with the goal of finding short paths from source to sink, while also minimising the amount of energy expended. The protocol, Energy-Efficient Ant-Based Routing Algorithm

(EEABR), periodically launches ants from every node in the network. The ants choose the next node to travel through with probability based on 2.4 but the visibility is now related to energy level remaining on the sensor nodes. In a similar fashion to AntNet and [134], on receiving a forward ant, the destination node launches a backward ant to travel back through the network updating pheromone. This process runs for a number of iterations before each node will be aware of the best node to travel to next by storing a routing table. The work is limited in that it requires each node to have knowledge of the energy levels of its neighbours, which will change over time, as well as a potentially large setup time as ants are not aware of how close the sinks are. Though efforts have been made to minimise the amount of data required to be stored in routing tables, these still may scale large with many sources, many sinks or larger networks.

ACO-based quality-of-service routing (ACO-QoSR) [14] is a protocol that aims to find a route that satisfies time delay constrains while conserving energy. This protocol also makes used of routing tables for nodes to decide where to forwarded to next. The visibility value here is the proportion of energy of the potential next node to the total energy remaining of all the neighbouring nodes. Backward ants are also used here, with pheromone values being updated according to residual energy and hop count of the route. Entries in the routing table will expire after a set time, and be removed. Periodic HELLO messages are sent by all nodes, and if a link has failed the pheromone value is set to 0. A technique is developed to avoid stagnation; pheromone limiting sets a maximum pheromone amount, prevent one path from becoming too dominant. The protocol may be limited in its scaleability through its use of routing tables, and also is only targeted at networks with a single sink.

An example of a centralised ant based routing protocol is put forward with the AntChain algorithm [26]. Assuming that every node can communicate with the sink, AntChain uses ACO to form a chain from sensors to the sink, similar to LEACH and PEGASIS. It also assumes that all nodes are able to communicate with the sink and are location aware, however, unlike PEGASIS, does not require each node to have prior global knowledge of topology. In AntChain, all sensor nodes send their location information to the base station (sink), which then performs the MMAS algorithm to form the chain structure. The head of the chain, furthest from the base station, starts data gathering and sending the data to its neighbour. The data is passed along the chain being aggregated as it travels, with the head of the chain, the node closest to the base station, finally sending the data to the sink. A bidirectional variant is also described to deal with failures, where the direction of travel along the chain is alternated, and if a node does not receive data from a neighbour it sends directly to the base station instead. The centralised aspect of AntChain, as well as the requirement for all nodes to be within range of the sink, will greatly limit its possible applications.

In [67] a clustering based protocol for data gathering that makes use of ACO is presented. The protocol aims to improve reliability by introducing multiple sinks, with nodes using a different sink in the case of failure of the original sink. The network is divided into clusters, each associated with a sink, with nodes sending messages to the sink in its cluster. The clusters formed dynamically change in size using ACO, in order to adapt to failures in the network. In this protocol, sink nodes start ACO by flooding backward ants through the network to establish pheromone values between nodes in tables, with pheromone depending on residual energy levels of nodes along the path. Pheromone tables are periodically updated with the broadcast of more backward ants through the network, known as "hello" ants. To allow ants to build clusters, a concept called "cluster pheromone" is introduced, which is representative of the attractiveness of joining that cluster. The hello ants are then used to build the clusters. Failures are detected through the use of an expiry time; if no hello ants are received in the time, the node is removed from neighbour tables. The protocol may be limited by the amount of "hello" ants that need to be sent as they are used for both network formation and fault detection. Additionally, upon failure detection, the node is removed completely from tables, possibly making the protocol vulnerable to false positives.

Singh et al. present a method for using ACO in wireless sensor networks for computing minimum Steiner trees [113]. First, an offline centralised method is described, then an online distributed version. The algorithm forms a minimum Steiner tree that is rooted at the sink, and this tree can then be used for routing from sources to the sink. The tree is formed when two ants launched from source nodes towards the sink meet and combine to form a single ant. The testing of the algorithm is focused on a single sink node, though it claims to be easily adapted to multiple sinks. An issue with the algorithm is that it could be difficult to implement in practice as it is susceptible to node failures and topology changes; if an ant is lost, the tree may not form correctly. Additionally, applying the same algorithm multiple times for each sink is not scaleable for larger networks with many sinks.

### 2.2.2 Bee Colony Algorithms

The Artificial Bee Colony algorithm (ABC) [62] has been developed based on the foraging behaviour of bees in a honey bee swarm. The algorithm consists of three types of bee; employed bees go to food sources, onlookers are bees waiting to make a decision, and scouts go on random searches. Employed bees and onlooker bees are repeatedly placed on the food sources, while scouts search the area for new food sources. Bees gather information about the "nectar" amount on the food sources in order to decide where to go. Bees will switch between being onlooker bees and employed bees as food sources are found.

Bee based algorithms have been applied in the context of routing in wireless sensor networks. BeeSensor [107] is an energy aware routing protocol inspired by bees. It uses different types of bees to perform different roles. Forward scouting bees are launched from the sources and stochastically broadcasted by nodes as they receive them, with nodes dropping scouts if they have already received a copy. Sinks interested in events launch backward scouting ants to the source. When the scouts have found a route, foraging bees are then sent to deliver data to the sinks. BeeSensor aims to maintain a single path to avoid overhead, however for continuous data delivery multiple paths can be maintained with additional overhead.

A protocol for cluster based routing in WSNs using an artificial bee colony algorithm is presented in [63]. ABC performs the cluster head selection, with employed bees being associated with each head. The protocol is limited by the requirement for a centralised base station that runs the algorithm.

### 2.2.3 Other Particle Swarm Algorithms

A scheme for sensor deployment based on Glowworm Swarm Optimisation (GSO) is presented in [73]. In the scheme, sensor nodes are considered as glow-worms emitting a luminescent substance, with the intensity being proportional to the distance between the node and its neighbours. Sensors are attracted to areas with less brightness, leading to a well covered network of sensor nodes.

## 2.3 Fault Tolerance in Wireless Sensor Networks

Fault tolerance in wireless sensor networks is the ability for a network to continue operation when faults occur. The nature of WSN means that they are often unpredictable in terms of faults, being deployed in areas where node failures are both possible and common [135]. These node failures could come in the form of nodes running out of energy, being moved out of range of other nodes through environmental factors such as wind, or being destroyed by adverse environmental conditions such as fire or extreme temperatures. Links between nodes in particular may fail if the environment changes around them, leading to links being blocked. Any of these scenarios or more could lead to the failure of delivery of sensing data to the network sinks. Often in networks where multihop communication is required, the effects of node failure is exacerbated as recovery is more difficult when a new route from source to sink needs to be discovered [96].

In cases of node failure, the remaining nodes in the network will often need to detect that a failure has occurred before taking actions to ensure that data is still routed through the network to the sinks without compromising efficiency or significantly decreasing network lifetime even further. WSN bring specific challenges due to their often distributed nature, the requirement for energy efficiency, and the fact that failures are common. This section investigates existing solutions to provide fault tolerance in wireless sensor networks. It is largely divided into three main concepts of fault tolerance, fault prevention, fault detection, and fault recovery, though most fault tolerant schemes will use a combination of these methods in order to provide protection against faults.

### 2.3.1 Fault Prevention and Robustness

Fault prevention and robustness in wireless sensor networks is the concept of either preventing failures from occurring, or setting up protocols such that they are largely unaffected by failures that occur. Fault prevention in wireless sensor networks tend to be provided through one of three main mechanisms [96]:

- Providing full network connectivity and/or full coverage at deployment.
- Monitoring network in order to take action to prevent failures.
- Redundancy through building multiple paths (or multipath)

A Coverage Configuration Protocol (CCP) is presented in [121], which aims to provide minimum coverage and connectivity in the network in order to provide robustness against node failures. The protocol only requires each node to have localised knowledge, not global topology information, in order to self configure. Another coverage calculation is presented in [85], where both best and worst case coverage are found. Paths are generated to either maximise or minimise the distance to the closest sensor, with the best case scenario minimising distance. However, this method may not be possible for all network setups where full coverage is not an option.

A technique for providing both coverage and connectivity in WSN using the least number of sensors is shown in [60]. The minimum number of nodes required to provide a good chance of coverage is found, and also a incremental deployment scheme is presented where random sampling without replacement is used.

The work in [136] is a network wide scan that reports on approximate energy levels of each node using in-network aggregation. Each node does a local scan of its energy level, with aggregation occurring if local nodes have similar levels of energy. This process enables a user of the network a view of the remaining energy in order to make preventative actions. Another work that attempts to create a map of residual energy is presented in [88], however this uses a prediction based approach. Nodes will send limited information and a predictive model is used to create an energy map. This method means that there is less overhead in monitoring energy, however this comes at the expense at less accurate reporting. Fault prevention based on monitoring network health may be limited, as it requires further action following identification of potential problem nodes, which may not be possible in inaccessible environments.

Multipath routing can be used for both fault prevention and fault recovery. It is the technique of creating multiple paths between sources and sinks, in order to provide redundancy in the network. For the case of fault prevention, multipath routing has the advantage of load balancing; nodes share the load of routing data leading to energy be spent more evenly throughout the network, meaning that each node uses less energy. The GRAdient Broadcast [128] protocol builds a “mesh” of nodes that indicates the cost of routing from source to sink. Nodes only continue forwarding data if it has a lower cost than the node it received from. Using this method, multiple paths may be followed, leading to a protocol that is reliable and robust to node failures. There is also the concept of “credit”, a value assigned by the source, which widens the mesh and allows more routes to be taken, as long as the cost is allowed by some credit value. GRAB works well in dense networks, however may not scale well with multiple sinks. Multipath routing will be discussed further in section 2.3.3, as often alternative paths are only used in the case of node failure as a reactive method, as opposed to being used to prevent errors from occurring.

### 2.3.2 Fault Detection

Fault detection is essential in many applications of WSNs, due to the adverse conditions they are placed in, and so a large variety of schemes have been developed [7][131]. Fault detection is the process of finding where faults have occurred in the network, for example nodes ascertaining if a neighbour is no longer reachable. Fault detection comes in many forms and varying levels of complexity, and are explored in this section[17].

Many existing protocols utilise a centralised approach to fault detection. Often this involves a central controller or sink detecting faults for the entire network. [100] is an example of this, taking a centralised approach with a sink gathering and analysing distributed data in order to detect faults. The sink must continually monitor network traffic in order to detect changes in expected traffic that could indicate a fault, which limits its use in networks with a single powerful sink node. In [114], the sink has knowledge of the topology of the network, enabling it to deal with failures using normal routing messages. The sink node then makes routing changes in order to trace where faults occur, with a number of different tracing schemes described in order to locate faults.

Centralised approaches often lead to successful failure detection and recovery, however this often comes at the cost of increased overhead in both messages sent and initial setup costs to learn the topology. Additionally, for many networks a centralised approach is not feasible, for instance where there are multiple sinks, or where a powerful central node does not exist in the network. This approach also does not scale well with larger networks, due to the increasing message and computational costs. Distributed approaches to fault detection involve nodes making local decisions to deal with faults in the network, often requiring fewer overall messages sent at some cost to other performance metrics such as recovery time. A number of approaches have been developed, which will be explored here.

In [25], a protocol is presented with the aim to identify faulty sensors locally in order to keep overhead low. Faults are detected by finding sensors with readings that differ largely with that of its neighbours. The algorithm works well for large networks but accuracy decreases when there are a large amount of faults. [126] also provides localised fault detection but is an improvement upon [25] as it works well with a large amount of faults. Good sensors are chosen within an area, and results are compared to these sensors in order to find faulty ones. There are some issues with scaleability due to the number of messages sent between neighbours.

[84] provides a watchdog that identifies failing nodes and a "pathrater" that routes around them. The watchdog keeps track of sent packets, comparing them with packets it overhears being sent from other nodes. If there are inconsistencies between sent and heard packets, for instance if the packet is never resent to another node, it is assumed a failure has occurred. Disadvantages of this technique are they it may miss failing nodes due to collisions, and it also has a lot of overhead as each node must monitor its neighbours.

[118] implements a heartbeat-style failure detection scheme where nodes within a cluster periodically broadcast a "heartbeat" message to the cluster head. In addition to these heartbeat messages, nodes also send messages containing all the nodes that they have received heartbeat messages from. The cluster head determines which nodes have failed using these messages and a failure rule.

This may lead to high overheard in terms of messages sent. [55] has each node actively monitor its neighbours in order to detect faults. [19] is an example of passive neighbour monitoring, where it is assumed that faulty nodes send data inconsistent with other nodes.

Directed Diffusion [58] is able to detect faults that occur along the routes from sources and sinks. Nodes expect a certain rate of reporting from its neighbours, and when this lowers unexpected it assumes a link has degraded. This initialises a fault recovery mechanism where alternative routes are reinforced.

The work in [114] traces the location of the failed nodes by sending network topology information to the base station. Nodes send information about their local neighbours to the base station so that the base station knows the network topology. Tracing of faulty nodes is then performed, with two methods described. In the first, the base station broadcasts a route update to the network, and is able to work out which nodes are faulty by waiting for measurements to be sent back. In the second, the network is subdivided in order to send route updates. The method of fault tolerance presented here relies upon specific network circumstances, a power base station, and routing updates being periodically sent, and so may make it not applicable to many scenarios.

### 2.3.3 Fault Recovery in WSN

Following the detection of a network fault, the role of a fault recovery protocol is to handle that fault. Early work often assumed constant flooding of the network in order to route around node failures. Other simple fault recovery mechanisms may just involve rerunning the whole route discovery protocols with the failing nodes excluded [42] [96]. Techniques such as these are expensive, involving a large amount of messages sent decreasing overall network lifetime, and would become inefficient with multiple failures. Therefore, protocols are developed to recover from faults quickly and efficiently in a way that will extend network lifetime. Fault recovery comes in several forms, which can be categorised into retransmission based and replication based schemes, which will be discussed in this section.

### Retransmission

Retransmission based schemes often rely upon a sink node sending acknowledgement messages on receiving a message. If the sending node never receives an acknowledgement, it retransmits the message. Such schemes have the advantage of increasing reliability at the cost of higher resource consumption in the form of increased messages sent [7].

A fault tolerant scheme for wireless ad-hoc networks is presented in [84], consisting of both a "watchdog" and a "pathrater". The watchdog, previously discussed in the fault detection section, detects faulty nodes by listening to network traffic. A buffer of recently sent packets is maintained on the watchdog, and if a packet is received matching that in the buffer, it is removed from the buffer. When packet information remains in the buffer for a while, this may indicate a failure. The watchdog may not detect certain types of failure, for instance those due to certain collisions. The pathrater chooses routes to follow using reliability information to calculate a path rating. In the case of faulty nodes, the pathrater attempts to find a path without them; if this is not possible to routing algorithm is run again. This method of route tolerance therefore monitors for node failures and recalculates routes to follow. It may be limited in that recalculation of full routes could be expensive.

The fault tolerant mechanism used in Directed Diffusion [58] is based upon the principle of retransmission. The sink node continually sends reinforcement messages throughout the network in order to reinforce the best path. In the case of node failures, the sink node will no longer receive data, and so will decide to reinforce a different path. A balance must be struck with the frequency of reinforcement messages; higher frequency means a quicker response however higher overhead. The main drawback of this method of fault tolerance is the requirement of periodic flooding of events to maintain paths.

The work presented in [42] is based upon the concept of Directed Diffusion while introducing the additional concept of braided multipaths. A braided multipath is several possible paths between nodes with all paths having some nodes in common, allowing each path to be a relatively efficient route while also having the chance of avoiding faulty nodes. A "best" path according to some metric is found, with a number of alternate paths also being found ahead of time and maintained in order to recover from failure. This enables the network to quickly switch to an alternate path without the need for periodic flooding, though flooding may still be required if all of the previously found routes fail. An advantage of braided paths is that the overhead for maintaining the paths is lower, however the paths do still require reinforcement messages.

Reliable Energy Aware Routing (REAR) [52] is a distributed routing protocol where both a primary path and a disjoint backup path to the sink is setup whenever an interest is received from a source. If a node failure on the primary path occurs, the data packet is sent both back to the source and towards the sink, removing the routing information from routing tables switching to the backup path.

### Replication

Replication based schemes will generally send the same message more than once through the network from source to sink. Often these schemes come in the form of multipath schemes, where multiple paths are followed from sources to sinks. Disjoint multipaths will consist of multiple paths with no nodes in common, whereas braided multipaths will have some shared nodes between paths. Disjoint paths may have worse latency outcomes, however unlike braided multipaths will not fail if a particular common node between paths fail[7].

Reliable Information Forwarding (ReInForm) [21] sends multiple copies of the same message over many paths. It requires the sink node to periodically broadcast to all other nodes in the network update information about its neighbouring nodes and its hop count to the sink, which may lead to large overheads. The protocol has each node calculate a Dynamic Packet State, a figure based on reliability and hops to the sink, to decide how many packets to forward and where to send them until the packets reach the sink. ReInForm achieves fault tolerance through sending multiple copies of packets to the sink, assuming that at least one path will be successful. This protocol may have high overhead, and relies upon the sink node being powerful enough to send information to all other nodes.

Erasure coding is a technique that provides redundancy whereby data is split into $m$ fragments and then recoded into $n$ fragments where $n > m$ [122]. It has been used in multipath routing techniques to enable fault tolerance. The fragments are sent over multiple paths, with a minimum number of fragments that is less than the total number of fragments being required to fully reconstruct the data [7]. An example of the use of erasure coding in WSN is presented in [34], where fault tolerance is provided through the redundancy of sending packets along multiple paths while reducing the increase of traffic involved in traditional multipath routing.

The work in [27] also makes use of erasure coding, through the concept of "prongs"; nodes that are connected to the sink by reliable, high bandwidth links. Multiple prongs means there are multiple disjoint paths through which erasure coding can be applied. The work in [28] also uses a similar prong based method, also with erasure coding. Reed-Solomon erasure coding [103], a type of erasure coding with good overhead factors, with prongs is used in [6], with the number and size of fragments being adapted to network requirements. These methods that use prongs may be limited in the types of networks they can be used in, and are sensitive to failures in the prong nodes.

The N-to-1 Multipath Routing Protocol [123] finds multiple disjoint paths from every source to the sink and sends data to the sink when requested. Routes are found with a simple flooding technique that constructs a spanning tree, and an extended flooding method to create multiple paths. Source nodes split their sensed data between the multiple disjoint paths, which leads to the scheme to be able to deliver messages in the presence of faults. Additionally, nodes are able to make local decisions of next hop choices in order to deal with faults.

H-SPREAD [124] is a hybrid multipath scheme based on the N-to-1 path discovery protocol with the goal of improving both security and reliability of the network. A secret sharing scheme is used to split a secret message into multiple shares, which is sent sent to the sink via multiple paths, with the advantage of the data remaining secret if less than a threshold of shares are found. Both N-to-1 and H-SPREAD are applicable to networks with a single sink.

A protocol with the goal of providing Quality of Service (QoS) constraints is Multipath Multispeed Protocol (MMSPEED) [39]. The forms of QoS to be delivered are timeliness and reliability. To improve reliability, the QoS constraint most applicable to fault tolerance, MMSPEED makes use of probabilistic multipath forwarding, where multiple copies of the data is sent over many paths. Next hop decisions are made locally in order to improve scaleability, however the protocol does assume that the sensor nodes are location aware, and periodic location updates are sent to the neighbours which limits network lifetime.

### 2.3.4 Fault Tolerance with ACO in WSN

Though Ant Colony Optimisation has been implemented for WSNs with success in other works, the technique being applied with a focus on fault tolerance is less common. This is surprising as ACO naturally lends itself well to fault tolerance. The evaporation of pheromone trail over time should eventually lead to unsuccessful routes, such as those routes affected by node failures, being abandoned by the ants as there will eventually be no pheromone trail leading to such nodes. Additionally, it has been shown that ants in nature are able to divert around blocked routes [101], a process which could be considered analogous to a route failing due to a faulty node. This section considers the existing work that uses ACO in a fault tolerant context in wireless sensor networks.

ACO-based quality-of-service routing (ACO-QoSR) [14] has a simplistic level of fault tolerance. Nodes send out periodic HELLO messages, and if a link has potentially failed the pheromone trail is set to 0. This may be vulnerable to false positives, as it will discount the node entirely.

In [23], ACO is used to construct tours of the network ahead of time, before a Local Search Procedure improves upon routes found, and pheromone is then updated (the evaporation step). Here, it was found that using ACO, routes could be found that met with minimum levels of reliability with minimal deployment costs. In this work, reliability is provided by finding unconnected minimal covers of a region of interest in a sensor network. At any one time, a single cover will be used by the network to deliver messages to the sink, until a node failure occurs and the network switches to another cover. In this work all routes are found by ACO ahead of time, and it is not explicitly distributed. If nodes fail over multiple different covers, there is no online recovery of the network and the whole algorithm must be run again.

[86] uses ACO to create a minimum spanning tree in the network. Following this, a Quadratic Minimum Spanning Tree or Q-MST is found using the Artificial Bee Colony algorithm (ABC). If a node fails, edges are deleted and ABC is rerun. In this case, fault tolerance is provided using ABC to find substitute edges for failed edges rather than using ACO.

Another work that uses ACO to develop routes and manage faults is described in [38]. Here, ACO is used to develop multiple routes in the network in a similar manner as described in other protocols. Faults are detected when no acknowledgement is received from a receiving node. Following this, the pheromone value of that node is set to 0 and a new path is followed. If there are no other available paths, the route discovery phase is started again. This protocol relies more upon discovering multiple potential routes ahead of time, as opposed to an online algorithm. If there are no alternate routes available, there is the potential of running the more expensive process of rerunning route discovery completely. The protocol may also be sensitive with regards to acknowledgement messages, as if one fails the entire pheromone is set to zero, which means an alive node may be completely removed from the route unnecessarily. This could mean that the protocol is very sensitive to false positives, or temporary failures of nodes.

In [117], ACO is used to provide fault tolerance in Mobile Ad Hoc Networks (MANETs). In the initial route discovery phase, ACO is used to initialise pheromone values between nodes. Route selection then takes place, where a route is chosen based on the highest pheromone value. The fault tolerance of the protocol is reinforced through the use of QoS metrics; the sink node will calculate whether a path meets a QoS threshold, and only send backward ants to reinforce pheromone values along those paths according to how well the path met the QoS requirements. The protocol had packet delivery ratios of around 80% when 10% faulty nodes, with delivery ratio decreasing when the faulty nodes increased. The testing of the protocol is limited to at most 40 nodes, and to having a single sink node.

## 2.4 Steiner Trees

The Steiner Tree Problem (STP) in graphs is defined as follows; given a graph with a given set of points, also known as terminals, the Steiner Tree Problem is to find a graph of minimal weight that connects the terminals, and that may include additional nodes called Steiner Points. Steiner Tree problems in graphs have been shown to be NP-complete [64] [57] [45] [43].

Successfully finding Steiner Trees has the potential of favourable outcomes in many application areas, for instance they are particularly advantageous in integrated circuit design [82]. Steiner Trees are also useful in both wired and wireless networks in many environments including healthcare, environmental modelling, transport, and internet of things due to their ability to minimally connect multiple nodes. This can lead to efficient algorithms for finding ideal setups for both wired and wireless sensor networks, as well as being applied in routing problems. Steiner Trees have useful applications in WSN, due to their ability to connect together a set of points.

### 2.4.1 Using and Generating Steiner Trees in WSN

A Steiner Tree has the outcome of connecting a set of points in a graph with a minimal weight, and so the formation of Steiner Trees in WSNs often leads to more efficient utilisation of resources. For instance, a Steiner Tree connecting nodes will lead to optimal routes between the nodes. A Steiner Tree is a preferable data structure to, for example, spanning trees, as it is able to connect a subset of nodes as opposed to requiring all nodes to be connected. Steiner Trees are particularly suited to multisink WSN as it allows all sinks to be connected in single route, rather than creating multiple trees for data delivery. For these reasons, previous work has been carried out with the goal of creating minimal Steiner Trees between sensor nodes in wireless sensor networks. In addition to the problem being NP-complete, finding minimal Steiner Trees in wireless sensor networks has the additional difficultly of often requiring a distributed implementation [91]. This means that often previous works have used heuristic approaches to solve this problem. This section will review works that attempt to find and use Steiner Trees in WSNs.

Much work that aims to find Steiner Trees in WSN is aimed at creating protocols for multicast, where multiple destinations must be reached simultaneously. Multicast is often used in video conferencing, where time delay should be kept to a minimum [68]. A common approach for this is to treat the problem as a minimum spanning tree, with the tree rooted at a particular source or sink node. The work in [44] presents an algorithm for finding minimal Steiner Trees that involves finding minimal spanning trees of the network and then deleting edges such that a Steiner Tree is formed. The distributed version of the algorithm builds a "shortest path forest", where all nodes know the shortest path from itself to each Steiner points. The limitations of this is that each individual node is required to store a lot of information about the network.

The authors of [106] proposes a distributed algorithm for constructing Steiner Trees based on the Cheapest Insertion algorithm [97], an extension of Prim's algorithm for Minimum Spanning Trees. The process starts with a single tree with the nearest terminal nodes being added to the tree iteratively. Adding a neighbour to a tree is a relatively expensive process, as information waves from nodes in the network must all be sent to the root node before it makes a decision about which node to add next. The root node makes a global decision which must be spread throughout the network in a wave like process, which may lead to many packets sent during the process of setting up the tree.

A distributed protocol for generating Steiner Trees for multicast is presented in [112]. Each node creates a subtree, or fragment, which are then incrementally combined to form larger fragments eventually forming a single large fragment. Each node has a routing table containing the shortest distance to all other nodes in order to form the fragments. Nodes within the fragment sends messages to the root node in order to determine nodes to add to the subtree, as well as nodes between subtrees sending connect messages in order to connect subtrees. Generally the protocol involves flooding of messages in order to build the tree. Similar to [106] and [44], the process of creating the tree requires each node to know a reasonably large amount of information about the network, as well as some level of flooding the network with messages in order to create the tree.

Particle swarm optimization (PSO) is used to solve the Steiner Tree problem for multicast networks in [99]. PSO [65] is a meta heuristic where multiple potential solutions, or particles, are moved around a search space in order to move the swarm to better locations. In this case, a swarm of potential trees are randomly created, with the particles moving (replacing paths in the trees) to find the optimal solution. The protocol is not explicitly distributed.

In [87], a method for constructing connected dominating sets in wireless sensor networks using Steiner Trees is presented. A dominating set is a subset of vertices of a graph such that all vertices are either in the dominating set, or adjacent to a vertex in the dominating set. A connected dominating set has all the vertices in the dominating set connected. In [87], a Steiner Tree is used to take a dominating set and ensure it is connected. The distributed algorithm starts with a maximal independent set marked in black, with other nodes in grey. Grey nodes are changed to black based on how many black nodes it is adjacent to, with the final set of black nodes forming the Steiner Tree. This work is limited as no results of an implementation were shown, and the distributed implementation has the potential to lead to many messages sent while not being particularly adaptive to changes.

In [75], Steiner Trees are used to create a wireless sensor network such that the number of sensor nodes are minimal whilst also covering maximal sensing area based on an importance weighting for each area. Three heuristics are presented to find Steiner Trees to cover the network; a greedy algorithm starting with a high value node and merging further nodes greedily, a group based algorithm that merges groups, and a profit based algorithm that adds nodes based on a profit measure.

A biology inspired protocol based on slime mould to find Steiner Trees in networks is presented in [76]. The protocol is based upon fluid flow through tubes of differing widths, with the flow in and out of each vertex being balanced. Using this technique, the protocol creates Steiner Trees in wireless sensor networks to solve the minimum exposure problem; the path between points of interest with least observability.

### 2.4.2 Steiner Trees using ACO

There is a limited number of previous works that aim to create Steiner Trees using Ant Colony Optimisation. [98] uses ACO as part of a two stage process to form Steiner Trees in large graphs representing real world topologies. Clustering is used to divide the large graphs into smaller subgraphs before ACO is used to find Steiner Trees on these subgraphs. These subgraphs are then combined to form one larger Steiner Tree of the total graph in a divide-and-conquer approach. A variation of ACO is used where multiple subcolonies are generated on each subgraph, and each forming a tour on the subgraph. Where there are common vertices between subcolonies, these subcolonies are merged until only one remains. This method is a centralised approach, not aimed at working in a distributed way on sensor networks.

Another centralised approach to finding Steiner Trees using ACO is presented in [56]. This protocol attempts to solve the Rectilinear Steiner Tree variant of the problem. A Hanan grid of the terminals is created and ants placed on each terminal nodes. Ants choose where to travel to next out of the edges of the Hanan grid, remembering where it has travelled with a tabu list. When two ants meet these lists are combined and only one ant continues. After moving, ants leave a pheromone trail between nodes. Ants choose a node to travel to next based on the cost to travel to that node as well as the distance to the other ants routes, in order to encourage ants meeting.

Singh et al. present a method for using ACO in wireless sensor networks to compute minimum Steiner trees [113]. An offline centralised version of the algorithm is described where an ant is placed at each terminal node in the network. The ants then subsequently choose which node to travel to next based on information about the potential of each node. Each ant keeps track of where it has been to prevent repeat visits and is drawn towards routes formed by other ants, merging when it meets either another ant or another path formed by another ant. After this has taken place, pheromone trails between nodes are updated based on the success of the tree created. In addition to an offline algorithm, an online distributed version of the protocol is also presented. This version of the protocol has the goal of providing data centric routing in wireless sensor networks. The distributed algorithm is similar to the centralised one, however ants have less information about the shortest paths between nodes and the sink. Additionally, the concept of backward ants are introduced that have the function of updating pheromone trail. The protocol is limited as creates a tree routed at a single sink, and doesn't take into account multiple sinks. There is the potential to run the protocol multiple times for each sink if there are more than one, however this could be an expensive process and will lead to multiple trees rather than one tree that can route from sources to many sinks simultaneously.

Quality-of-Service multicast routing is reduced to a Steiner Tree problem in the work presented in [129], where ACO is used to create routing trees that satisfy QoS constraints. It considers networks with a single source travelling to multiple destinations. Multiple ants equal to the number of destinations each create a routing tree constrained by QoS requirements, and pheromone is updated based on the overall success of each tree.

### 2.4.3 Fault Tolerant Steiner Trees

Some previous works that use Steiner Trees to provide fault tolerance in wireless sensor networks exist, particularly attempting to solve the problem of restoring connectivity of the network. In this problem case, multiple sensor nodes fail leading to a segment of the network being cut off from the rest of the network. One solution to this problem is proposed in [69], where Steiner Trees are used for minimal placement of mobile relay nodes to reconnect a partitioned network. The Steiner tree is approximated using an approximation algorithm, k-restricted loss-contracting algorithm (k-LCA) [104], limited to 3 terminals. The work presented in [130] also attempts to restore connectivity using Steiner Trees by rearranging network topology avoiding the location of the failed node. This work also makes used of k-LCA with 3 terminals in order to connect partitions with a small Steiner Tree. There is little work in developing Steiner Trees that are fault tolerant in themselves, i.e. recovering from a faulty node that forms the tree.

## 2.5 Summary of Routing Protocols

A summary of the routing protocols discussed in this chapter is shown in the table below. A brief description of their strengths and weaknesses is included.

Table 2.1: Summary of Routing Literature Reviewed

| **Protocol** | **Strengths** | **Weaknesses** |
|---|---|---|
| Flooding [54] | Easy to Implement | Many messages sent so inefficient, collisions need to repeat for every source |

*continues on next page*

| | | |
|---|---|---|
| Gossiping [53] | Easy to implement, fewer collisions | Potentially long routes, need to repeat for each source and sink, no guarantee of delivery |
| Sensor Protocols for Information via Negotiation (SPIN) [54] | Reduced redundancy and fewer collisions compared with flooding | No guarantee of delivery, separate routes for each source sink pair |
| Directed Diffusion [59] | Works with multiple sources and multiple sinks, each node only needs to know about local neighbours | Initial flooding is expensive, continuous reinforcement of multiple routes required, essentially repeating for each sink so many nodes involved and not much aggregation |
| Rumor Routing [13] | Advantages of DD but with less flooding, maintains only one path | Can have less efficient routes, no aggregation for paths to multiple sinks, only maintains one path so may be slower to recover from faults |
| Energy Aware Routing [110] | Based on DD so advantages are similar, and maintains only energy efficient routes | Flooding required to maintain paths, routing tables for each path need to be stored on each node |

*continues on next page*

| | | |
|---|---|---|
| Gradient Based Routing [108] | Based on DD so advantages are similar, also aggregation of data to reduce packet size, load balance scheme to increase life time | Limitations of DD |
| Low Energy Adaptive Clustering Hierarchy (LEACH) [51] | No global knowledge of topology required, Data aggregation in each cluster | Overhead of selecting cluster heads, nodes can only communicate with cluster head so not many-to-many, dependent on nodes being a 1 hop distance to cluster head |
| Power-efficient GAthering in Sensor Information Systems (PEGASIS) [74] | No global knowledge of topology required, Data aggregation in each cluster | Relies on each node having global knowledge of network topology, requires nodes all the way along the chain to send messages every round, longer delays in receiving messages from the ends of the chain |
| Minimum Energy Communication Network (MECN) [105] | Spanning tree created is optimal, so fewer messages need to be send, takes power consumed into account | Requires GPS for each node, assumes all nodes can communicate with every other node in network, doesn't cover |

*continues on next page*

| | | |
|---|---|---|
| Small Minimum Energy Communication Network (SMECN) [72] | Same as MECN but can also deal with obstacles | Same as MECN and Higher overhead |
| Geographical Adaptive Fidelity (GAF) [127] | Cycles which nodes are awake to save energy | Requires location awareness, assumes all nodes the same cost, minimal aggregation |
| Geographic and Energy Aware Routing (GEAR) [132] | Based on DD but uses location awareness to direct interests rather than flooding | Requires flooding to find routes, constant cost of route maintenance, no aggregation for multiple sinks |
| Collection Tree Protocol (CTP) [46] | Creates low cost trees from multiple sources to sink at root | Multiple sinks would require multiple trees to be built, which means maintaining a lot of information and lots of repeat transmissions, need to periodically broadcast ETX value to neighbours |
| Two-Tier Data Dissemination (TTDD) [78] | Can delivery to multiple mobile sinks, nodes don't need global topology | Requires location awareness, sinks need to flood to request data |

*continues on next page*

| | | |
|---|---|---|
| Multisource Multisink Trees for Energy-Efficient Routing (MUSTER) [89] | Creates efficient trees to route from multiple sources to multiple sinks with data aggregation, element of load balancing | Periodic network wide broadcasts to refresh trees, trees built using flooding |
| Multiple Sink Placement [95] | Finds best placement for sinks to increase network lifetime | Not a routing protocol |
| Multipath Routing in large scale sensor networks with Multiple Sink nodes (MRMS) [18] | Path switching for increased network lifetime | Costly to maintain and switch cluster heads, doesn't deliver to all sinks, nodes only delivery to one of many sinks |
| AntNet [24] | Distributed, one of the first adaptations of ACO in networks | Designed for wired networks, single destination, requires storage of routing tables |
| ACO protocols by Zhang et al. [134] | Adaptation for WSN, cost aware to find shorter paths | Only single sink, uses flooding |
| Energy-Efficient Ant-Based Routing Algorithm (EEABR) [15] | Energy aware to increase network lifetime | Only single sink, may have long setup time, neighbours need to be aware of energy levels of neighbours which will change |

*continues on next page*

| | | |
|---|---|---|
| ACO-based quality-of-service routing (ACO-QoSR) [14] | Satifies time delay constraints, some fault tolerance | Only single sink, fault tolerance sensitive to false positives |
| AntChain algorithm [26] | Doesn't require knowledge of global topology | Only single sink, centralised, requires location awareness, all nodes need to be in range of the sink |
| ACO clustering protocol [67] | Forms clusters dynamically with ACO to adapt to failures in the network, fault tolerance using 'hello' ants | Additional cost in form of 'hello' ants, sensitive to false positives, periodic updates required |
| ACO for Steiner Trees [113] | Creates minimal costs Steiner trees | Needs to be repeated for each sink in the network, which is costly |
| BeeSensor [107] | Energy aware, can maintain a single path for efficiency | Need to maintain multiple paths for continuous data delivery |
| Cluster based routing in WSNs using artificial bee colony algorithm [63] | Forms clusters | Requires centralised network |
| Glowworm Swarm Optimisation (GSO) [73] | Creates a well covered network of sensor nodes | Sensor deployment not routing |

## 2.6 Summary of Fault Tolerance Protocols

A summary of the fault tolerance protocols discussed in this chapter is shown in the table below. A brief description of their strengths and weaknesses is included.

Table 2.2: Summary of Fault Tolerance Literature Reviewed

| **Protocol** | **Strengths** | **Weaknesses** |
|---|---|---|
| Coverage Configuration Protocol (CCP) [121] | Provides minimum coverage in the network, only local knowledge needed | Minimum coverage may not be possible, no recovery from failures in coverage found |
| Coverage using least number of sensors [60] | Minimum number of nodes required to provide a good chance of coverage is found | Minimum coverage may not be possible, no recovery from failures in coverage found |
| Energy level reporting [136] | Network wide scan for energy levels | Requires human intervention to then act on energy levels |
| Predicting residual energy [88] | Less overhead than actually measuring energy | Less accurate, requires intervention to act on energy levels |
| GRAdient Broadcast [128] | Multiple paths followed to be robust to failures on any one path | May not scale well with multiple sinks |
| Sink detection of faults [100] | Sink finds faults | Requires powerful sink to constantly monitor |
| Sink detection of faults using routing messages [114] | Sink uses normal messages to detect faults | Still requires powerful sink to constantly monitor |

*continues on next page*

| | | |
|---|---|---|
| [25] and [126] | Localised fault detection of neighbours, finds inaccurate results | No recovery process |
| Watchdog and Pathrater [84] | Uses packets it sees to find faults and then find alternative route | May miss collisions, overhead in listening to neighbours, may need to recalculate routes from scratch |
| Heartbeat failure detection [118] | Only send heartbeat to cluster head | Still overhead in heartbeat messages, only detection |
| [55] and [19] | Comparing with neighbours to detect faults | Overhead, may miss collisions |
| Directed Diffusion [58] | Detects faults of paths and finds alternative route | Requires maintenance of alternative routes, must monitor reporting from neighbours constantly |
| Tracing failed nodes [114] | Traces failed nodes | Requires powerful base station with knowledge of network topology |
| Highly-resilient, energy-efficient multipath routing [42] | Uses braided multipaths so all paths remain relatively short, overhead in maintenance is lower | More chance of multiple paths failing, still requires reinforcement messages |
| Reliable Energy Aware Routing (REAR) [52] | Distributed, backup path for recovery | Only two paths so if both fail have to restart |

*continues on next page*

| | | |
|---|---|---|
| Reliable Information Forwarding (ReInForm) [21] | Uses redundancy for fault tolerance | Requires periodic broadcasts to all nodes from the sink, high overhead as need to send multiple packets |
| Erasure coding in WSN [34] | Don't need all packets to arrive to deliver all data | Can still fail if don't receive enough packets |
| The N-to-1 Multipath Routing Protocol [123] | Local decisions to deal with faults | Routes may not be efficient as found via flooding , single sink |
| H-SPREAD [124] | Improves both security and reliability by secret sharing scheme | Multipaths may be inefficient as found via flooding, single sink |
| Multipath Multispeed Protocol (MMSPEED) [39] | Locally made decisions, multipath for fault tolerance | Periodic updates required and location awareness required |
| ACO-based quality-of-service routing (ACO-QoSR) [14] | Fault tolerance with hello messages | Sensitive to false positives, overhead in sending hello messages |
| ACO approach[23] | Uses ACO to find efficient routes and switch between them in case of failure | All routes must be stored ahead of time, no guarantee that there will be a route without failures |
| ACO based spanning tree with ABC[86] | ACO creates spanning tree and edges deleted if node failures | Needs to rerun parts of protocol whenever a failure occurs |

*continues on next page*

| | | |
|---|---|---|
| ACO based routing [38] | ACO sets up multiple routes ahead of time and a different route is switched to ahead of time | Need to store multiple routes, may have to re-run again if failure on multiple routes |
| ACO look ahead approach[117] | ACO based uses pheromone only to reinforce good QoS routes | Needs powerful sink to calculate QoS and decide which route to reinforce |

## 2.7 Summary of Protocols relating to Steiner Trees in WSN

A summary of protocols related to Steiner Trees discussed in this chapter is shown in the table below. A brief description of their strengths and weaknesses is included.

Table 2.3: Summary of WSN Steiner Trees Literature Reviewed

| **Protocol** | **Strengths** | **Weaknesses** |
|---|---|---|
| Steiner Tree problem in distributed computing systems [44] | Distributed, finds ST | Each node needs to store a lot of information |
| Distributed multicast routing[106] | Distributed, finds ST | Need to inform root every time a node is added to the tree, expensive setup |

*continues on next page*

| | | |
|---|---|---|
| Distributed protocol for multicast [112] | Distributed, finds ST | Flooding and lots of information needed to be stored on each node, expensive setup |
| Dominating sets using ST [87] | Distributed | Large amounts of messages sent, not very adaptive to changes |
| ST to to cover maximum weighted critical square grids [75] | Minimises nodes required in network | No focus on routing |
| Physarum optimization [76] | Uses slime mould inspired method to find ST | Quite complicated to implement |
| ACO for rectilinear STs [56] | Finds ST successful | Centralised, on rectilinear ST |
| ACO for ST [113] | Finds minimal ST for a range of networks | Needs to be repeated for each sink, takes a reasonably long time to find ST |
| QoS multicast routing [129] | Creates ST satisfying QoS constraints | Single source |
| Recovery from simultaneous failures using ST [69] [130] | Successful at restoring connectivity in networks | Relies on mobile sensor nodes to do so |

## 2.8 Summary

The summary of routing literature reviewed in this chapter shows that often existing state-of-the-art protocols have similar disadvantages. Many only consider a single sink, and where protocols do consider more than one sink, this is through repeating the routing protocol for each sink. This is both expensive in terms of set up, for instance requiring many more messages being sent using more energy, as well as the amount of information each node needs to store in for routing tables. Additionally, routing to multiple sinks in this manner means that there are multiple disjoint routes for each sink, when these could be aggregated to form a more efficient route, again a cost in energy for messages sent. The work in this these improves upon this by developing a protocol that is specifically designed for use with multiple sinks. The routing protocol only needs to be run once regardless of the number of sinks, additionally the amount of information needed to be stored for more than one sink will not significantly increase. The route itself formed will be a single route, that will minimise the total number of nodes involved and messages sent for delivery to all sinks from all sources.

Fault tolerant protocols often rely upon setting up multiple routes ahead of time, and then if a failure occurs switching to a different route. This has higher overhead in terms of the information that nodes need to store, and also will fail if a failure occurs on all the routes that are found ahead of time. There is a need for a fault tolerant routing protocol that is able to route around failures online, reacting to failures that occur as they happen. The work in this thesis addresses this, developing a fault tolerant protocol using ACO that recovers from faults without needed to set up routes ahead of time, or reruning the protocol. Existing work that does this is minimal, or is often requires periodic network updates flooded to all nodes in the network, so the routing protocol developed here avoids this energy expensive process.

Existent work for finding Steiner Trees in WSN is minimal, with much work in the context of multicast which only considers a single source. Additionally, much work requires each node to store information about the Steiner Tree in order to build the tree. The work here aims to create Steiner Trees without needing to store large amounts of information on each node, as well as being effective in networks with multiple sources and multiple sinks.

# Chapter 3

# Problem Statement and Experimental Setup

The goal of this work is to solve a number of problems associating with routing in grid based wireless sensor networks using Ant Colony Optimisation. As an overview, the problems that will be investigated are as follows:

- Routing messages from multiple sources to multiple sinks (many-to-many routing) efficiently, minimising packets sent and nodes involved.
- Creating scaleable protocols that do not become less efficient in terms of messages sent and nodes involved as network size increases. Existing literature requires repeating the protocol for each sink, which can be inefficient and energy expensive, the work presented here aims to avoid this.
- Providing fault tolerance for many-to-many routing in the case of node failures in the network.
- Generating Steiner Trees in wireless sensor networks in order to connect arbitrary sources and sinks.

To solve these problems, the metaheuristic of Ant Colony Optimisation (ACO) is used. ACO has many advantages, largely that it can be used to find optimal solutions to NP-hard problems including various path finding problems in graphs. For instance, ACO is often used in travelling salesman and its variants. ACO is particularly useful for the the problem contexts explored in this work, as routing problems in WSN can be thought of as path finding problems. The same ACO based framework for routing in many-to-many WSN can be applied to all three problem areas using the same basic ideas of the protocol without having to make large changes, showing the advantage of this approach in its ability to solve varied problems in differing circumstances without the need for a specific set of starting conditions.

This chapter will describe in detail the specification of the problems that are to be solved, and the motivations behind design decisions taken to solve them. The assumptions about network topologies and abilities are also described. In addition, the metrics used to measure performance over the course of the work are also described in detail. For each problem, appropriate performance metrics will be chosen to evaluate success for each solution, though many metrics being common between problems. Finally, an overview of the experimental setup used to test solutions will be described. The experimental setup will remain largely consistent between experiments.

## 3.1 Objectives

The overall objective of this work is to develop a framework based on Ant Colony Optimisation in order to solve a number of problems in wireless sensor networks with multiple sources and multiple sinks. A key advantage of the ACO based framework is that is able to solve different problems without large changes to the protocol, indicating that it could be adapted to solve further problems in varying network circumstances without needing highly specific network starting conditions.

The first objective is to develop an ACO based routing protocol that is able to deliver messages from multiple sources to multiple sinks. The protocol should be efficient, minimising the use of network resources in order to increase overall network lifetime. Additionally it should have success in delivering messages from all sources to all sinks, such that the advantages of many-to-many communication is maximised. An element of this objective is scalebility, which means that the protocol should not use more resources in terms of messages sent and nodes involved disproportionately to increases in network size. The benefits of aggregation should remain when the network gets larger, without increasing amounts of information needed to be stored on each node. The protocol should not need to be repeated for each sink in the network, as is the case in existing literature.

The second objective is to extend the ACO based routing protocol for fault tolerance, and so expanding the framework of many-to-many routing. The protocol should react to and recover from faults whilst also maintaining successful communication between sources and sinks. Again, the process should be efficient, minimising the number of messages sent and increasing network lifetime.

The final objective is to take the developed ACO based routing protocol and investigate its ability to form Minimal Steiner Trees in a many-to-many WSN. The structure is an efficient way of connecting sources and sinks in the network, and shows the adaptability of the protocol to solve different problems. In this case, the basics of the algorithm is maintained with some changes to the start up phase and how the backbone is combined and split. Additionally, the protocol should be scaleable with multiple sinks, and not require rerunning the protocol for each sink in the network.

## 3.2 General Network Characteristics

The network taken into consideration for the protocol developed has a number of assumed characteristics. All nodes are considered to be capable of both sensing data about the environment as well as forwarding data throughout the network from sources to sinks. In this sense, all nodes may be sensing nodes or relay nodes. Similarly, any node could also potentially be a sink node for routing purposes. For each network, sources and sinks are chosen from the nodes in the network. It is assumed that the sources in the network will want to continuously deliver sensed data towards the sinks. This work only takes into consideration grid based networks.

The protocols developed are aimed at a sensor network where nodes in the network do not have complete knowledge of the network topology. This means that there is no sink nodes or base station with a centralised view of the network, and as such distributed protocols are required. Though some networks will have such a base station or will have each node aware of its location within the network, this work assumes this is not possible. The networks considered also assume that few nodes will be able to communicate directly with the sink, relying upon multihop communication.

## 3.3 Experimental Setup

All versions of the ACO based protocol for routing in many-to-many wireless sensor networks were implemented in Contiki OS for simulation on the COOJA network simulator [35]. Contiki OS is a lightweight OS designed for devices for the internet of things, and is commonly used in WSNs. COOJA is a simulator used with Contiki OS that is able to simulate a wide variety of network types. For all simulations performed, emulated Sky motes are used, with the UDGM radio medium, CSMA MAC driver, and the null RDC driver for transmission. These choices where made as it allowed consisting delivery between nodes and few failures of delivery, such that faults usually only occurred when added on purpose to test fault recovery.
The network topologies tested varied in size, but always consisted of a square grid of sensor nodes spaced equally apart. All transmission ranges were set such that each node could only communicate with its most immediate horizontal

and vertical neighbours. The choice for the grid was taken as this is often a realistic setting for WSN, for instance in large buildings. Additionally, it would enable the benefits of the backbone to be most obvious, allowing a large amount of data aggregation through shared routes. The grid of nodes was of length $n \times n$, where $n \in \{5, 7, 9, 11, 13\}$. When not comparing sizes, a network of size 11 x 11 was used by default. The network of 121 was chosen as it was large enough to feasibly represent a realistic large sensor network, but also the computing resources available could simulate the network size in a reasonable amount of time for a large number of repeat experiment. Generally sources will be placed in the lower half of the grid and sinks in the top half, varying where appropriate.

## 3.4 Performance Metrics

A number of performance metrics are calculated in order to determine the success of the protocol. Some of these metrics will be common for all versions of the protocol for each problem specification, such as delivery ratio. This is because some particular metrics are used to determine common success factors, such as ability to delivery messages to the sinks. Others are more specific, for instance fault tolerant related measures. Performance metrics that will be used to measure success are described in this section.

- **Delivery Ratio**: The ratio of messages sent from the sources to those received on the sinks. This metric has the goal of measuring how successful the protocol was in delivering sensing data to the sources. Where sources and sinks are varied, the delivery ratio measure is changed to take into account where the unequal number of sources and sinks. For instance, if there are 2 sources but 4 sinks, it is expected that 4 messages are received on the sinks for each 2 sent, and the ratio is changed accordingly to reflect this. In addition to the base delivery ratio, there is also the “All Sinks” measure, which aims to only count cycles where the backbone is formed of ants originating from all sources and all sinks receive messages.

This is used to determine how often all sinks received from all sources.

- **Number of Nodes involved per cycle**: This is how many nodes are involved in forming the route from the sources to the sinks, and is a measure of how efficient the route formed has been. The goal is to have as lower number of nodes involved in forming the route as possible. Some variations of the protocol will have more nodes involved than there are on the final route in the process of finding that route, and this metric includes these nodes too. This metrics also has an "All Sinks" measure, which only looks at cycles where all sinks receive from all sources. The goal is to have as fewer nodes involved as possible, as this implies short routes, and so less energy expended and a longer network lifetime.

- **Number of Nodes involved on route per cycle**: This metric only counts the number of nodes involved on the final route from sources to sinks. This is required as in some variations of the protocol multiple routes will form, but only one is considered to be the final route. This number should also be low in order to reduce over all energy use and increase network lifetime.

- **Packets sent per cycle**: Packets sent includes the forward, backward, and beacon ants. A lower number of packets sent is more successful, as this leads to a longer network lifetime with less energy being expended on sending packets.

- **Backbone Length**: The number of nodes that form the backbone. This metric should ideally be a higher proportion of the total route from sources to sinks as possible, as this indicates that the benefits of aggregation are greater. Also, a route that has a higher proportion of it consisting of the backbone would imply fewer nodes involved and fewer messages sent overall.

- **Fault Recovery Time**: This metric is investigated when measuring the success of fault tolerant variants of the protocol, and is the time to recover from a faulty node in cycles. Recovery is defined to be the time from the fault occurring to the first cycle where all sinks receive from all sources via a backbone. A lower fault recovery time is considered to be a better performance, as less time is spent unsuccessfully delivering from sources to sinks.

- **Fault Detection Time**: The time from a fault occurring to its neighbours recognising the failed node as faulty. This should also be low, as this means that faults can be recovered from more quickly as recovery will start after the fault is detected.

Chapter 4

# Routing in Many-to-Many Wireless Sensor Networks using Ant Colony Optimisation

Wireless Sensor Networks consist of multiple sensor nodes communicating with each other using radio messages. Routing protocols directing messages from nodes that have sensed data (source nodes) to nodes that require the data (sink nodes) is a highly researched area. The sink node will then act upon this data in some way, be that forwarding it to a base computer, sending it to a larger computer network, or initiating an actuator. Some routing protocols assume that all nodes are able to communicate directly with the sink node in a single hop, however, due to limitations with radio power or range, particularly in networks with large distances between nodes, this is not always possible. Such circumstances require multiple hops to route data from source to sink, or multihop routing.

Many protocols consider a single sink, however a less studied routing problem is where a network has multiple sinks, all requiring data from the sources.

This is can be referred to as many-to-many routing. A many-to-many network may be useful in situations where sink nodes are some form of actuator, or for reliability purposes where it is helpful to have multiple sinks in case of failure. Often existing solutions for many-to-many routing are simply many-to-one routing protocols run for each sink in the network. This is inefficient and not a scaleable solution as the number of sinks increases. Other solutions involve a centralised view of the network, or require a flooding the network with messages, leading to a large number of messages being sent, reducing network lifetime. As a solution to the problem of routing from multiple sources to multiple sinks, this chapter introduces a distributed protocol based on ant colony optimisation (ACO) that successfully delivers messages from all sources to all sinks while remaining scaleable and efficient.

Ant Colony Optimisation was chosen as a solution to many-to-many routing for its ability to solve complex routing problems using simple agents making local decisions. This makes it well suited to the context of distributed routing protocols for sensor networks, as global topology knowledge is not required. As a meta heuristic, it is able to create optimised solutions to hard problems, and improve upon the solution found over time. The heuristic works on the basis of individual agents being sent from sources and choosing a sensor node to travel to based on local information. In this case, the agents are the Ants, which are represented as messages being sent between nodes. Ants will eventually reach a sink node, where they are transformed into backward ants that travel back through the network following the original route to the source. These backward ants update a value between nodes called "pheromone trail" proportionally to the success of the route taken by the ant. This pheromone trail is used by subsequent ants to decide which node to travel to, and is also evaporated periodically in order to encourage route improvement.

The ACO protocol that is described in this chapter makes use of data aggregation through a shared backbone of nodes. This is a process where ants meet and combine on the same node. From this point, a single ant continues travelling, reducing the total number of messages sent. The backbone enables the protocol to delivery messages from all sources to all sinks while also minimising both the total number of nodes involved as well as the total packets sent.

To summarise, the solution proposed to the problem of scaleable many-to-many routing in wireless sensor networks is a distributed protocol based on ACO. Results show that the protocol is able to successfully deliver messages from multiple sources to multiple sinks through the used of a shared backbone of messages, enabling scalebility. The contributions made in this chapter are:

- The protocol for many-to-many wireless sensor networks using ant colony optimisation.
- A distributed implementation of the protocol that is scaleable with network size.
- A novel distributed implementation of ACO that does not use tabu lists or ant memory (a common feature of ACO based routing protocols in WSN), to reduce packet size.
- Simulations of the distributed protocol on the COOJA network simulator with the implementation written in Contiki OS.

## 4.1 Problem Specification

The problem to be solved in this chapter is stated to be: Given a WSN, a number of sources $s$ and a number of sinks $S$, find a minimal set of edges such that data can be routed from multiple sources $s_1, \ldots, s_k$ to multiple sinks $\Delta_1, \ldots, \Delta_l$ and that the number of nodes involved is minimal. This scenario is considered to be NP-hard, as it is a variant of the vehicle routing problem which is known to also be NP-hard [70].

Many existing solutions build a routing data structure over the network, often one structure per sink node. This type of solution is not scaleable, often requiring large routing tables or periodic network flooding. To avoid these issues, the population-based metaheuristic Ant Colony Optimisation (ACO) is used for routing. Metaheuristics are generally problem independent and so can be applied to many different types of problems, and can be adapted to this routing problem. An advantage of a distributed ACO protocol is that decisions are made locally within the network; each node makes local decisions about where to send messages to next, without requiring network wide knowledge of topology, or any knowledge about which route to take in advance. Through the use of pheromone evaporation, good routes are reinforced, while also making the metaheuristic adaptable to changes.

## 4.2 Description of Protocol

Ant Colony Optimisation is a heuristic algorithm based on the movements of ants in a colony. In nature, ants will travel towards a desired destination, such as a food source, while dropping a pheromone on the ground. Subsequent ants will then follow the pheromone trails left by ants, preferring routes with a higher level of pheromone. As the pheromone will evaporate over time, shorter routes are reinforced more, and so ants tend to follow these routes [31].

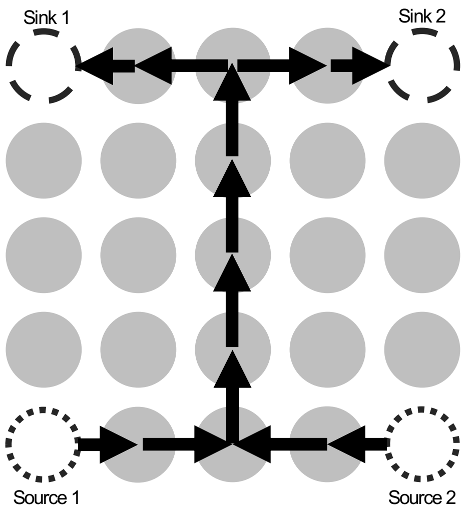


Figure 4.1: Example of a network with 2 sources and 2 sinks with a backbone formed.

The protocol developed represents ants as packets being sent through the network, travelling on a multihop basis from the sources to the sinks. Every node will keep track of its neighbours using a linked list which also contains a pheromone value associated with each neighbour. Each ant chooses the next node to travel to using a probability function. The ants keep a memory of the route which is used to prevent repeat visits to the same node, referred to here as "ant memory". When the ant reaches a sink, it is converted into a backward ant, which then uses the ant memory to travel backwards through the network to the original sources following the same route as the forward ants, updating the variable $\Delta\tau$ that is used for pheromone updates as it passes through each node. The update value depends on the success metrics of the forward route and is used to increase the pheromone values between nodes. The pheromone value is also periodically reduced to account for evaporation. Forward ants are launched periodically from the sources, with the time between each launch referred to as a *cycle*. The cycles are timed such that backward ants will have completed their update of pheromone trail before more forward ants are sent.

In addition to the base ACO implementation, the protocol also aims to combine ants in order to aggregate data. When the multiple forward ants meet on the same node (within some time period $s$), these ants are combined into a single ant. This combined ant then travels along a shared path, or *backbone*, for a long as possible, before splitting into several ants again such that all sinks are visited. An example network with such a backbone is shown in figure 4.1. Through this aggregation of messages, the total number of packets sent will be less, thus improving network lifetime.

The ACO based protocol presented in this chapter has two main variations:

- A base ACO protocol that is able to form a backbone to send messages from sources to sinks.

- An extension of the base protocol with no ant memory, a feature of ACO based protocols often used in previous ACO based protocols. Without this, packets sent are much smaller with very little reduction in performance, making the protocol more scaleable as long routes no longer need to be recorded.

### 4.2.1 ACO for Many-to-Many Routing

The ACO protocol starts with a set up phase, where neighbour lists are initialised. This involves each node discovering which nodes are in communication distance, referred to as neighbouring nodes, as well as working out the hop counts of all neighbouring nodes to the sinks.

#### Set Up Phase

1. Each node periodically sends local broadcasts containing an node identification number to identify neighbours. Each node, $i$, keeps track of its neighbours in a list, denoted $N(i)$, which also stores other information about each neighbouring node, including the pheromone values to the neighbours updating throughout the main algorithm.

2. Hop counts to sinks are also established and stored in $N(i)$. Each node maintains this knowledge of hop counts for itself and its neighbours throughout. To ascertain hop counts, the sink nodes initiate flooding with hop count of value 0.

3. On first receiving a broadcast, nodes note their own hop count to that sink by incrementing the received hop count by 1. It also updates $N(i)$ with the hop count of the neighbour it received from. The node then broadcasts their own hop count to its neighbours.

4. On receiving further broadcasts, nodes will update their hop count if there is an improvement, i.e. if they receive a hop count that would lead to their calculated hop count being smaller than the current recorded

value. This step is required as broadcasts may not be received in order i.e. the most direct route with the actual hop count to the sink doesn't necessarily reach the node first. Similarly, if it has received a hop count that represents an improvement on the currently recorded value for a neighbour, it will also update this value.

5. When a non sink node changes its hop count, either for the first time or on an improvement, it will announce this figure to its neighbours.

6. This process continues for a set amount of time appropriate to fully populate neighbour lists.

Following this set up phase, the main running of the protocol begins. This consists of continuous delivery of data from the sources to the sinks.

### Data Delivery

1. Initially set all pheromone values to neighbours to be some constant $c$. Set a time variable $t$ to be 1 and cycle count variable to be 1. A cycle is defined to be the time between each periodic broadcast of ants and is a constant value known to all nodes.

2. Forward ants are periodically launched from the source nodes in the form of a packet sent. Each ant, $a$, has a stored ant memory containing the route, denoted $route_a$, in its payload.

3. The forward ant chooses the next node to travel to with the probability shown in equation 4.1, below. This probability is based on that posed in Ant System, an early ACO algorithm, [31]. The original probability uses both pheromone trail amount between nodes, and visibility, a measure of distance to bring ants closer to a solution. Here, pheromone trail is used in order to encourage following successfully followed routes. The visibility measure present here is the use of average hop count, which brings the ants closer to the sinks by preferring nodes with lower hop counts. Additionally, hop range is used in order to encourage the formation of a

backbone, as encouraging ants towards nodes with smaller ranges means getting closer to a backbone that is equal distance between all sinks. The probability equation is therefore

$$p_{i,j} = \begin{cases} \frac{[\tau_{i,j}]^{\alpha} \cdot \frac{1}{\eta_j}^{\beta} \cdot \frac{1}{\varepsilon_j}^{\gamma}}{\sum\limits_{k \in N(i) \notin route_a} [\tau_{i,k}]^{\alpha} \cdot \frac{1}{\eta_k}^{\beta} \cdot \frac{1}{\varepsilon_k}^{\gamma}} & \forall j \notin route_a \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $p_{i,j}$ is the probability that the ant $a$ will travel from node $i$ to node $j$, $\tau_{i,j}$ is the pheromone value between node $i$ and node $j$, $\eta_j$ is the average hop count to the sinks for node $j$, $\varepsilon_j$ is the hop range to the sinks for node $j$. $\alpha$, $\beta$, and $\gamma$ indicate the weight of these parameters. $route_a$ indicates the memory of ant $a$ and $N(i)$ represents the neighbours of node $i$.

4. When a node has received messages from multiple ants within time $s$ of each other, the data is aggregated and a single ant is forwarded, forming the backbone. This ant continues choosing nodes to travel to with equation 4.1.

5. The backbone ant will travel until it reaches a node where there are no possible neighbours to travel to that will decrease the hop counts for all sinks. At this point, the backbone ant will split into individual forward ants again through a broadcast. Neighbouring nodes that are not already part of the forward path, that is those that are not in $route_a$, will continue travelling towards the sinks. After the backbone is split, the probability of travelling to a neighbouring node becomes the equation shown in 4.2, taking into account the smallest hop count to any sink as the visibility measure. This will direct the ant to its closest sink. Hop range is no longer considered as there is no need to form the backbone. Pheromone trail is still considered as ants still need to try to follow

previously successful routes. The probability is

$$p_{i,j} = \begin{cases} \frac{[\tau_{i,j}]^{\alpha} \cdot \frac{1}{h_j}^{\beta}}{\sum_{k \in N(i) \neq prevnode} [\tau_{i,k}]^{\alpha} \cdot \frac{1}{h_k}^{\beta}} & \forall j \notin route_a \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where $p_{i,j}$ is the probability that the ant $a$ will travel from node $i$ to node $j$, $\tau_{i,j}$ is the pheromone value between node $i$ and node $j$, and $h_j$ is the smallest hop count to any sink for node $j$. $\alpha$ and $\beta$ indicate the weights of these parameters. $route_a$ indicates the memory of ant $a$ and $N(i)$ represents the neighbours of node $i$.

6. Once the ants reach the sinks, they are transformed into backward ants. If a forward ant has reached a sink without having formed a backbone, backward ants are not generated, so that only paths where a backbone is formed are reinforced. Backward ants follow the reverse of $route_a$, travelling back to the source nodes. The first backward ant to reach the backbone continues along it, such that only one backward ant traverses the backbone, with subsequent ants halting at the backbone. At the end of the backbone the backward ant is split and continues towards the sources using a broadcast. As the backward ants travel, they update a value in the neighbours list, $\Delta\tau$, for the neighbour it has travelled from, which represents the change in pheromone value, using:

$$\Delta\tau_t = \Delta\tau_{t-1} + \frac{1}{l_r} + (l_b \cdot B) \tag{4.3}$$

where $\Delta\tau_t$ indicates the change in pheromone value at current time $t$, $l_r$ represents the total forward route length, including the multiple routes to and from the backbone, $l_b$ is the length of the backbone, and $B$ is the weight of the backbone length.

7. At the end of each cycle the pheromone trail between all nodes is evaporated using equation 4.4. Each node independently updates the values in

their neighbours list. The pheromone update is

$$\tau_t = (\rho \cdot \tau_{t-1}) + \Delta\tau_t \tag{4.4}$$

where $\tau_t$ represents the pheromone value at time $t$, $\rho \in 01$ indicates the rate of evaporation, and $\Delta\tau$ is the trail change value.

8. Increment the cycle count by 1, $t$ by 1, and launch one ant from each source node to start the next cycle.

9. This process will be repeated indefinitely, or until some stop condition is met adapted to the circumstances of the particular WSN.

The protocol is described further in Algorithms 2, 3, and 4. Algorithm 2 is continually ran by all nodes in the network, with the current node represented as *node*. If a forward ant is received, then Algorithm 3 is used; similarly Algorithm 4 is called when a backward ant is received on a node.

**Algorithm 2** Description of ACO Protocol for Many-to-Many Routing

```
1: procedure ACO-PROTOCOL
2:     $cycle \leftarrow 0$
3:     $t \leftarrow 0$
4:     $cycle_{max} \leftarrow C$
5:     while $cycle < cycle_{max}$ do
6:         $messages \leftarrow 0$          ▷ The number of messages received by this node
7:         if $node \in sources$ then
8:             Choose next node $n \in neighbours$ with Eqn (4.1)
9:             Initiate forward ant memory, $route_a$          ▷ A list of visited nodes
10:            $route_a \leftarrow node$
11:            Send message from $node$ towards $n$, with packetbuf $route_a$
12:        if $node$ receives a forward ant then
13:            FORWARDANTRECEIVED()
14:        else if $node$ receives a backward ant then
15:            BACKWARDANTRECEIVED()
16:        Evaporate pheromone between $node$ and $n$ with eq:pheromone-update
17:
```

### 4.2.2 ACO Protocol with no Ant Memory

The ACO protocol was then extended remove the use of ant memory, which is analogous to the tabu list in traditional ACO algorithms. Ants no longer travel with a memory consisting of all the visited nodes, instead each node keeps track

**Algorithm 3** Description of Forward Ant Protocol

```
1: procedure ForwardAntReceived
2:     Add $node$ to ant memory
3:     $route_a \leftarrow node$
4:     $messages \leftarrow messages + 1$
5:     if $node \in sinks$ then
6:         Reverse $route_c$ to form backward ant memory $route_{bw}$
7:         Choose next node $n$, the node in $route_a$ after $node$
8:         Launch backward ant; send message from $node$ towards $n$, with packetbuf $route_{bw}$
9:     else
10:        if Backbone has not been formed then
11:            if $messages \neq |sources|$ then
12:                Store $route_a$ internally
13:                Set a timer for $s$ seconds.
14:                while Timer is not yet finished do
15:                    Wait
16:                if $messages \neq |sources|$ then
17:                    $messages \leftarrow 0$
18:                    Choose next node $n$ with probability (4.1)
19:                    Send message towards $n$, with packetbuf $route_a$
20:            else                                              ▷ Form the backbone
21:                Combine $route_a$ with other stored routes to form $route_c$.
22:                Choose next node $n$ with probability (4.1)
23:                Send message towards $n$, with packetbuf $route_c$
24:        else
25:            if Backbone has been split then
26:                Choose next node $n$ with probability (4.2)
27:                Send message towards $n$, with packetbuf $route_c$
28:            else if $\exists n \in N(n) \rightarrow (\forall i \in sinks \rightarrow h_{n,i} < h_{node,i})$ then
29:                Choose next node $n$ with probability (4.1)
30:                Send message towards $n$, with packetbuf $route_c$
31:            else                                              ▷ Split the backbone
32:                Send $|sinks|$ messages from $node$ to the neighbours with the smallest $h_{n,i}$
   for any sink, with packetbuf $route_c$
33:
```

**Algorithm 4** Description of Backward Ant Protocol

```
1: procedure BackwardAntReceived
2:     $bwmessages \leftarrow bwmessages + 1$
3:     Update change in pheromone with equation 4.3
4:     if $node \in sources$ then                        ▷ Backward ant route complete
5:         Stop ant, send no more messages.
6:     else
7:         if $node$ is the first node of the backbone in $route_{bw}$ then
8:             if $bwmessages > 1$ then
9:                 Stop ant, send no more messages
10:        else if $node$ is the last node of the backbone in $route_{bw}$ then
11:            Broadcast messages to $node \in N(n)$ where $node \in route_{bw}$
12:        else
13:            Choose $n$ the next node in $route_{bw}$
14:            Send message towards $n$ with packetbuf $route_{bw}$
```

of the previous node, i.e. the node it received an ant from, and the next node, i.e. the node it chooses to send an ant too. When choosing a node to send an ant to, the immediately previous node is the only node to be discounted, rather than all the nodes in the ant memory. Using this method, there is no guarantee that the ant will not loop back round and repeat a particular node more than once. However, in practice it will be shown that this is unlikely to happen. This can be seen through the high delivery ratios and similar number of nodes involved to the version of the ACO protocol with ant memory. This shows that loops cannot be occurring often, or the nodes involved value would be high. Despite the lack of ant memory, loops won't form due to the use of hop counts and pheromone trail to guide the ants in the correct direction. This leads to changes of the probability for choosing a node to be

$$p_{i,j} = \begin{cases} \dfrac{[\tau_{i,j}]^{\alpha} \cdot \frac{1}{\eta_j}^{\beta} \cdot \frac{1}{\varepsilon_j}^{\gamma}}{\sum\limits_{k \in (N(i)-prevnode)} [\tau_{i,k}]^{\alpha} \cdot \frac{1}{\eta_k}^{\beta} \cdot \frac{1}{\varepsilon_k}^{\gamma}} & \forall j \neq prevnode \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where $p_{i,j}$ is the probability that the ant $a$ will travel from node $i$ to node $j$, $\tau_{i,j}$ is the pheromone value between node $i$ and node $j$, $\eta_j$ is the average hop count to the sinks for node $j$, $\varepsilon_j$ is the hop range to the sinks for node $j$. $\alpha$, $\beta$, and $\gamma$ indicate the weight of these parameters. *prevnode* indicates the noted previous node and $N(i)$ represents the neighbours of node $i$.

This change of removing ant memory has the advantage of reducing the required packet size to be sent between nodes. This is because the packet size included the ant memory, and without this the packet size will be reduced by the size of the ant memory, which could be as long as all the nodes in the network, depending on network size. This should help reduce energy requirements and leads to better reliability.

The other change made for this version of the protocol is that backward ants now choose which node to travel to using each nodes previous node, and not by scanning through the ant memory. This has the advantage of being quicker, as well as reducing the amount of data that each backward ant needs to carry.

## 4.3 Distributed Implementation

The ACO algorithm described in the previous section was implemented as a distributed protocol for wireless sensor networks. Each node requires local information about its neighbours but no knowledge of global topology in order to make routing decisions. The local information held about neighbouring nodes is held in a linked list data structure, which is limited in size to the number of neighbours the node has. The information required for each neighbour is the pheromone trail, and its hop counts to each sink. The distributed implementation has the advantage of requiring the same amount of information regardless of network size, and only relies upon the number of neighbours, helping the scaleability of the protocol.

Each choice of the next node to travel to is probabilistic, so no routing tables are required, reducing the amount of information needing to be stored. Evaporation occurs periodically, with each node evaporating the pheromone trail amounts stored in its neighbours list at equal intervals.

## 4.4 Simulation Setup

The protocol has been developed using ContikiOS, an open source operating system for the internet of things [36]. Simulations were performed using COOJA [93], a network simulator for Contiki, using emulated Sky motes with UDGM radio medium. The protocol is compared with flooding as a base line, which was also implemented in ContikiOS and simulations performed with COOJA. Flooding was chosen as it is commonly used, simple protocol that the ACO based protocol should at least improve upon in order to be successful. A more efficient approach from literature was not chosen, as these are often not made available in ContikiOS to directly compare with.

### 4.4.1 Network Configuration

All experiments were performed using a square grid of nodes of size $n \times n$, where $n \in \{5, 7, 9, 11, 13\}$. The distance between nodes and transmission range is set up such that only horizontal and vertical neighbours can communicate. Experiments have been carried out using two sources and two sinks, with the sources and sinks fixed in the corners of the grid. The two sources are in the lower two corners and the two sinks in the upper two corners. An example configuration is shown in figure 4.1. The configuration chosen at the start remains constant until the end of the experiment.

Forward ants were launched from the sources periodically using a timer, both sources launch ants at the same rate. Approximately 500 different simulations of each network setup will be performed on the simulator. Each run is finished when 115 cycles have been completed; insufficient cycles will lead to that run being excluded from the results. This is to ensure that the protocol is able to continually deliver data from sources to sinks over an extended period of time.

### 4.4.2 Parameters

The ACO based protocol involves setting a number of parameters, which are set once at the start and are constant between nodes. The process of choosing the next node to travel to is dependent on three parameters. These are the weighting of pheromone trail amount between nodes, average hops, and hop range. Pheromone indicates that previous ants have followed this route before, with higher values being preferred as this indicates more successful routes. The use of this parameter encourages the formation of successful routes based on previous performance. The average hops directs ants generally towards the sinks, so nodes with lower average hops are preferred. Hop range was chosen for the probability function as this will help to form the backbone; a lower hop range indicates the *middle* route between sinks, and so it is more likely

| Name | Value |
|---|---|
| Pheromone Impact | 4 |
| Average Hops Impact | 1 |
| Hops Range Impact | 4 |
| Backbone Length Impact | 1 |
| Route Length Impact | 1 |
| Evaporation Constant | 0.8 |

Table 4.2: Simulation parameters for ACO based Routing in Many-to-Many WSN

for ants to meet. This will encourage ants to travel towards each other. Each input is weighted using the input parameters, which were set to be 4, 1, and 4 respectively, for all network sizes. These parameters were found to be most effective through repeated testing. These are used for both variations of the protocol tested. However, the parameters can be easily changed to suit different networks with different requirements.

The second set of parameters to be set are largely used for pheromone trail updates. The amount of pheromone laid is dependent on the success of the route followed by the forward ants. Backbone length is a factor for pheromone updates; longer backbones are encouraged as this means better use of data aggregation, fewer messages sent, and better scaleability, so more pheromone is laid. Backbone length impact was set to be 1. Additionally, route length is also used when updating the trail change, shorter routes are more desirable, however proportionally with longer backbones.

The final parameter to consider is evaporation constant, a value that determines how quickly pheromone trail evaporates between nodes. The evaporation constant was set to be 0.8, which leads to the pheromone value dropping very little between cycles, leading to fast convergence on a route. It was found that despite little evaporation happening cycle to cycle, the protocol was still able to converge on an efficient route.

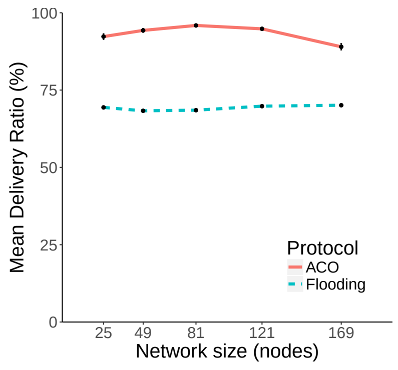
## 4.5 Results

The results for the two variations of the ACO based protocol for routing in many-to-many WSN are presented here. The performance metrics that are being considered are:

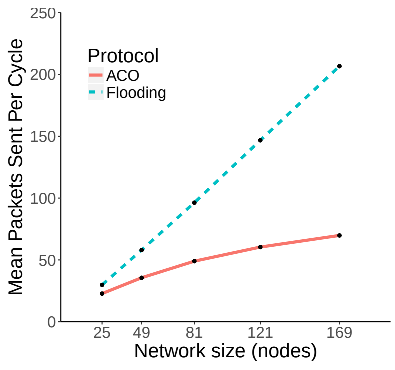
- **Delivery Ratio**: The ratio between total messages sent and total messages successfully received. With the setup presented here of two sources and two sinks, this is the ratio between all messages sent launched from both sources and all messages received at both sinks.
- **Number of nodes involved**: The mean number of nodes involved in the communication from sources to sinks, as a percentage of total number of nodes in the network.
- **Packets Sent per cycle**: The number of packets sent in a cycle. This includes packets sent by both forward and backward ants, but excludes packets sent during the set up phase.
- **Backbone length**: The mean number of nodes in the backbone formed. A longer backbone is considered more successful, as benefits from the advantages of data aggregation to a greater extent.
- **Backbone Convergence**: The converged backbone for an experiment is considered to be the backbone that most routes follow for that experiment. A higher percentage of experiments following its converged route is more successful as this shows consistency.

### 4.5.1 Base ACO Protocol

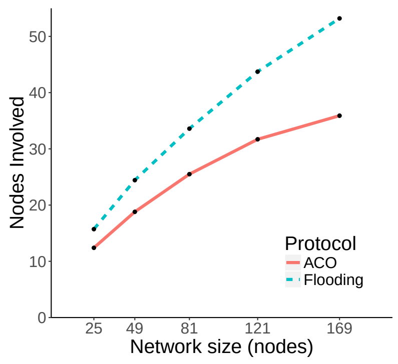
Figure 4.2 shows that the base ACO routing protocol achieves a high mean delivery ratio for all network sizes, indicating the consistent delivery of messages from both sources to both sinks. The average delivery ratio is 93.3% between network sizes. Additionally, the standard error in the delivery ratio shown in figure 4.2a is very small, which shows consistency between experiments



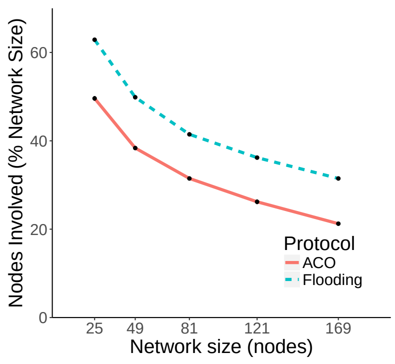
(a) Mean Delivery Ratio



(b) Mean Packets Sent Per Cycle



(c) Nodes Involved in Routing from all sources to all sinks



(d) Nodes involved in routing as a percentage of network size

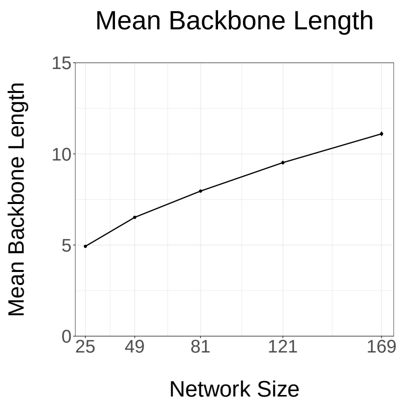
Figure 4.2: Delivery Ratio and Nodes Involved for the ACO based protocol

performed at each network size. The delivery ratio remains similar between network sizes, but falls slightly at the largest network size of 169 nodes. This could be due to difficulty in forming the initial backbone, or just due to the necessity for more packets needing to be sent overall leading to more failures in delivery. The timer used for combining ants into a backbone remained constant between network sizes, so delivery ratio may be improved with increasing this timer. Despite this, the protocol still has a high delivery ratio consistently between network sizes. Compared with a simple flooding protocol, the ACO protocol achieves consistently better delivery ratio indicating that it was more successful at packet delivery.

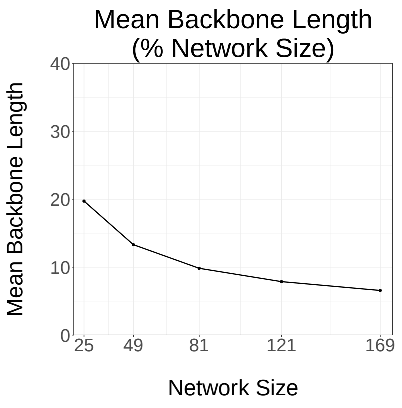
When comparing the ACO with flooding in the figure 4.2, the equivalent network tested with a flooding protocol achieved on average lower delivery ratios and more packets sent overall. The flooding protocol achieved an average delivery ratio of 69.2%, which is significantly lower than the ACO protocol. This indicates that the ACO protocol improves over both flooding and ACO with ant memory for many-to-many routing, and that the protocol is reliable for a number of network sizes. The packets sent increases with network size, but at a significantly slower rate that the equivalent increase of the flooding based protocol. This shows that the ACO protocol is scaleable, as it becomes more efficient with larger networks in terms of packets sent. A similar trend is seen with the nodes involved in route, shown in Figures 4.2c and 4.2d.

Figure 4.3b shows the mean number of packets sent in each cycle. Each cycle starts when ants are launched from the sources and ends when backward ants are received. The packets sent slowly increases with larger networks sizes for the ACO protocol. This is expected, as more packets are required to traverse the network when it is larger. The packets sent does not increase linearly, indicating that the advantages introduced by the use of a backbone have more impact as the network size increases, showing the scaleability of the protocol. Similarly, when looking at the nodes involved in routing, figure 4.2c, the ACO protocol uses more nodes as the network size increases, but this increase is not constant, again indicating scalability as the protocol becomes more effective as the network becomes larger. Figure 4.2d shows that with an increase in network size, the number nodes involved as a percentage of network size decreases. This is also representative of the scalebility of the protocol, as the route to the sinks becomes proportionally shorter as the network size gets larger, using less of the nodes in the networks. Both the packets sent and the nodes involved in routing is consistently less than flooding, especially as the network size gets larger.

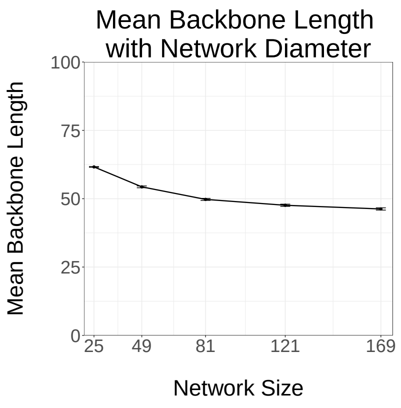
When analysing the role of the backbone in routing for the ACO protocol, it was found that the backbone length increased in larger network sizes, as shown in figure 4.3a. This is expected, as the larger network size necessitates a larger number of hops from source to sink. However, when looking at the



(a) Mean backbone length



(b) Mean Backbone Length as a Percentage of Network Size

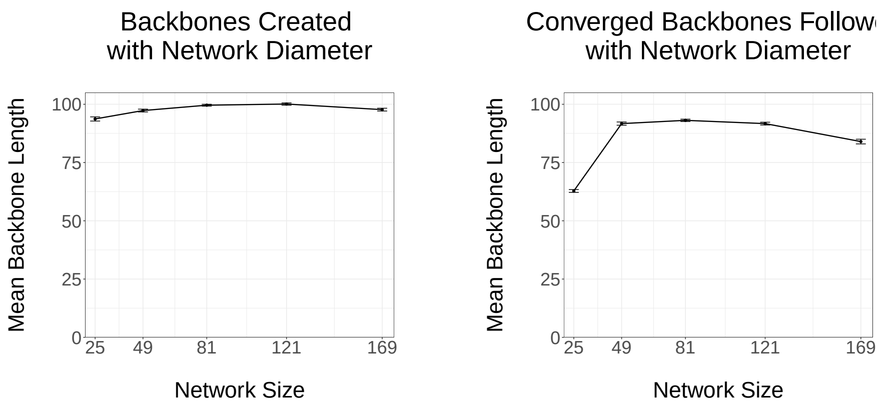


(c) Backbone length with increasing network diameter, as a percentage of network size

Figure 4.3: Backbone Analysis of the protocol

backbone length as a percentage of the total network size, this now decreases with larger networks. This shows the scalability of the protocol, as larger networks benefit from the data aggregation of the backbone to a greater extent. This is collaborated with figure 4.3b, showing that the packets sent increases with network size, but the size of this increase is less with larger networks. When considering the length of the backbone in proportion to the maximum distance between a source and a sink, figure 4.3c, we see that a similar decrease happens, again indicating the scalability of the algorithm.

Figure 4.4a shows that the backbone is successfully formed the majority of the time for all network sizes. The success rate is lower for smaller networks,



(a) Backbone Creation (b) Backbone Convergence

Figure 4.4: Further Backbone Analysis

which may indicate that ants tend towards the sinks more often when they are relatively close. There is also a drop in the creation of backbone with the largest network size, which may be due to similar issues regarding delivery ratio. In some cases the backbone is formed more than once in a cycle, leading to a percentage higher than 100%, however this is a very rare occurrence and is not considered to be affecting the running of the protocol in the vast majority of cycles.

Backbone convergence is investigated in figures 4.4b and 4.5. Figure 4.4b shows the percentage of cycles that follow the converged backbone route. Similar issues occur with the smallest network size, however a large proportion of cycles follow the converged backbone for other sizes. There is a small drop for larger network sizes, consistent with other metrics. Figure 4.5 shows that experiments tend to converge on a backbone at approximately the third cycle. Ants then tend to follow that converged backbone through subsequent cycles to the end of the experiment, indicating a persistent route. The converged backbone is defined as the backbone that is followed by ants for the majority of the experiment. The smallest network size of 25 nodes has more variation in convergence than larger networks as well as a lower convergence rate, which may be caused by similar issues involving backbone creation shown in figure 4.4a.

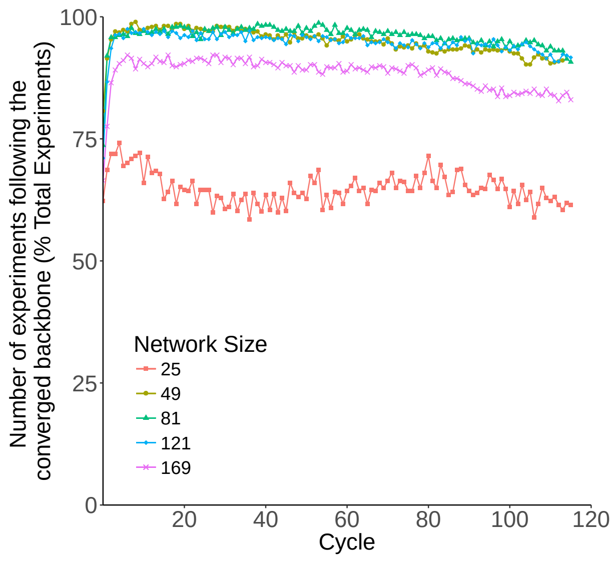


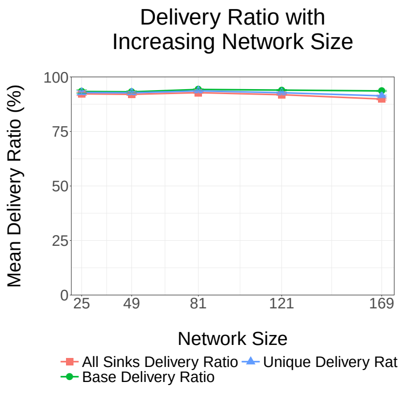
Figure 4.5: Number of Experiments following the converged backbone as a percentage of total experiments.

### 4.5.2 ACO with No Ant Memory

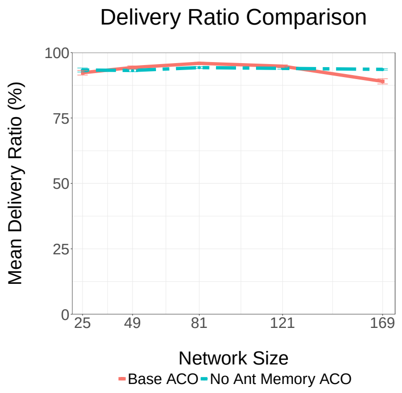
The same set of experiments were performed for the version of the ACO protocol with no Ant Memory, and the results are shown in this section. The base ACO protocol is compared with the no ant memory variation of the protocol, to investigate the effect removing ant memory has upon performance.

Variations on the measurement of delivery ratio is shown in figure 4.6a. There is not a great deal of difference between the three measures of delivery ratio that are investigated, indicating that most times a message is received on a sink, that the other sink also receives a message, and that message consists of ants from both sources. This shows that the protocol is successfully delivering messages from both sources to both sinks. The error bars on each point are very small, indicating that the protocol is consistent over all simulations.

The no ant memory variation of the protocol is compared with the base ACO protocol in figure 4.6b. It can be observed that both the base protocol and the variation with no any memory have very similar delivery ratios. It

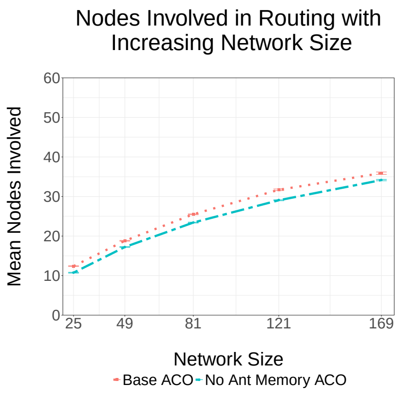


(a) Mean Delivery Ratio of No Ant Memory ACO

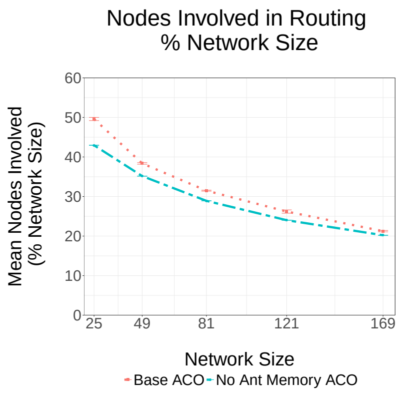


(b) Comparing mean delivery ratio of base ACO and no ant memory ACO

Figure 4.6: Delivery Ratio for the ACO based protocol with no ant memory



(a) Nodes Involved in routing



(b) Nodes involved as a percentage of network size

Figure 4.7: Nodes Involved for the ACO based protocol with no ant memory

can be argued that the no ant memory variation is slightly more consistent than the base ACO protocol, as it doesn't have a decrease in performance for the largest memory size. This could be explained by the ants not having to carry the larger routes in memory necessary for larger network sizes, leading to improved performance. This also indicates that the effects of removing ant memory does not lead to a significant amount of "loops" forming in the route, as there is no reduction in performance compared to the base ACO protocol.

Figure 4.7a shows the nodes involved in routing messages from sources to sinks. The trend travels upward, with more nodes being involved in routing with larger networks. This is expected, as as the network size becomes larger,

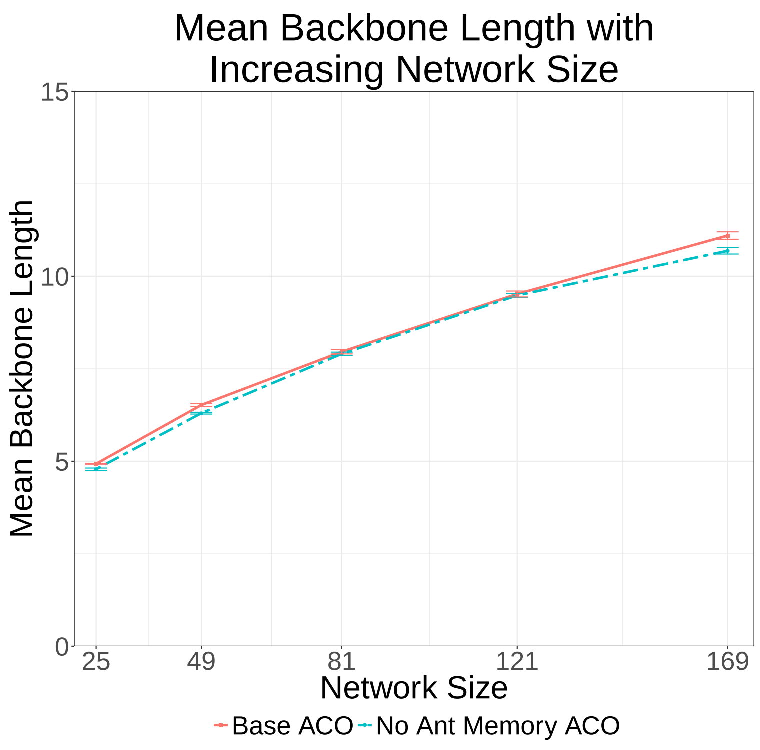


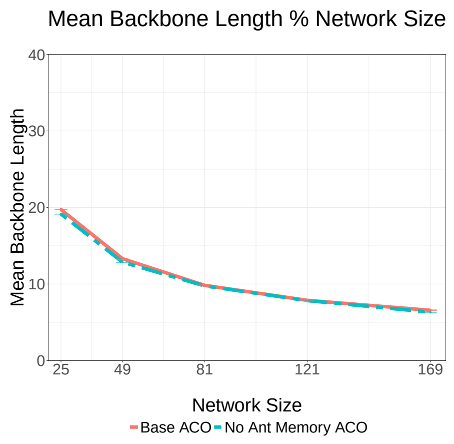
Figure 4.8: Mean Backbone Length No Ant Memory ACO

the minimum possible distance increases. The error on the points are small, indicating that the number of nodes involved is consistent between simulations, as well as being consistent between cycles. Looking at the number of nodes involved as a percentage of network size, 4.7b, it can be observed that with a larger network, proportionally fewer total nodes are involved. This decrease in the percentage of nodes as networks get larger shows the scaleability of the protocol. Nodes involved when considering only cycles where all sinks receive from all sources is a slightly lower figure, this could indicate that there are a small number of cycles without a backbone that increases the average nodes involved figure.

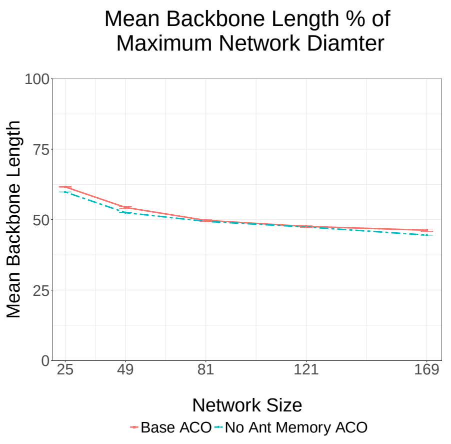
When comparing the base ACO protocol with the no ant memory version, it can be seen that both versions of the protocol follow a similar trend of increasing number of nodes involved with larger networks, but decreasing the percentage of nodes involved as a percentage of network size. For both methods, the no ant memory version uses less nodes in routing than the base ACO protocol.

Figure 4.8 shows the mean backbone length of the no ant memory version of the protocol. The backbone length increases with network size, as expected as a longer backbone is required to traverse the network. Figure 4.8 also compares the backbone length of the no ant memory protocol with the base protocol. Generally there is very little difference between the two versions of the code and backbone length, showing that removing the ant memory doesn't have an adverse effect on the effects of data aggregation through the backbone. Similarly, figures 4.9a and 4.9b show that there is very little difference for the percentage of network size and network diameter measures.

As a percentage of network size, figure 4.9a, it can be shown that the backbone length decreases with network size. The backbone length as a percentage of the maximum distance from a source to sink (network diameter) remains approximately constant with increasing network size, at around 50%. This figure indicates that most of the route is usually backbone ants, indicating a good level of data aggregation in the network, as well as a consistent one.

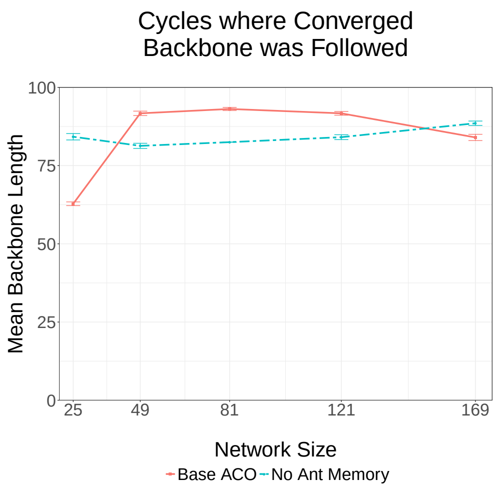


(a) Mean Backbone Length as a percentage of network size

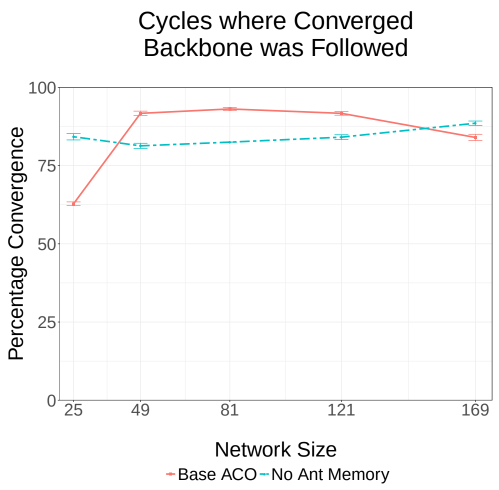


(b) Mean Backbone Length as a percentage of network diameter

Figure 4.9: Comparison of Backbone Length for no ant memory ACO protocol



(a) Number of times the backbone was created over the simulation



(b) Percentage of cycles where the converged backbone was being followed by ants.

Figure 4.10: Analysis of Backbone Creation for no ant memory ACO protocol

It can be shown from figure 4.10a that a backbone was consistently created for most cycles over the course of the simulation. Additionally, figure 4.10b shows that most cycles follow what was calculated to be the converged backbone, that is the most commonly followed backbone. This shows consistent convergence on a route. Figure 4.10a also shows that there is very little difference between the no ant memory protocol and the base ACO protocol in terms of how often the backbone was created. The no ant memory version of the protocol was more consistent in following the converged backbone regardless of network size, as shown by figure 4.10b.

The convergence of routes for the no ant memory code reasonably consistent, as shown in figure 4.11. Most cycles are following the converged routes, with less variation between network sizes as compared to the base ACO protocol.

## 4.6 Conclusion

In this chapter, a novel protocol for routing in wireless sensor networks was presented. It is successful in continuous data delivery from all sources to all sinks in grid based networks when considering a number of delivery metrics.

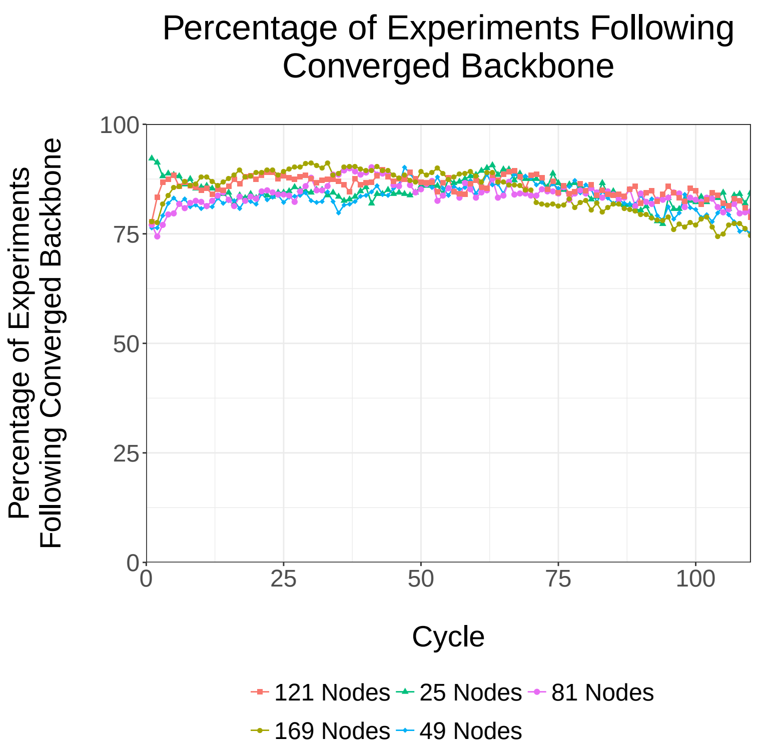


Figure 4.11: Convergence of no ant memory routes

Delivery ratio is consistently high with increasing network sizes, indicating the scaleability of the protocol, as the protocol remains successfull even when scaled up to a larger network. Nodes involved and packets sent in routing also remains low, which should lead to increased network lifetime as minimal number of packets are sent. Additionally, results show that the protocol tends to become more efficient as network size increases, as proportionally fewer nodes are involved. This is achieved through the technique of a shared path or backbone, which increases scalebility of the protocol through both the use of data aggregation as fewer packets sent are necessary, and minimising the path length needed to reach all sinks in the same route.

The protocol presented makes use of the meta heuristic Ant Colony Optimisation in order to create routing paths online throughout data delivery. Each node only makes localised decisions, with no greater knowledge of the network topology required. The ACO protocol presented in this chapter is novel in that it uses no concept of ant memory, which has previously been required for routing protocols using the algorithm. This has the effect of reducing the size of the payload needed to be carried for routing purposes. There is no impact in performance, and it could be argued that removing the ant memory improves delivery ratio for larger networks.

Chapter 5

# Fault Tolerant ACO Routing in Many-to-Many Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are used in a wide variety of environments for a great deal of applications, each coming with their own problems. As sensor nodes are generally limited in terms of battery life, it is likely that some sensor nodes will run out of energy during operation of the network, and so cannot be used in routing. Links between nodes may fail for a number of reasons, including energy problems with the node reducing transmission power, even if the node itself does not run out of energy completely. Environmental factors such as wind may move sensor nodes out of range of its neighbours, or other environmental events may block sensors from properly sending messages to its neighbours. For some applications of wireless sensor networks, such as extreme environments like battlefields, it may be common for a sensor node to be destroyed by adverse conditions. If a WSN has been set up with the goal of monitoring an environment that humans cannot access easily, it is likely to be difficult for sensor nodes to be fixed, recharged, or moved back into place. For this reason, it is necessary to develop fault tolerant protocols that are able to detect and recover from faults in the network during operation of routing protocols without the need for human intervention.

In this chapter a fault tolerant variant of the ACO based protocol introduced in the previous chapter is presented (FT-ACO). FT-ACO has a number of variations in order to achieve fault tolerance in the event of node failures within a WSN. FT-ACO uses the same base version of the routing protocol presented in the previous chapter, with some extensions to enable fault recovery. Each ant makes localised decisions on where to travel to next, with the goal of travelling a shared backbone in order to minimise messages sent, optimising data aggregation, and thereby increasing network life time. To address the issue of random node failures, a window-based active fault detection scheme which uses a *novel* type of ants, referred to as “beacon ants” is proposed. The purpose of beacon ants is to detect potential failed nodes. In addition to this concept, a reliable unicast callback fault detection scheme is also investigated. Utilising the pheromone evaporation aspect of ACO, the protocol adapts to failures in an online fashion during the running of the routing protocol. The concept of “targeted evaporation” of links between nodes in introduced, which will encourage ants to avoid failed nodes in their decisions. This enables fault recovery without requiring expensive network wide broadcasts, setting up multiple routes ahead of time, or finding new routes from scratch. The protocol will remain fully distributed, and is able to recover from faults that occur during the operation of the protocol in networks with multiple sources and multiple sinks. The protocol is able to achieve more than 80% delivery ratio with 5% node failures while remaining scaleable compared to other approaches requiring periodic topology maintenance.

In summary, the solution proposed is for the problem of fault tolerant routing in many-to-many wireless sensor networks. Results show that the protocol is able to recover from node failures whilst still being able to successfully deliver packets from sources to sinks in cost efficient routes. The contributions made in this chapter are:

- An ACO based fault tolerant routing protocol for many-to-many wireless sensor networks (FT-ACO) that is an extension of the previous ACO based protocol.
- A distributed implementation of FT-ACO.
- Simulations of the distributed implementation with varied node failures in various sized networks.

## 5.1 Problem Specification

The problem to be solved can be stated as follows: Given a WSN, a number of sources $s$ and a number of sinks $S$, find a minimal set of edges such that data can be routed from multiple sources $s_1, \ldots, s_k$ to multiple sinks $\Delta_1, \ldots, \Delta_l$ even in the presence of node crashes.

Previous fault tolerant works rely upon centralised routing algorithms in order to reroute around failures, or the setting up of multiple routes ahead of time in order to switch to an alternate route in the case of node failures. This has the potential to be inefficient, requiring large numbers of packets sent or the amount of information needed to be stored on each sensor node. Other protocols will require constant reinforcement of alternate routes or network wise broadcasts to update topology information, which again is expensive in terms of messages sent. To avoid these issues, the ACO based protocol is used where only local information about neighbours is required for fault recovery. Targeted pheromone evaporation is used to naturally reroute around failed nodes without the whole network needing knowledge of the failure.

The ACO based protocol developed in the previous chapter is extended to become a fault tolerant ACO based protocol, FT-ACO. Three different types of ants are used in fault-tolerant routing: (i) Forward ants are ants launched from sources in order to carry sensing data to the sinks, (ii) Backward ants are launched from the sinks following the successful delivery of a forward ant, and they follow the forward path in reverse, back to the source nodes, updating the pheromone trails between nodes and reinforcing successful routes. Finally, (ii) in order to ensure fault tolerance, *Beacon Ants* are introduced, which are used to provide active fault monitoring throughout the network.

The novelty of the approach is the integration of fault tolerance in ACO. Beacon ants have been introduced with the goal of identifying failed (i.e., crashed) nodes. These ants are periodically broadcast from each node to its neighbours, with the periodicity of beacon ants enabling a balance between number of packets sent and crash detection latency. A window of beacon counters is developed that keeps track of the number of beacon ants received, which has the impact of reducing the chance of false positives. When a node fails to receive expected beacon ants from a neighbour, it will initiate fault recovery mechanisms. Fault recovery is provided through the targeted evaporation of the pheromone between nodes; when a node has been identified as failed by a beacon ant, the pheromone is partially evaporated, with evaporation amount varied based on the likelihood of failure, calculated from the window of beacon counts, reducing the chance that an ant will travel to it proportionately. This is akin to the case of achieving the *strong eventual accuracy* property of a failure detector.

## 5.2 Description of the Fault Tolerant ACO Protocol

The fault tolerant ACO based protocol is an extension of the protocol as described in the previous chapter. Changes have been made to implement forms of fault detection, in addition to a fault recovery process to route around failed nodes. The fault tolerant implementation of the ACO protocol is described in this section. It consists of both a set up phase and a continuous data delivery phase as in the previous protocol, as well as a continuous fault detection mechanism. FT-ACO is based upon the variant of the ACO based protocol for many-to-many routing with no ant memory.

There are again some variations to FT-ACO, in order to investigate the effectiveness of different forms of fault detection. These are:

- Active fault detection using beacon ants. Beacon ants are periodically broadcast to find local node failures.
- Active fault detection using both beacon ants and reliable unicast.
- Passive fault detection using only the base ACO protocol

**Set up Phase**

The set up phase occurs once throughout the running of the protocol and does not need to be run again if node failures occur. It is the same setup phase as described in the previous chapter.
Following this set up phase, the main running of the protocol begins. This consists of continuous delivery of data from the sources to the sinks.

**Data Delivery** The data delivery phase of the protocol remains the same as the data delivery phase of the base ACO protocol with no ant memory as described in Chapter 4, until the forward ants reach the sink. At this point there is a change in how the pheromone is updated in order to take into account the number of ants that form the backbone. This change was made to prioritise routes with more ants comprising the backbone, to encourage more successful routes.

1. Once the ants reach the sinks, they are transformed into backward ants. If a forward ant has reached a sink without having formed a backbone, backward ants are not generated, so that only paths where a backbone is formed are reinforced. Backward ants travel to the *prev* node, as marked by the forward ants, travelling back to the source nodes. The first backward ant to reach the backbone continues along it, such that

only one backward ant traverses the backbone, with subsequent ants halting. At the end of the backbone the backward ant is split and continue to the sources using a broadcast. As the backward ants travel, they update a value in the neighbours list, $\Delta\tau$, which represents the change in pheromone value, using:

$$\Delta\tau_t = \Delta\tau_{t-1} + \frac{A}{l_r} + \frac{B \cdot l_b}{l_r} + (C \cdot |ants|) \tag{5.1}$$

where $\Delta\tau_t$ indicates the change in pheromone value at current time $t$, $l_r$ represents the total forward route length, including the multiple routes to and from the backbone, $l_b$ is the length of the backbone, $|ants|$ is the count of ants that combined to the form the backbone, and $A$, $B$, and $C$ are the weights of these parameters. The number of ants represents a change from the original ACO based protocol, and was introduced in order to prioritise routes for which more than one ant has been combined to form the backbone. If a failure occurs, it is more likely that the backbone will form, so this was more necessary to include.

2. The pheromone value is periodically updated using the equation:

$$\tau_t = (\rho \cdot \tau_{t-1}) + \Delta\tau_t \tag{5.2}$$

where $\tau_t$ represents the pheromone value at time $t$, $\rho \in 01$ indicates the rate of evaporation, and $\Delta\tau$ is the trail change value.

3. Increment the cycle count by 1, $t$ by 1, and launch one ant from each source node to start the next cycle.

**Fault Tolerance with Beacon Ants**

The fault detection protocol using beacon ants and targeted evaporation runs while the main protocol is finding routes and delivering data. This is described below:

1. At the start and end of each cycle, beacons ants are broadcasted from each

node. Each node keeps track of how many beacons that it has broadcasted, as well as how many broadcasts it has received from neighbours. To do this, each node associates each neighbour with a window of size $w$ keeping track of how many beacon ants have been received from that neighbour.

2. When a broadcast is received from a neighbour, a counter associated with that neighbour is incremented

3. When a node sends a broadcast, it increments its own counter indicating how many broadcasts it has sent. Additionally, all windows of all the neighbours are updated. This involves inserting the counter of broadcasts received from that neighbour at the start of the window, while shifting the rest of the window values down.

4. When the window has been fully populated, then fault detection begins. To do this, an evaporation amount is calculated using

$$\rho_{ft} = D - \frac{q}{w} \tag{5.3}$$

   where $\rho_{ft}$ indicates the evaporation amount that is used if there is a fault detected, $D$ is the highest number of adjacent values in the window that are the same, and $w$ is the window size.

5. If $\rho_{ft} < 1$, this is considered to be a potential fault as the received beacons from a neighbour has stagnated. Pheromone trail to this node is evaporated using

$$\tau = (\rho_{ft} \cdot \tau_{prev}) \tag{5.4}$$

   where $\tau$ represents the new pheromone value, $\tau_{prev}$ is the previous trail and $\rho_{ft}$ indicates the evaporation amount calculated in 5.3

6. The runicast method of fault detection makes use of the the reliable unicast method of sending packets in ContikiOS. This returns a “timed

out" function when a message fails to send, using this the trail is then evaporated using

$$\tau = (\rho_{rft} \cdot \tau_{prev}) \tag{5.5}$$

where $\tau$ represents the new pheromone value, $\tau_{prev}$ is the previous trail and $\rho_{rft}$ indicates the evaporation amount used to reduce trail. This will be different to the beacon ants evaporation amount as the equation is different, and also to account for different false positive rates.

### Fault Tolerance with Passive Pheromone Evaporation

This method of fault tolerance represents no change in the base ACO protocol, however the evaporation constant is varied in order to investigate how well ants naturally recover from failures using evaporation.

The fault tolerant protocol ACO based routing protocol is further explained in Algorithms 5, 6, 7, 8, and 9.

**Algorithm 5** Description of the Fault Tolerant ACO based Protocol for Many-to-Many WSN

```
 1: procedure ACO-PROTOCOL
 2:     $cycle \leftarrow 0$
 3:     $t \leftarrow 0$
 4:     $cycle_{max} \leftarrow C$
 5:     while $cycle < cycle_{max}$ do
 6:         BeaconAntSend()
 7:         $messages \leftarrow 0$          ▷ The number of messages received by this node
 8:         if $node \in sources$ then
 9:             Choose next node $n \in neighbours$ with Eqn (4.1)
10:             $prevnode \leftarrow 0$
11:             Send message from $node$ towards $n$, with packetbuf $route_a$
12:         if $node$ receives a forward ant then
13:             ForwardAntReceived()
14:         else if $node$ receives a backward ant then
15:             BackwardAntReceived()
16:         else if $node$ receives a beacon ant then
17:             BeaconAntReceived()
18:         Evaporate pheromone between $node$ and $n$ with eq:pheromone-update
19:         BeaconAntSend()
20:         $cycle_{max} \leftarrow C + 1$
21:
```

**Algorithm 6** Description of Forward Ant Protocol

```
 1: procedure ForwardAntReceived
 2:     $prevnode \leftarrow from$                    ▷ The node we received the forward ant from
 3:     $messages \leftarrow messages + 1$
 4:     if $node \in sinks$ then
 5:         Launch backward ant; send message from $node$ towards $prevnode$
 6:     else
 7:         if Backbone has not been formed then
 8:             if $messages 1$ then
 9:                 Set a timer for $s$ seconds.
10:                 while Timer is not yet finished do
11:                     Wait
12:                 if $messages \neq |sources|$ then
13:                     $messages \leftarrow 0$
14:                     Choose next node $n$ with probability (4.1)
15:                     Send message towards $n$
16:                 else
17:                     Aggregate data
18:                     $messages \leftarrow 0$
19:                     Choose next node $n$ with probability (4.1)
20:                     Send message towards $n$
21:         else
22:             if Backbone has been split then
23:                 Choose next node $n$ with probability 5.2)
24:                 Send message towards $n$, with packetbuf $route_c$
25:             else if $\exists n \in N(n) \rightarrow (\forall i \in sinks \rightarrow h_{n,i} < h_{node,i})$ then
26:                 Choose next node $n$ with probability (4.1)
27:                 Send message towards $n$
28:             else                                                  ▷ Split the backbone
29:                 Send $|sinks|$ messages from $node$ to the neighbours with the smallest $h_{n,i}$
    for each sink
30:
```

**Algorithm 7** Description of Backward Ant Protocol

```
 1: procedure BackwardAntReceived
 2:     $bwmessages \leftarrow bwmessages + 1$
 3:     Update change in pheromone with equation 5.3
 4:     if $node \in sources$ then                        ▷ Backward ant route complete
 5:         Stop ant, send no more messages.
 6:     else
 7:         if $node$ is the first node to reach the backbone then
 8:             if $bwmessages > 1$ then
 9:                 Stop ant, send no more messages
10:         else if $node$ is the node where the backbone was created then
11:             Broadcast messages to $node \in N(n)$
12:         else
13:             if $nextnode = from$ then
14:                 Send backward ant toward $prevnode$
```

**Algorithm 8** Description of BeaconAntSend

```
 1: procedure BeaconAntSend
 2:     $beaconsSent \leftarrow beaconsSent + 1$
 3:     Broadcast Beacon Ant
 4:     Update windows of all neighbours with $receivedCount$
 5:     Let $j$ be the number of adjacent values from the start of the window with the same
    count
 6:     if (D - ($j$ / $|W|$) ¡ 1 then
 7:         Evaporate trail between this node and faulty neighbour with eqn 5.7
```

**Algorithm 9** Description of BeaconAntReceived

```
1: procedure BeaconAntReceived
2:     $receivedCount \leftarrow receivedCount + 1$
```

### 5.2.1 Changes to Base ACO protocol

The Fault Tolerant variant of the ACO routing protocol for many-to-many wireless sensor networks is based on the original protocol, however introduces some new concepts. The additions made to the protocol are summarised in this section.

- Beacon ants: The introduction of a new type of ant in the protocol, the beacon ant, is the main change for the Fault Tolerant variant. The beacon ants are periodically launched by all nodes, and all nodes keep track of the beacon ants they receive from their neighbours.
- Fault Detection using reliable unicast: An additional avenue of fault detection has also been introduced, exploiting the reliable unicast mechanism developed for Contiki OS.
- Targeted Pheromone Trail evaporation: On detection of a failure, nodes will evaporate the pheromone trail between itself and the failed node.
- Updated pheromone trail update: The equation to update pheromone trail has been altered in order to take into account the number of ants forming the backbone, in order to prioritise routes formed of ants from multiple sources.

## 5.3 Experimental Setup

FT-ACO was implemented using ContikiOS, an open source operating system created for the Internet of Things [36]. ContikiOS has a simulator associated with it called COOJA, that can be used to simulate wireless sensor networks [93]. All simulations were carried out with COOJA using emulated Sky motes with the UDGM radio medium and the null RDC driver. The null RDC driver was used as this means that the radio is constantly on, leading to fewer failures in transmissions. This means that the link failures that occur can be controlled more easily, as there should be fewer unplanned failures. The MAC driver used was CSMA, the default mechanism for COOJA, as this retransmits in the case of collision, which again should lead to fewer unplanned failures in transmission.

For each simulation, approximately 150 runs of each setup are performed. For each simulation, at least 115 cycles will have been completed, in order to see how the protocol performs long term. Simulations where insufficient cycles have completed or insufficient failures occurs were excluded.

The experimental setup in terms of the OS the protocol was written in and the simulation software used has remained constant with the base ACO routing protocol. This will allow comparisons to be made between the two protocol variants. The main change made is the failure model, where a number of nodes will be simulated failing for experiements carried out.

### 5.3.1 Network Configuration

All experiments were performed on a square grid of nodes with varying sizes $n \times n$, where $n \in \{5, 7, 9, 11, 13\}$. The distance between all nodes is constant, and the transmission range and power setup such that each node can only communicate with its horizontal and vertical neighbours. The number of sources and sinks as well as the placement of these sources and sinks are varied for some experiments, but for most experiments there are two sources and two sinks placed in the corners of the network. Forward ants are launched from the sources at the same time for each cycle.

The network configurations have largely remained the same as with the base ACO protocol, so as to allow comparisons to be made between them.

### 5.3.2 Failure Model

Node failures are simulated by randomly removing nodes from the simulation. At the start of each cycle, each node has a percentage chance of being removed, with this probability varying based on the total number of failures required for the experiment. Additionally, a number of experiments were performed with predetermined "patterned failures". These failures are a set of neighbouring nodes being removed at the same timed, in order to explore the response to patterned failures. This may occur, for instance, when neighbouring nodes are destroyed by the environment. When a node is removed from the network, its neighbours will not be able to send any packets to it.

## 5.4 Results

To determine the success of the fault tolerance many-to-many protocol, the following performance metrics will be examined:

- **Delivery Ratio**: The ratio of packets sent to packets received on the sinks. Due to the nature of the protocol, when there are node failures there is the potential for sinks to receive packets multiple times. This could be caused, for instance, by multiple splittings of the backbone leading to multiple ant paths, or both ants from both sources being received on the same sink, without forming a backbone. For this reason, several forms of the delivery ratio are used, in order to investigate how successfully each sink receives packets originating from both sources.
  - Base DR: Total received over total sent, whether at backbone has formed or not. This is just a simple overlook of the delivery ratio in its simplest form.
  - Unique DR: This delivery ratio only counts one packet receiving on a sink, so no repeat visits are included in this number.
  - All Sinks DR: This delivery ratio measure only considers cycles where both sinks receive a packet from a formed backbone, without

including any repeated visits on sinks. This measure is used to look at how often the protocol is able to successfully deliver to all sinks, i.e. how many ideal cycles occur.

- **Number of Nodes involved per cycle**: This metric looks at how many nodes are involved in sending packets from sources to sinks in a particular cycle. It simply counts how many nodes form the routes from sources to sinks. Again, two variations are used.
    - Base: Total nodes involved in each cycle, for routes where the backbone formed and where they did not. This may look lower than the backbone routes, as if a backbone is not formed, not every sink will receive a packet. Additionally, the backbone routes are able to deliver messages from both sources to both sinks, however a route with no backbone will only be able to deliver packets from a source to a single sink. The equivalent level of sink delivery would require two different routes, which does not occur here.
    - All Sinks Nodes involved: A measure of how many nodes are involved only for the routes where all sinks receive from all sources, i.e. a backbone has been formed of ants from all sources.
- **Packets sent per cycle**: This metric counts how many packets are required to send an ant from the sources to sinks in each cycle. It is a similar measure to nodes involved, as generally for the protocol here each node will only send a message twice; once forward and one backward. However, this measure will also include the packets sent on unsuccessful routes that are taken during fault recovery. This is interesting to investigate as other fault tolerant routing protocols will require repeat sending of packets.
- **Backbone Length**: The number of nodes that form the backbone for a particular route. Generally, a longer backbone is considered to be more successful, as means that a greater amount of the total route length is

making use of the aggregated path along the backbone, implying fewer packets sent overall.

- **Fault detection time in cycles**: This is how long the protocol takes to detect a fault in the network, measured in terms of cycles.
- **Recovery time in cycles**: Recovery is defined as the cycle where packets from both sources have been successfully delivered to all sinks using a route with a backbone. This means that recovery time in cycles will be the number of cycles from the cycle the failure occurred in, to the first cycle with both sinks receiving from both sources. This is often 0, for instance in the failed node is not a node on the main path.

The following results look at three different forms of fault detection, as described in earlier sections:

- Beacon Ants
- Beacon Ants and Reliable Unicast Callback
- Passive Fault Detection

In all cases of fault detection, the same fault recovery mechanism is used, as described in the previous section.

### 5.4.1 Parameters

The fault tolerant ACO based protocol has a number of parameters to set. The three factors for choosing the next node are pheromone, average hops, and hop range. Pheromone indicates that previous ants have followed this route before, with higher values indicating more successful routes. The average hops directs ants generally towards the sinks, so nodes with lower average hops are preferred. After the backbone has been split, the probability function changes to take into account the smallest hop count to a sink and has a different weight function associated with it. Hop range was chosen for the probability function as this will help to form the backbone; a lower hop range indicates the *middle* route

| Name | Symbol | Value |
|---|---|---|
| Pheromone Impact | $\alpha$ | 4 |
| Average Hops Impact | $\beta$ | 1 |
| Hops range Impact | $\gamma$ | 1 |
| Post Backbone Hop Count Impact | | 8 |
| Route Length Impact | A | 1 |
| Backbone Length Impact | B | 3 |
| Num. Ants Impact | C | 1 |
| Evaporation Constant | $\rho$ | 0.9 |
| Beacon Ants Constant | D | 1.35 |
| Runicast Constant | $\rho_{ft}$ | 0.2 |

Table 5.2: Simulation parameters for Fault Tolerant ACO based Protocol

between sinks, and so it is more likely for ants to meet. Each factor is weighted using the same input parameters for all network sizes. These parameters were found to be most effective through repeated experimentation

Pheromone update that takes place during the backward ants path also has a number of parameters associated with it. These relate to the relative impact of the length of the total route, the length of the backbone in proportion to the total rote length, and the number of ants. The vales for these weights are 1, 3, and 1 respectively.

The evaporation constant was set to be 0.9, which leads to the pheromone value dropping very little between cycles, leading to fast convergence on a route. The evaporation constant for the fault tolerant parts of the protocol is set to 0.2 for reliable unicast. For beacon ants, the evaporation is variable but depended on a constant $D$ which was chosen to be 1.35 as it was found this led to the best compromise between reducing pheromone and so enabling recovery while also not letting false positives affect normal routes.

### 5.4.2 Experimental Results

Figure 5.1 illustrates that generally the delivery ratio decreases with increased node failures, regardless of the delivery ratio type that is considered. This is expected, as with more failed nodes, there will be more cycles consisting of either fault detection or fault recovery, leading to cycles without ants receiving on the successfully sinks. Additionally, with more nodes failing during the simulation, there is an increased chance that some routes to a sink become impossible, for example if all neighbours around a sink fail, leading to worse delivery ratios.

When considering the two different fault detection methods, beacon ants only and beacon ants and runicast, there are some small variations in how the delivery ratio changes. The fault detection method that makes use of the runicast callback leads to lower delivery ratio at the highest number of node failures. This may indicate that making use of both fault detection schemes leads to too much pheromone trail being evaporated, leading to a break down of routes. When looking at the lowest number of node failures, the delivery ratio for the beacon ants only form of fault detection is lower. This may imply that the runicast method of detecting faults is more effective at lower node failures, perhaps detecting faults faster, or that the higher amount of trail evaporation is more effective when only a few nodes fail in the network.

For both fault detection methods, it can be found that the All Sinks delivery ratio is lower. This could be explained by the fact that this measure does not include cycles where fault recovery is in progress and either the backbone is not created correctly, or only one sink receives a packet. As node failures increase, it can be observed that the delivery ratio for all ants receiving to all sinks lowers, as it becomes more difficult to form an ideal route with a backbone. The base delivery ratio remains higher, as this figure includes routes where no backbone is formed, or only one sink receives an ant.

The difference between the base delivery ratio and the unique delivery ratio, that is where only one received ant per sink is included in the figure, is very small, indicating that situations where there are multiple ants received on

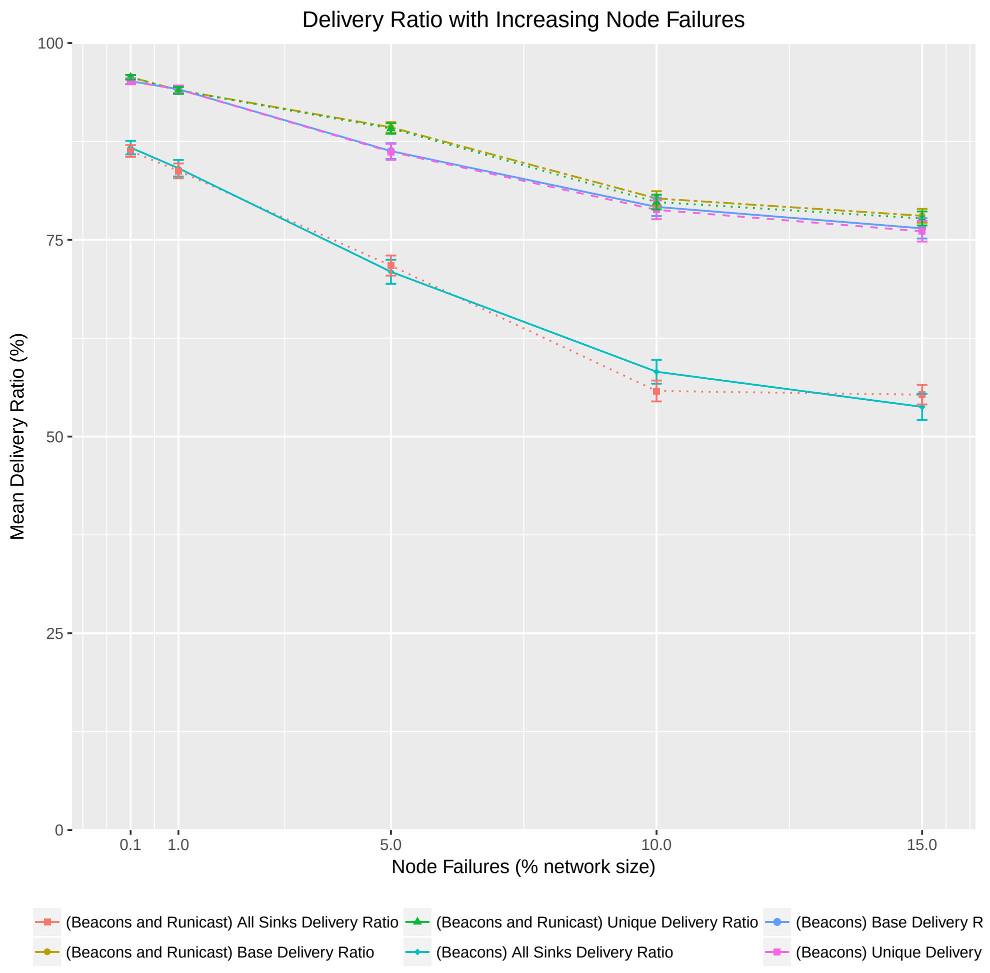
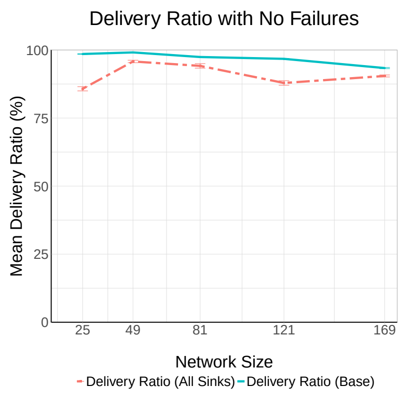
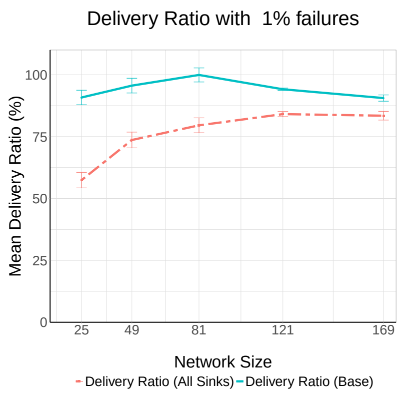


Figure 5.1: Delivery Ratio with increasing node failures for FT-ACO



(a) Delivery Ratio of networks with varying size with no node failures using beacon ants only



(b) Delivery Ratio of networks with varying size with 1% node failures using beacon ants only

Figure 5.2: Delivery Ratio of FT-ACO with no failures and 1% failures

the same sink in one cycle is low for both methods of fault detection.

Figures 5.2a and 5.2b show the delivery ratio of networks of increasing sizes with no node failures and 1% node failures respectively. This fault detection mechanism here is beacon ants only. The base delivery ratio remains approximately constant with varying network sizes, indicating that the protocol can be considered scaleable with increading network size. However there is a decrease in delivery ratio for the smallest size network. As seen before in 5.1, the delivery ratio for all sinks in generally lower for a number of possible reasons, however there is not significant decrease.

Figure 5.3 indicates that the fault tolerant adaptation of the protocol does not significantly impact the delivery ratio when there are no node failures compared with the base ACO protocol. Generally, the delivery ratio of the base ACO protocol with no ant memory is similar to the fault tolerant version of the protocol. There is a decrease in effectiveness for the smallest network size for the fault tolerant protocol when considering all sinks. This could be due to the level of false positives of failure detecting being relatively large compared to network size. This could indicate that the periodic of beacon ants could be reduced in smaller networks, for instance sending a beacon any every

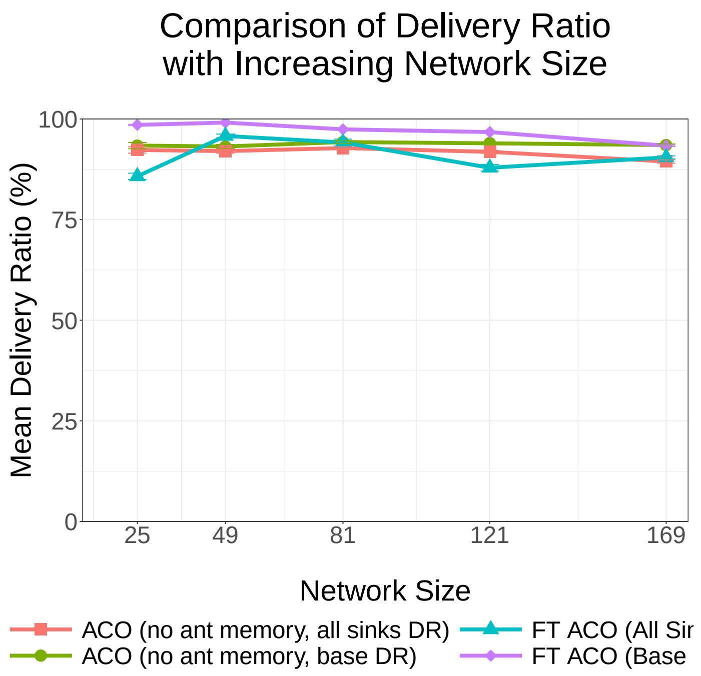


Figure 5.3: Comparing original ACO protocol with Fault Tolerant variant

other cycle.

The average number of nodes involved in sending packets from the sources to the sinks for each cycle is shown in figure 5.4. Looking at the average nodes involved for all cases, including cycles where not all sinks receive messages or where the backbone is not formed, the number of nodes involved with increased node failures decreases slightly. This is due to the fact that some of these cycles will be during fault recovery, when full routes are not formed. For instance, if an ant from one source attempts to travel towards the backbone, but a node has failed on the route towards the backbone, this ant will halt at the failed node. The other ant will continue to follow it's preferred route towards the sinks, though without a backbone, and so the number of nodes involved in this cycle will be less. However, despite fewer nodes being involved in sending messages, this is not an optimal scenario, as the data from both sources will not have been aggregated and sent to both sinks.

The average number of nodes involved when looking at only cycles where both

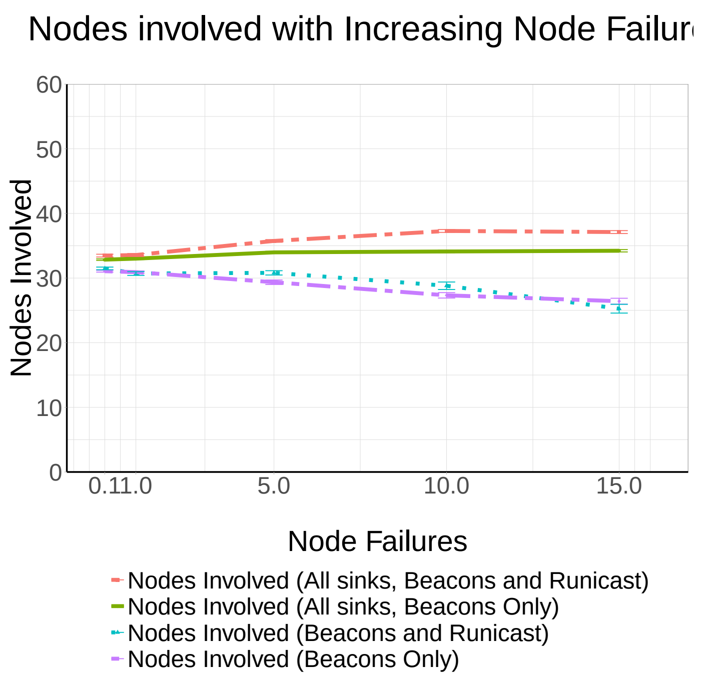


Figure 5.4: Nodes involved in Fault Tolerant ACO Routing Protocol

sinks receive messages from both sources shows a very slight increase in the number of nodes involved. This is expected, as often additional nodes may be required in routes where there has been a failure. For instance, ants may route around nodes that have failed, including additional nodes in the process, or there may be a slight shortening of the backbone, and so the positive effects of aggregation is reduced.

There is a small change in the average number of nodes involved when comparing the beacons only method of fault detection to the beacon ants and runicast method of fault detection. The beacons only method of fault detection uses fewer nodes in routes where all sinks receive from all sources. This could again be explained by there being too much trail evaporated for this fault detection method, leading to worse routes. When considering the average nodes involved for all cycles, the beacon ants and runicast method has more variation between node failure amounts, which adds credence to the theory that this method causes too much pheromone trail to be evaporated.

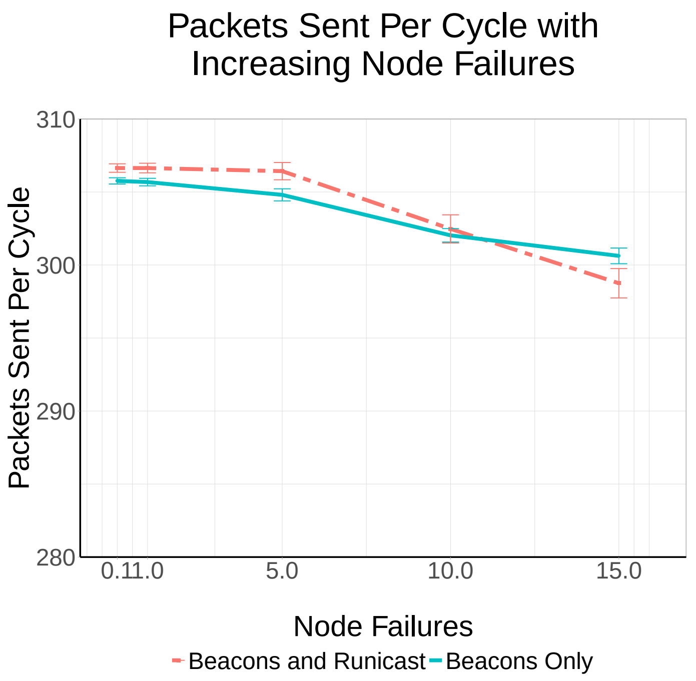
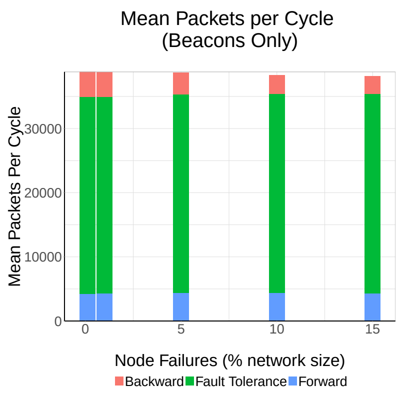


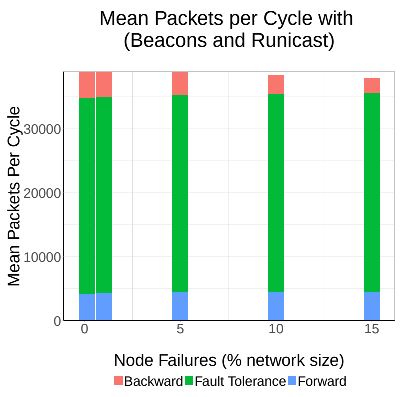
Figure 5.5: Packets sent per cycle in Fault Tolerant ACO Routing Protocol

Figure 5.5 shows the mean packets sent per cycle with increasing node failures. It can be observed that there is a decrease in the number of packets sent per cycle with increased node failures. This is likely due to a reduction in the number of beacon ants being broadcasted as more nodes in the network fail, as these types of message consist of most of the packets sent per cycle, as shown in figures 5.6a and 5.6b. The beacons and runicast method has more packets sent per cycle on average than the beacons only method, until the number of failures is at its largest amount. This trend approximately mirrors the nodes involved trend for the beacons and runicast method of fault detection, indicating that this is at least partially explained by the number of nodes involved in routing. Additionally, the runicast method of routing involved multiple repeat transmissions while trying to send to a failed node, leading to increased packets. This figure could be varied depending on network setup, leading to a variation in the number of packets sent due to retransmissions.

When considering only packets sent on the forward ant route (Figure 5.7),



(a) Packets sent per cycle using the Fault Tolerant ACO Routing Protocol with Beacon Ants



(b) Packets sent per cycle using the Fault Tolerant ACO Routing Protocol with both Beacon ants and Runicast

Figure 5.6: Comparing the mean packets sent per cycle using both fault detection methods

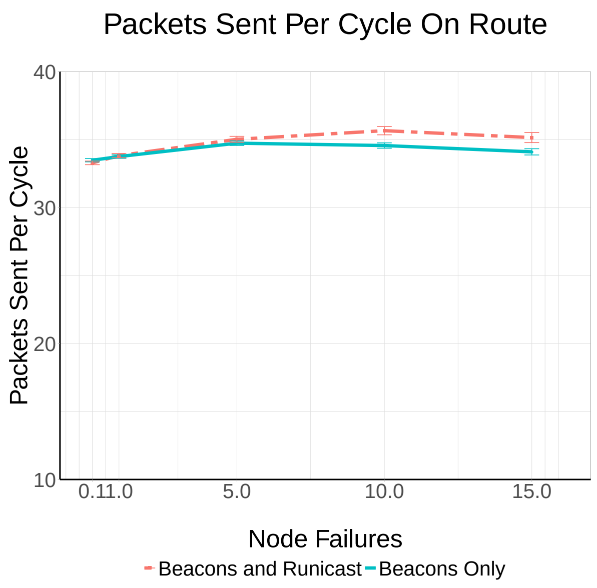


Figure 5.7: Mean Packets Sent On Route in Fault Tolerant ACO Routing Protocol

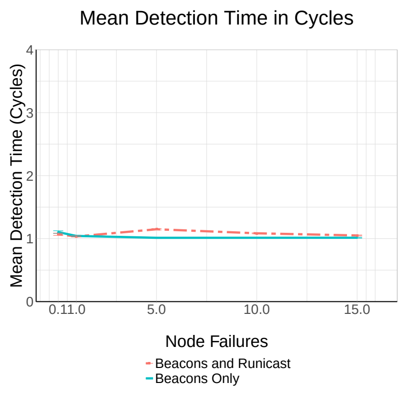
fewer packets are sent as node failures increase for both forms of fault detection. This again can be explained by there being more cycles consisting of fault recovery and incomplete routes when node failures increase. The number of beacons for beacons only is again slightly less, due to better routes.

Figure 5.8a shows that the mean detection time remains largely constant regardless of the number of failures. This is understood to be because the method is localised, and so number of total network failures should not make a significant difference to fault detection.

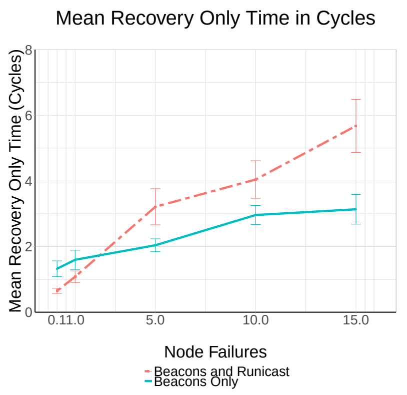
The recovery only times shown in figure 5.8b shows that the time to recover after detection increases as the total node failures in the network increase. This is expected as more node failures leads to more difficulties in finding routes, as fewer possible routes exist.

The average backbone length is shown in figure 5.8c, where a comparison is made showing the difference between backbones in all routes, and with solely routes where all sinks receive a message, as well as between fault detection methods. Generally the backbone length remains constant between all these variations, with most measurements being with error of each other. This indicates that the fault tolerant variations are still able to consistently form a backbone.

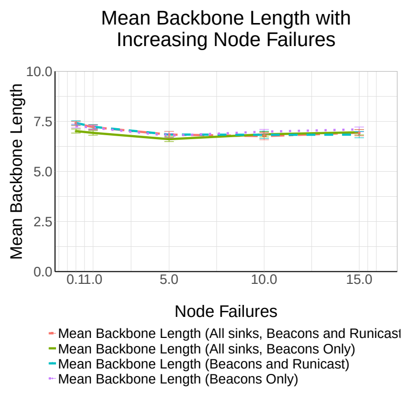
The total time to recover from node failures, including the time to detect the fault, generally increased with an increased number of node failures, as seen in figure 5.8d. This is expected, as as the number of failures increases, the harder it becomes to recover from the failure, due to failures elsewhere in the network. The error on the figures are large, indicating a relatively large amount of variation in the recovery time. This could be due to the fact that a lot of failures will occur for nodes that are not on the route followed by ants, meaning the recovery time is 0 cycles. It can be seen that the beacons and runicast fault detection method has a quicker recovery time for a smaller number of failures, but is slower for the larger number of failures. As we can see in figure 5.8a, this is likely not caused by a difference in detection time, but actually a difference in recovery time, figure 5.8b. It seems likely that the reduction in trail was too much with this method of fault recovery.



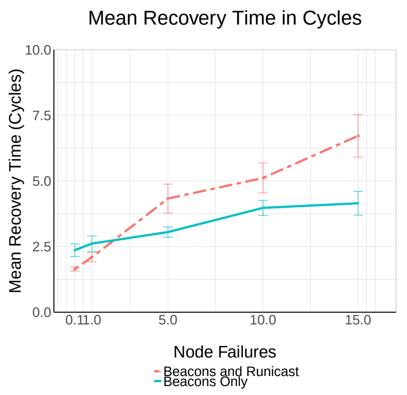
(a) Mean Detection Time in Fault Tolerant ACO Routing Protocol



(b) Mean Recovery Only (no detection) Time in Fault Tolerant ACO Routing Protocol



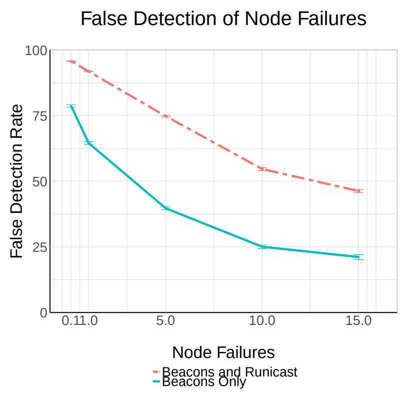
(c) Average Backbone Length in Fault Tolerant ACO Routing Protocol



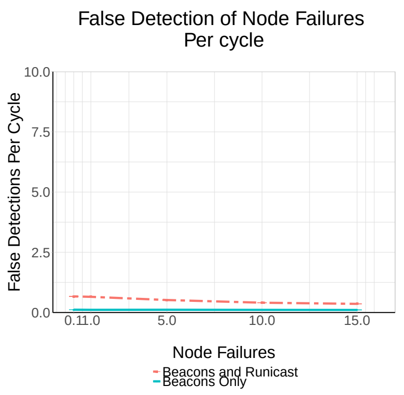
(d) Mean Recovery Time in Fault Tolerant ACO Routing Protocol

Figure 5.8: Recovery Analysis for FT-ACO

In the process of fault detection there will be a number of false positive reports, where a node has been detected as faulty when it is not. This false detection rate is shown in 5.9a for both beacons only fault detection and beacons and runicast fault detection. This is the overall false detection over the course of the experiment as a percentage of the total number of fault detections. The false detection rate is very high for lower number of failures, indicating



(a) False Fault Detection Rate in Fault Tolerant ACO Routing Protocol



(b) False Fault Detections Per Cycle in Fault Tolerant ACO Routing Protocol

Figure 5.9: False Detection of Faults

that the window size for the beacons likely needs to be made bigger, or the rate at which the beacon broadcast was sent should be lower. This explains the high number of packets sent. However, this does indicate that false detections does not have a great deal of impact on the final success of fault recovery, as other performance metrics remain good.

When considering the false fault detection per cycle in figure 5.9b, it can be shown that there are very few actual false fault detections for each cycle of the experiment. This could explain why these false reports have little impact on overall fault recovery. Generally there is less than one false fault detection for every cycle, despite the false positives forming a high proportion of the total detections.

### 5.4.3 Patterned Failures

A number of experiments with patterned failures were performed, with the resulting routes explored in this section.

Figure 5.10 shows a patterned failure occurring on the main backbone of the network, with an example of a resulting recovered route. This shows that that the ants tended towards diverting around the node failures, with a resulting

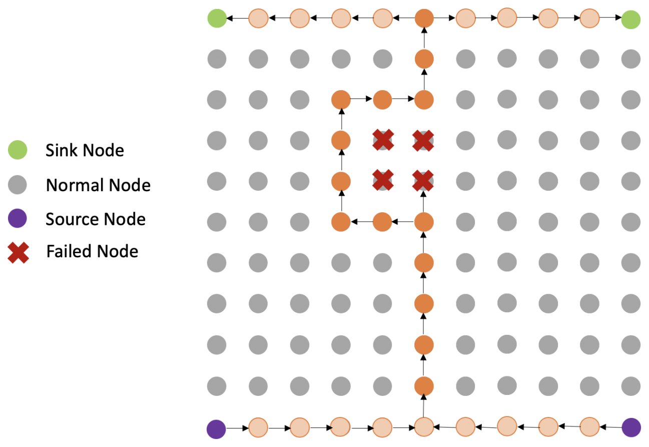


Figure 5.10: Patterned Failure on the backbone and an example route.

longer backbone. With repeated experiments, the average delivery ratio where this failure occurred was 89.3% ± 0.7. This shows that the protocol was generally able to recover reasonably quick from a patterned failure on an important part of the route. The all sinks measure was slightly lower 81% ± 1, likely due to some cycles consisting of fault recovery. The average backbone length was 8.8 ± 0.1 nodes, and the average number of nodes involved 32.5 ± 0.2 nodes. This represents an increase on the random failure results, as expected from visual inspection of the route. The mean detection time is 1 cycle, and the recovery time is 4.25 ± 0.08 cycles. This represents a relatively increase on the random failures, however for a patterned failure on an important part of the route should be considered a reasonably quick recovery.

An example of a recovered route that is formed when a patterned failure occurs after the backbone is split is shown in figure 5.11. For this failure, the mean delivery ratio is 89.5% ± 0.6 and the all sinks delivery ratio 81% ± 1. The base delivery ratio is slightly better than the previous pattern, though the all sinks measure is around the same. This could be due to the sink on the other side still receiving messages as the affected branch recovers from the failures. This pattern saw an average backbone length of 6.3 ± 0.1 nodes, indicating that is led to a smaller backbone in its recovered routes. The average nodes involved was 33.2 ± 0.3 nodes, similar to the previous pattern. The mean detection

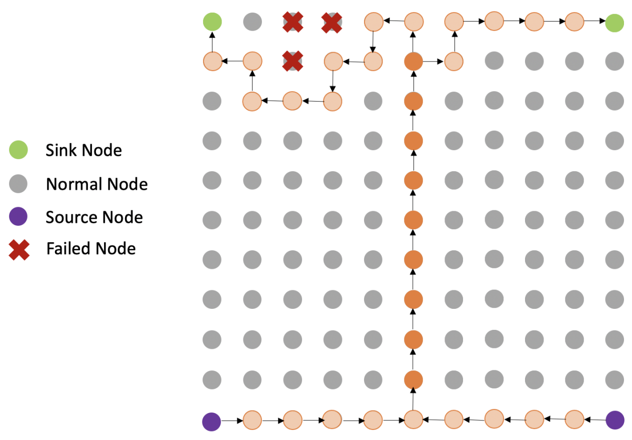


Figure 5.11: Pattern Failure on a ant after the backbone has split.

time was 1 cycle, and recovery time 5.9 ± 0.4.

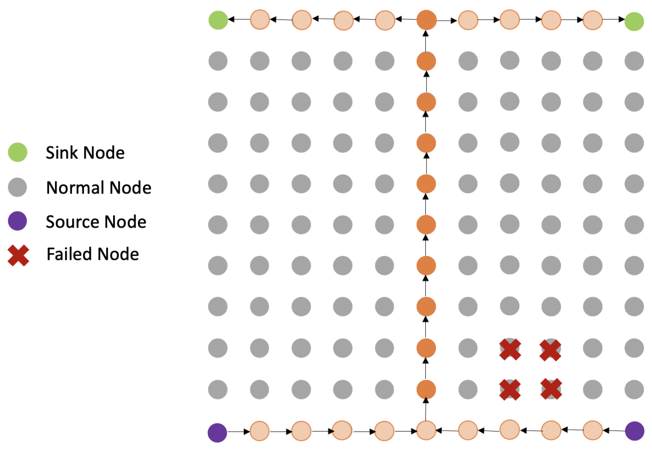


Figure 5.12: Pattern Failure on nodes note usually on a route.

Figure 5.12 shows a patterned failure in an area not usually forming part of a forward ants route, with an example of how forward ants tend to behave. In this scenario, no recovery is necessary, and so delivery ratio does not see a significant drop. The base delivery ratio is 94.0% ± 0.3 and the all sinks ratio is 89.4% ± 0.7, which is similar to other measures with no failures. Generally recovery was not necessary.

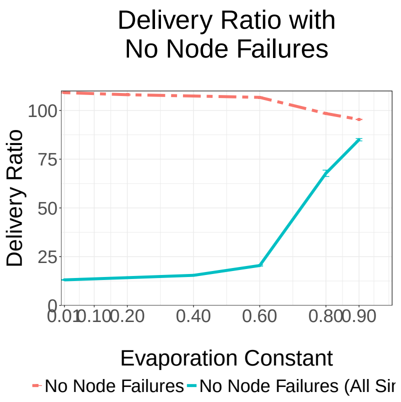
### 5.4.4 Passive Fault Recovery

A concept that has been designated as “Passive Fault Recovery” was investigated, shown in figure 5.13. This relies upon the idea that pheromone trail evaporation over time improves routes and discourages worse routes, and so there should be some amount of recovery possible through the use of normal pheromone evaporation. However in reality performance was not high using this method. Observing the “all sinks” delivery ratio metric, it can be observed from figure 5.13a that delivery ratio suffered with increased evaporation, i.e. lower evaporation constant. This metric followed a similar pattern when there are 1% node failures in the network, though the delivery ratio is less overall (figure 5.13b). The base delivery ratio gives deceptively high values, due a very large amounts of repeat receives on sinks. This scenario occurs when no backbone is formed, but the ant still splits at the appropriate point to receive on both sinks. This can be confirmed by the mismatch between the base and all sinks delivery ratios measured. Figure 5.13c shows the unique delivery ratio, which indicates the delivery ratio when only one ant per sink is counted. This show that the delivery ratio is low for lower evaporation constants, and fault recovery was not very effective.

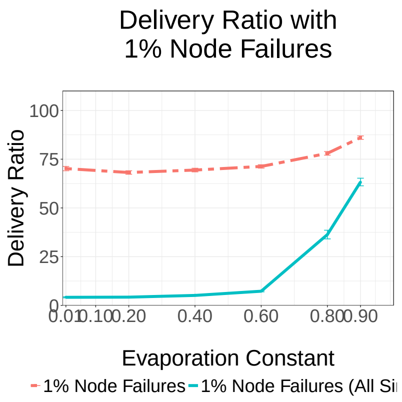
The recovery time of the passive fault recovery scheme is shown in figure 5.13d. The recovery time decreases with higher evaporation constant, before plateauing around 0.6 evaporation constant.

## 5.5 Conclusion

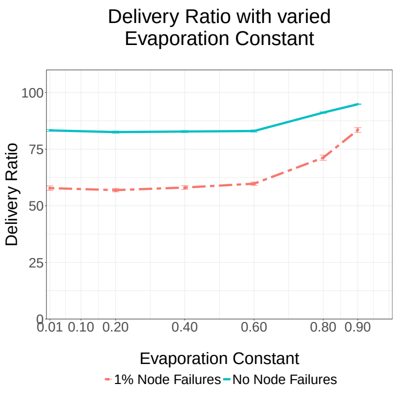
The work presented in this chapter extends upon the routing protocol discussed in the previous chapter in order to create a fault tolerant routing protocol for grid based many-to-many wireless sensor networks. The protocol exploits the localised nature of Ant Colony Optimisation in order to avoid the requirement for network wide updates or refreshes when a failure is detected. This improves the scaleability of the protocol, as each node only requires knowledge about its



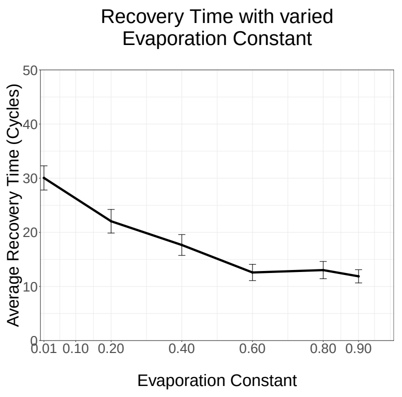
(a) Passive fault tolerance with no node failures



(b) Passive fault tolerance with 1% node failures



(c) Passive Fault Tolerance unique delivery ratio



(d) Passive Fault Tolerance recovery time

Figure 5.13: Summary of passive fault tolerance results

local neighbours, and not the whole network, in order for the routing protocol to be tolerant to failures.

In order to detect and recover from failures, this chapters introduces two novel concepts for ACO based routing. Firstly, the concept of the beacon ant, and ant that is continuously broadcast to all neighbours at a relatively low rate in a heartbeat like fashion, in order to detect if a neighbour has failed. Secondly, the concept of targeted pheromone trail evaporation is introduced in order to recover from failures. If beacon ants indicate that a node may have failed, the neighbours of the node will evaporate the pheromone trail between itself and the node. This evaporation only occurs on that one link, as well as being proportional to the likelihood of failure i.e. number of missed beacon ants. This technique leads to successful fault detection and recovery for high numbers of node failures throughout the network, as well as quick recovery times leading to few delivery cycles failing to deliver messages from sources to sinks.

Chapter 6

# ACO Based Routing in Wireless Sensor Networks for Generating Minimal Steiner Trees

The previous two chapters introduced an ACO based routing protocol for many-to-many wireless sensor networks and a fault tolerant extension of that protocol. This chapter attempts to show that this concept can be applied to further problems in WSN with success. This can show that an ACO based framework is adaptable to a number of scenarios whilst still using the same basic concepts.

The problem of generating minimal Steiner trees in wireless sensor networks is NP-Hard, but if achieved efficiently can have a great deal of useful applications. Solutions for finding minimal Steiner trees is a much investigated problem in circuit design and wired networks, however when attempting to solve the same problem in wireless sensor networks a number of additional problems occur. A distributed solution is required, as nodes may not necessarily have knowledge of network topology, and the topology may change throughout the running of the protocol through node or link failures. Additionally, protocols for wireless sensor networks have a greater need for scalability, with energy costs being a large concern for network lifetime. This means that as the network size increases, the costs involved should not increase disproportionately, for instance dramatically increasing the number of packets set. The protocol should also be scaleable in terms of number of sinks, meaning that the protocol should not need to be repeated for each sink as this uses a disproportionate amount of resources. To solve the problem of finding Steiner Trees in WSN, this chapter presents an ACO based protocol for creating Steiner Trees based on the previously introduced routing protocol.

Wireless Sensor networks will commonly experience problems with faulty nodes or loss of links, requiring a level of fault tolerance in routing protocols. A fault tolerant variant of the ACO protocol for minimal Steiner Trees is also presented that can recover from node failures occurring on the tree that has been formed.

The work in this chapter presents an ACO based protocol that is able to find minimal Steiner trees in grid based wireless sensor networks with multiple sources and multiple sinks. The problem specification for this chapter will form a Steiner Tree with the terminal points being the sources and sinks. The solution proposed makes use of a newly developed type of ant denoted as a "pheromone initialisation ant", which is used to initialise a minimum level of pheromone level between nodes in the network. This allows ants to be directed towards the sinks without any knowledge of their distance to the sinks, or their neighbours distance to the sinks. The contributions made in this chapter are:

- A novel ACO based protocol for generating Steiner Trees in wireless sensor networks through the use of pheromone initialisation ants, and without the use of ant memroy.
- A distributed implementation of the ACO protocol for Steiner Trees.

- A fault tolerant variant of the Steiner Tree protocol.
- Simulations of the distributed protocol on the COOJA network simulator.

## 6.1 Problem Specification

A Steiner tree consists of a number of terminal points with are connected together. Additional nodes may be used to connect the terminal points, known as Steiner points. A minimal Steiner tree attempts to do this with minimal edge weight. For the problem presented here in wireless sensor networks, the sources and sinks in the network are treated as the terminal points that must be connected together with minimal weight, and any other nodes in the network are treated as Steiner points. In this work, all edge weights are taken to be 1, equivalent to hop count. The goal of this protocol is to connect multiple sources $s_1, \ldots, s_k$ to multiple sinks $\Delta_1, \ldots, \Delta_l$ such that data may be continually delivered via a Steiner Tree structure. The Steiner Tree problem is considered to be NP-hard, and to solve it the meta heuristic of Ant Colony Optimisation is used.

## 6.2 Description of Protocol

The ACO protocol developed to form Steiner trees in networks with multiple sources and multiple sinks is described here. Ants are represented as messages being sent throughout the network from sources to sinks. The ants meet and combine on a particular node to form a backbone, before splitting and continuing on to the sinks. In this way, the sources and sinks, i.e. the terminal nodes, are connected in the form of a Steiner tree. Through the process of pheromone evaporation, a near optimal tree should be found. The protocol differs from the previous ACO based routing protocol in that it initiates with a start up phase to initialise trail amounts, before starting to send messages from sources to sinks for continuous data delivery. The protocol depends more on pheromone trail than on hop count. This has the advantage of not having to store the hop counts to all sinks for each neighbour, instead payload information is stored of the last visiting ant in the form of a bloom filter [12]. Additionally, changes are made to how the backbone is created and split, as with more sources and sinks more creation and split points are needed.

In addition to the base Steiner Tree protocol, a fault tolerant extension of the protocol has also been developed. This involves additional beacon ants being used, as well as a method of routing ants around failed nodes using targeted pheromone evaporation.

### 6.2.1 Steiner Trees using ACO

The ACO based protocol for minimal Steiner Trees is described in detail here, followed by an algorithmic description. As with previous chapters, the protocol starts with a start up phase followed by continuous data delivery.

#### Start up Phase

The goal of the start up phase is to initiate the pheromone trails between nodes so as to encourage formation of Steiner Trees. Essentially, the sources and sinks initiate a flooding broadcast, and the locations where the different broadcast floods meet will approximately form the Steiner Tree. The pheromone trails between nodes are updated in order to encourage forward ants to follow these routes, with more pheromone added with shorter, more desirable routes. Throughout the course of main running of the protocol, the original pheromone updates that occur during the start up phase are maintained as the minimum possible pheromone trail between nodes. This forms a difference with the previous two versions of the protocol, as the hop counts of the neighbouring nodes to all sinks are not maintained. This choice was made as with increasing sources and sinks this may not be sustainable for the nodes to keep in memory. Instead, a greater emphasis is placed on pheromone trail, with the start up phase initiating these values. The steps of the phase are as follows:

- A neighbour discovery process takes place, where each node will send a series of broadcasts in order to identify its neighbours. On receiving a broadcast, the node will add the node to a list of neighbours. All pheromone trails are initially set to be some value $\tau_{init}$, in this case 0.1.

- The source nodes initiate a flood of broadcasts to all neighbouring nodes. The broadcast flood contains information of the originating source as well as the hop count from the source.

- On receiving a broadcast, if the node has not received a broadcast before it will note the hop count from the source and continue broadcasting. If the node has received a broadcast before, it will only continue the process if the hop count is lower than the currently recorded hop count. This one hop count value is the one that is stored, and only for the start up phase.

- If the node chooses to continue its broadcasting, it records the node it had received from in *prevnode* and also sets a timer of length $c$. During the course of the timer of length $c$, if multiple broadcasts from originating from different sources are received, the node is marked as "combineable" using the variable *combineable*. This means that this is a node that may combine to form a backbone during the main operation of the protocol. The combineable variable is set to be the total number of hops of broadcasts taken to reach the node from the sources, and is used later to decide how longer to set a combine timer for.

- If no other broadcasts are received in during the timer, the node sends a broadcast to its neighbours, incrementing the hop count.

- Once all nodes have finished broadcasting, backward ants are launched from the combine nodes. The backward ants follow the route of the broadcasts from the combine node to the sources, remembering the total hop count on the combineable node, and updating the trail between the nodes using:

$$\tau = \tau_{init} + A/l_r \tag{6.1}$$

where $\tau$ indicates the new pheromone value, $\tau_{init}$ is the initial trail value, $l_r$ represents the total hop count on the combineable node, and $A$ is a constant chosen through experimentation in order to weight the pheromone value appropriately. This equation was used as it encourages route formation on nodes that will combine together the most ants, whilst also allowing some variation in choices made by ants such that the pheromone values are not too bias in any one direction. This allows improvement of routes as the algorithm runs.

- Backward ants originating from the combineable node also perform checks that it is following the minimal hops path. Each node on receiving a backward ant will compare the received total hop count from the combineable node with what it has current recorded as the lowest hop count. If the received hop count is lower or this is the first backward ant received, the trail is updated and the backward ant continues. Otherwise, the backward ant stops.

- The same process is repeated with the sink nodes initiating broadcast floods, except the equivalent combineable nodes are now marked as "splittable".

- At the end of the splittable flood, all combineable nodes are also marked as splittable.

- If a node has not been marked as combineable, its combineable variable is set to 1. This is so that each node will wait a short amount of time during main operation of the protocol.

### Continuous Data Delivery

The main protocol operation is the normal formation of ant routes from sources to sinks to deliver data. The protocol works in cycles, with each cycle consisting of the forward ants being launched from the sources in order to travel to the sinks, the sinks launching backward ants to travel to the sources, and one step of pheromone trail evaporation. Generally, the goal of the protocol is to direct ants towards the creation of a backbone. Once the backbone has been created, it will likely split multiple times along the backbone on all splittable nodes. The most successful split, i.e. the split that is associated with the longest backbone while still leading all ants to the sinks, will end up being reinforced by backward ants. Subsequent cycles will have fewer splits as the most successful route is reinforced by pheromone evaporation, making it less likely for a split to occur, meaning fewer messages are sent.

The protocol presented here is different to other ACO based protocols as it allows repeat visits to nodes. Here, if a forward ant lands on a node that has been previously visited but by an ant following a worse route, for instance it has more hops, the ant may visit the node again. This is necessary due to the multiple splits that may take place. Additionally nodes no longer need to store the hops counts to all sinks to all neighbours, but instead store a bloom filter consisting of information of which sources it has received ants from. The data delivery steps are as follows:

- A cycle variable is initially set to be 1. Messages received is set to be 0. A timer is set of time $p$ which should cover the time for forward ants to travel from sources to sinks.
- Forward ants are periodically launched from the source nodes. The forward ant carries in its payload several pieces of information required for routing, which are:
  - The single originating source node of this ant
  - The hop count of the route the forward ant has travelled
  - The number of ants that this forward ant consists of. As backbones form, this number increases.
  - A bloom filter [12] of size $n_{bloom}$ that consists of all the originating source nodes that has formed this forward ant. When a backbone is

formed, all originators are added to the bloom filter.

- Each ant chooses the next node to travel to with equation:

$$p_{i,j} = \begin{cases} \dfrac{\tau_{i,j}^{\alpha}}{\sum\limits_{k \in (N(i) - prevnode)} \tau_{i,k}^{\alpha}} & \forall j \neq prevnode \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

  where $p_{i,j}$ is the probability that the ant $a$ will travel from node $i$ to node $j$, $\tau_{i,j}$ is the pheromone value between node $i$ and node $j$. $\alpha$ indicates the weight of pheromone trail. $N(i)$ represents the neighbours of node $i$ and *prevnode* is the previous node. The equation for ants choosing a node to travel to next has changed from previous variations of the protocol as the hop count to each sink is no longer considered, only the pheromone. Generally the ants will prefer a node with more pheromone trail, likely choosing previously successful routes, or routes that were found in the start up phase. The value of $\alpha$ was found through experimentation, and it was necessary for a balance to be made between what is likely to be a successful route, and making different choices in order to improve the algorithm over time.

- On receiving a forward ant, the node will set the *prevnode* to be the node it received from only if it wishes to continue the ant. The node will decide to continue forwarding the ant if any of the following conditions apply:

  - It is the first forward ant received on this node. In this case the ants payload is copied to the receiving node and used to compare future forward ants to in order to decide if the new ant should continue.
  - If the node has been visited before, the node will check if the source node of that previous visit is in the bloom filter of the ant that has just arrived. If it is, the ant will continue if it has a higher number of ants compared with the previous visit.

  - Alternatively, the ant may also continue if it represents some improvement in route. This means that either, the number of ants is higher, or the number of hops is equal to or lower than the previous forward ant **and** the backbone hops is equal to or higher than the previous forward ant. It is also considered an improvement in route if the hops is so much lower than the previous ant, that the total number of hops of the most recently received forward ant is lower than just the backbone hops of the previous ant.

- In addition to one of these conditions, a node will not forward a node if
  - It is a source; sources only launch forward ants once.
  - It is a sink; sinks don't forward ants but wait to launch backward ants.
  - The node is combineable and needs to wait for the combine timer to complete first.
  - The forward ant was received during a combine timer, in this case only one forward ant is launched after the timer completes.
  - It is a node that has previously combined ants to form a backbone.
  - It is a node that formed a route to a backbone.

- In all cases where a node chooses to continue forwarding the received forward ant, it updates its stored payload to reflect that values of the new forward ant. The messages received variable of the node is incremented by 1. The *prevnode* variable is set to be the node the current node received the forward ant from.

- If a node is combineable, the node will set a combine timer of length $c$. During this combine timer, if another forward ant is received, the node will check if it should combine this ant into a backbone if it satisfies either of these conditions:
  - No other ant has been received from the same originator, i.e. the originator is not in the bloom filter.

  - The originator of the received ant is in the nodes bloom filter, but the ant represents an improvement in route.

- If the node chooses to combine this ant into a backbone, the node will update its stored payload to be the amalgamation of the current ant with the new ant. This involves adding the current number of ants to the new ants number of ants (which may be higher than 1, if the received ant was already a backbone, and had previously combined), and adding the total hops together. Also, the bloom filter of the node is updated such that all locations set to 1 in the received ants bloom filter, are also set to 1 for the current node. This means that that nodes will be able to check the originators of all ants that have been combined to form this backbone. All previous nodes will be recorded.

- If a backbone has been formed, backward ants are launched from the creation node. The backward ants follow the path of the forward ants backward, either back to the sources, or back to the most recent creation node before this one. No pheromone trail is updated, only a flag set on the node to mark it as part of a route that formed a backbone. These nodes will not be travelled along by subsequent forward ants, as this would likely form loops from backtracking ants.

- After a backbone has been formed, a node may choose to split instead of sending to a particular node. A node chooses to split if it is splittable, and the pheromone trail coming out of the node is significantly higher than the pheromone trail going into the node. The nodes into the node is usually trail coming in from *prevnode*, unless the node is a combine node, where there will be multiple previous listed. The nodes out are all other neighbouring nodes. This decision to split is made using:

$$\mathrm{w_{in}} \cdot |\mathrm{nodes_{out}}| \cdot \psi < \mathrm{w_{out}} \tag{6.3}$$

where $w_{in}$ is the total amount of pheromone trail of the nodes leading into the node and $w_{out}$ is the total amount of pheromone trail leading out of the node. $|nodes_{out}|$ is the number of nodes leading out, and $\psi$ is referred to as the split factor, and is varied based on how often splits are desired to occur. The equation was chosen as a node with more trail out than in implies a split would be advantageous, and the constant factor $\psi$ found through experimentation in order to ensure a balance between following the preferred existing route and finding new routes.

- If a node chooses to split, a broadcast is sent from the node with the same payload as a normal forward ant.
- Nodes receiving a broadcast will treat it as a forward ant and respond the same.
- When a sink receives a forward ant, the ant stops. If this is the first forward ant on the sink, the sink will store its payload and previous node in *prevnode*. Otherwise, the sink will compare the forward ant with its stored payload from the current best previous forward ant. If this forward ant represents and improvement, the sink will update its stored payload information and *prevnode*.
- After timer p has completed, the nodes stop forwarding forward ants and the sinks each launch a single backward ant to the previous nodes stored in *prevnode*.
- Backward ants travel to the node stored in *prevnode* and update trail change using

$$\Delta\tau_t = \Delta\tau_{t-1} + (\frac{B}{l_r} \cdot (C \cdot l_b)) + (D \cdot |ants|) \tag{6.4}$$

where $\Delta\tau_t$ indicates the change in pheromone value at current time $t$, $l_r$ represents the total forward route length, $l_b$ is the length of the backbone, $|ants|$ is the number of ants that formed the backbone, and $B$, $C$, and $D$ are the weights of these values. The constant values were found through experimentation for the particular circumstance of the network in order to encourage the ants to follow particular kinds of routes.

- If a node receives more than one backward ant, this indicates that it has split and subsequent forward ants have received on multiple sinks. This means that this is a node where splitting should be reinforced, and so trail is updated slightly more on the nodes coming out of the node. The change in trail $\Delta\tau$ is multiplied by a constant referred to as the true end backbone factor, $x$.

- When a backward ant reaches a backbone creation node, the node will broadcast in order to continue backward ants.

- When the backward ant reaches the source, it stops.

- The pheromone value is periodically, using the equation:

$$\tau_t = \begin{cases} \tau_{start} + (\rho \cdot \tau_{t-1}) + \Delta\tau_t & \forall(\tau_{start} + (\rho \cdot \tau_{t-1}) + \Delta\tau_t) > \tau_{min} \\ \tau_{min} & \text{otherwise} \end{cases} \tag{6.5}$$

  where $\tau_t$ represents the pheromone value at time $t$, $\tau_{start}$ is the start up phase trail, $\rho \in 01$ indicates the rate of evaporation, and $\Delta\tau$ is the trail change value. $\tau_{min}$ indicates the minimal trail value between nodes, and equates to the initial trail laid between nodes during the start up phase. This value should remain the minimum amount of trail to prevent evaporation of the base values found in the start up phase, as generally these do not want to be variated from too much. However, this is a value that may not be necessary for all circumstances for the WSN.

- Increment the cycle count by 1, $t$ by 1, and launch one ant from each source node.

### 6.2.2 Fault Tolerant Steiner Trees using ACO

The fault tolerant Steiner Tree protocol using ACO follows the same start up phase and data delivery stage as the base Steiner Tree protocol with additional beacon ants. This beacon ants enable fault detection between nodes. The fault tolerant process is the same as the Fault Tolerant variant of ACO presented in the previous chapter, but without the use of reliable unicast as this was shown to not add much benefit. The addition steps are:

1. In addition to targeted trail evaporation, the node that has detected a faulty neighbour also sets itself as splittable. This allows quicker recovery around a node.

2. Neighbours within a one hop distance of the node that has detected a fault also mark themselves as splittable. This occurs during the first broadcast received from the original node that detected the fault.

### 6.2.3 Additions from base ACO protocol

A summary of the additions and changes made to the ACO protocol for the variant presented for creation of Steiner Trees is explained here.

- The start up phase is changed in order to initialise the pheromone trail between nodes to encourage Steiner Trees. This forms a change from the base protocol which mostly aims to find neighbours and hop counts, here though neighbour discovery still takes place hop counts are not stored, and only used in the start up phase to find pheromone trail amounts.

- Forward ants are allowed to repeatedly visit nodes as long as it represents an improved route, which means checking what has currently visited the node.

- Use of a bloom filter to keep track of what has been visited of each node

- Change in how the backbone is split to be probability based.

- Pheromone updates have been changed again to facilitate a different goal.

- Allowing more splitting of nodes for fault tolerance.

## 6.3 Simulation Setup

The ACO based protocol for generating Steiner Trees was, as with the previous ACO based protocols, implemented for Contiki OS, an operating system developed for Internet of Things based devices [36]. All simulations were carried out on COOJA, a simulator built for Contiki OS [93]. The sensor nodes are emulated sky motes using the UDGM radio medium, CSMA MAC driver, and the null RDC driver for transmission, to keep consistent with previous simulations for the ACO based protocol. The same network setup and settings are used as in previous chapters, in order to enable comparisons.

For each network topology approximately 230 repeat simulations were performed. Each simulation was run for long enough for 127 cycles to complete.

### 6.3.1 Network Configuration

All experiments were performed on a square grid of nodes with varying sizes $n \times n$, where $n \in \{5, 7, 9, 11, 13\}$. The distance between all nodes is constant, and the transmission range and power setup such that each node can only communicate with its horizontal and vertical neighbours. The number of sources and sinks as well as the placement of these sources and sinks are varied to explore the protocols ability to generate Steiner Trees. Forward ants are launched from the sources at the same time for each cycle. The main change from previous chapters is the changes in source and sink numbers and placement.

### 6.3.2 Failure Model

Both random and patterned failures were investigated. For random failures, a maximum number of node failures during the course of the experiment was set, and each node had an equal chance of failing at any point. Patterned failures

| Name | Symbol | Value |
|---|---|---|
| Pheromone Impact (Setup) | A | 7.4 |
| Pheromone Impact (Pre Backbone) | $\alpha_{pre}$ | 8 |
| Pheromone Impact (On and Post Backbone) | $\alpha_{post}$ | 3 |
| Split Factor | $\psi$ | 0.5 |
| Bloom Filter Size | $n_{bloom}$ | 18 |
| Route Length Impact | B | 2 |
| Backbone Length Impact | C | 2 |
| Num. Ants Impact | D | 4 |
| Evaporation Constant | $\rho$ | 0.99 |

Table 6.2: Simulation parameters for the ACO based protocols for Steiner Trees in Many-to-Many WSN

were predetermined ahead of time in order to investigate a particular scenarios. On failure, no nodes are able to communicate with the failed node.

### 6.3.3 Parameters

The selection of input parameters is an important step in the running of the ACO based protocol. Much of the adaptation of the protocol to differing scenarios is through the changing of input parameters. For instance, networks where it is more important to form a long backbone will require a longer Backbone Length Impact variable. This is true in the case of Steiner Trees, however it is also very important to include as many ants as possible along the backbone, and so the Num. Ants Impact is high. The input parameters are listed in table 6.2.

## 6.4 Results

In order to determine success of the protocol, a number of performance metrics are investigated. In addition to these metrics, a graphical representation of routes will also be presented, in order to examine how well Steiner Trees have formed. The main performance metrics are listed below:

- **Delivery Ratio**: The same as described in previous chapters, though the ratio will need to be changed with varied number of sources and sinks.

For instance, if there are 3 sinks then more messages will be expected, so the ratio calculation is adjusted accordingly.

- **Number of Nodes involved per cycle**: Measued in the same way as in the previous two chapters.
- **Number of Nodes involved on route per cycle**: This metric looks at how many nodes form the final most successful path from sources to sinks in each cycle. As the protocol runs, this figure should stay relatively constant. This metric can be thought of as the Steiner tree weight.
  - Base: Total nodes involved on the final route in each cycle, for routes where the backbone formed and where they did not.
  - All Sinks Nodes involved: A measure of how many nodes on the final are involved only for the routes where all sinks receive from all sources, i.e. a backbone has been formed of ants from all sources.
- **Packets sent per cycle**: Measured in the same way as previous chapters
- **Backbone Length**: The number of nodes that form the backbone. Backbone can be measured in multiple ways when the number of sources and sinks is varied, as there may be multiple combination nodes and splitting nodes on a single backbone. For instance, a network with three sources will likely have two nodes where ants merge to form a backbone, which means there a two possible nodes which to count the backbone length. For this reason multiple possible backbone lengths are investigated.
  - Maximum backbone length: the total number of nodes from the first split to the last split.
  - Minimum backbone length: the total number of nodes from the last create on the backbone to the first split.
- **Fault Detection time** and **Fault Recovery time**, measured in the same way as previous chapters

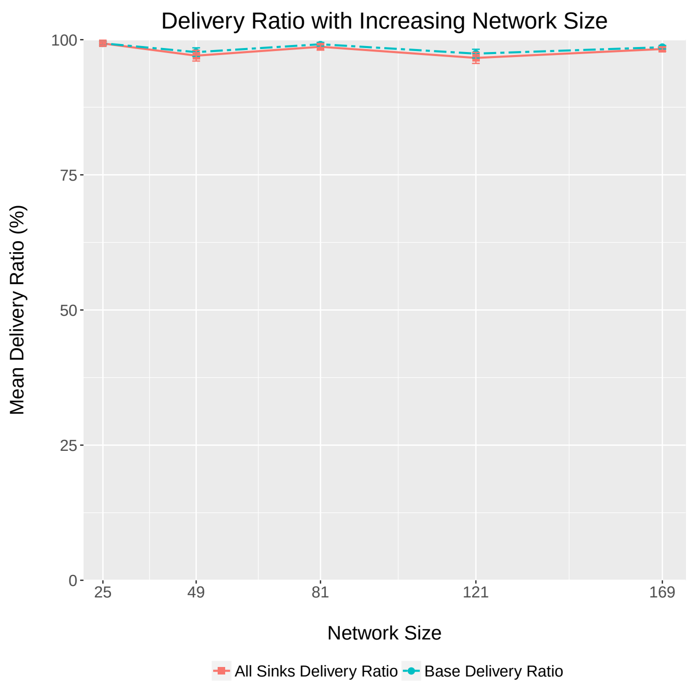


Figure 6.1: Delivery ratio of network of size 121 nodes with 2 sources and 2 sinks.

Initially the ACO protocol for Steiner Trees was tested on a simple topology with two sources and two sinks in the outer corners of the network, as was investigated for the previous chapters. This enables comparison between the two protocols, and will also provide a base representation of performance for the protocol. The results for this simple network was carried out on multiple network sizes. The delivery ratio as the network size increases is shown in figure 6.1, and it can be shown that the delivery ratio remains high for a range of network sizes. There is little difference between the base delivery ratio and the all sinks delivery ratio, indicating that most cycles a backbone successfully formed and all sinks received a forward ant. Figure 6.2 shows a comparison between the ACO protocol for Steiner Trees with the previous versions of the ACO code for wireless sensor networks. The Steiner Tree protocol consistently has a higher delivery ratio compared with the other version of the protocols, and also has the advantage of not appearing to reduce in performance at the largest network size.

The nodes involved performance metric is shown in figure 6.3a. The total

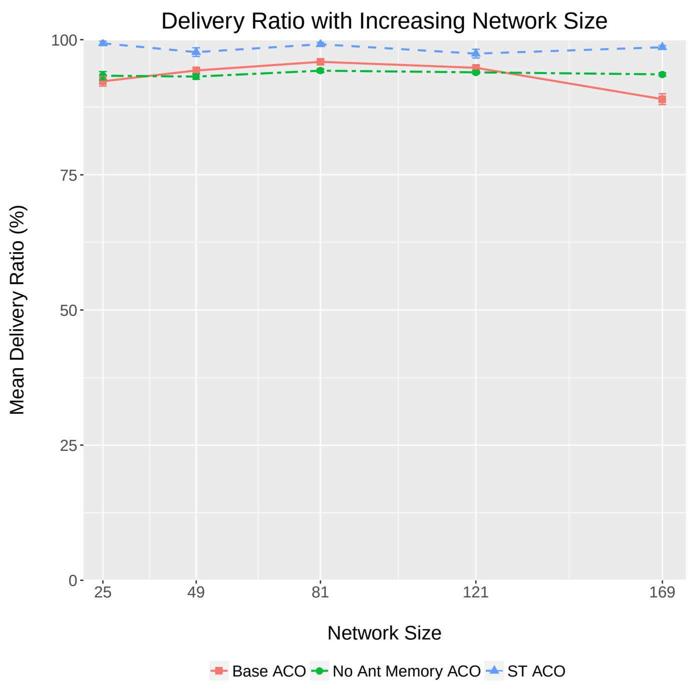
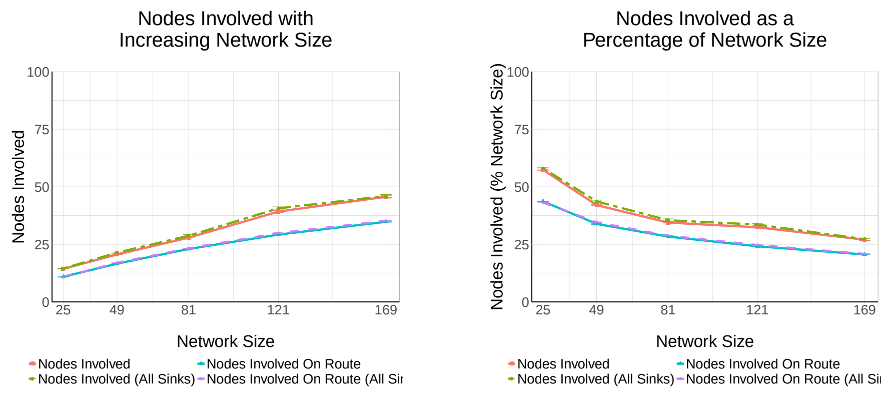


Figure 6.2: Comparison of delivery ratio of network of size 121 nodes with 2 sources and 2 sinks with other versions of the ACO protocol.

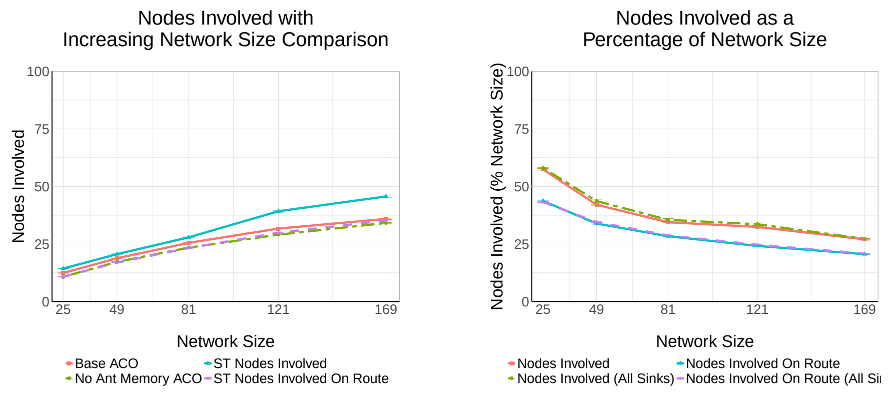
number of nodes involved gets larger with increasing network size, as expected due to the minimum distance between sources and sinks being higher when the network is bigger. The difference between the nodes involved base metric and the nodes involved all sinks metric is minimal, indicating that there were not many cycles where not all sinks received a forward ant. This is consistent with the delivery ratio figure. The nodes involved on route metric is lower than the total nodes involved, as expected. This metric does not include nodes that sent messages during the cycle but didn't end up on the final best route, a scenario that is common at the start of each simulation while the best route is being converged upon. Again there is not much difference between the base form of the metric and the all sinks form of the metric, indicating that most cycles received on all sinks.

Generally, the number of nodes involved, both total and on route, does not increase linearly with network size, showing that the protocol is scaleable with total network size. This is reinforced by the results shown in figure 6.3b, which



(a) Mean nodes involved in routing per cycle

(b) Mean nodes involved per cycle as a percentage of network size



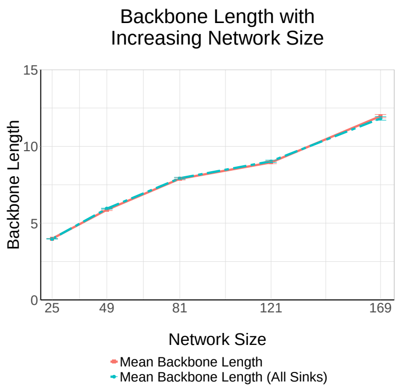
(c) Nodes Involved in routing comparison

(d) Nodes involved in routing as a percentage of network size

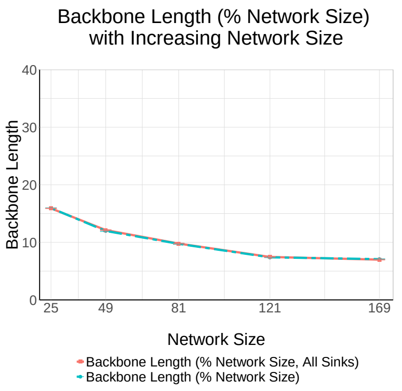
Figure 6.3: Nodes Involved in Steiner Tree Routing

shows the nodes involved as a percentage of network size. The nodes involved as a percentage of network size decreases as network size increases. This implies that for larger network sizes, proportionally fewer nodes are required in routing from all sources to all sinks, which means that fewer messages are sent leading to a longer network lifetime. A similar trend is seen with the nodes involved on route, with fewer nodes involved on route as the network size increases. For both total nodes involved and the nodes involved on route, there is again very little difference between the base metric and the all sinks version of the metric, as expected as this occurs in figure 6.3a. Also similarly to what is seen in figure 6.3a, the nodes involved on the route is lower than the total nodes involved.

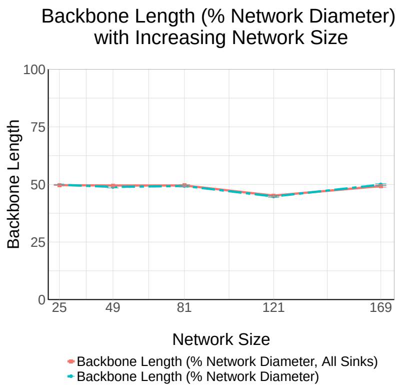
Figure 6.3c shows a comparison between the nodes involved performance metric



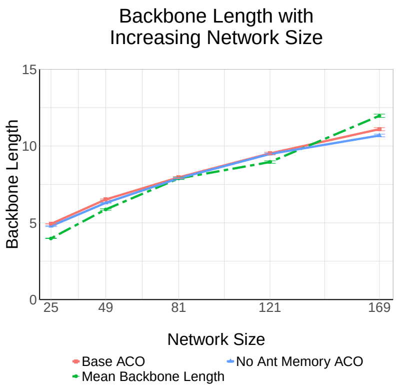
(a) Mean Backbone Length



(b) Mean backbone length as percentage of network size



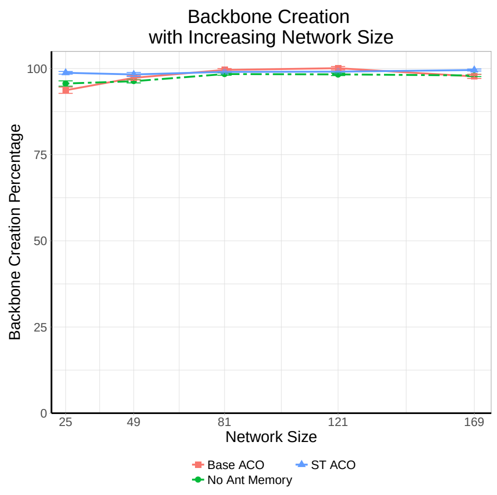
(c) Backbone length as a percentage of network diameter



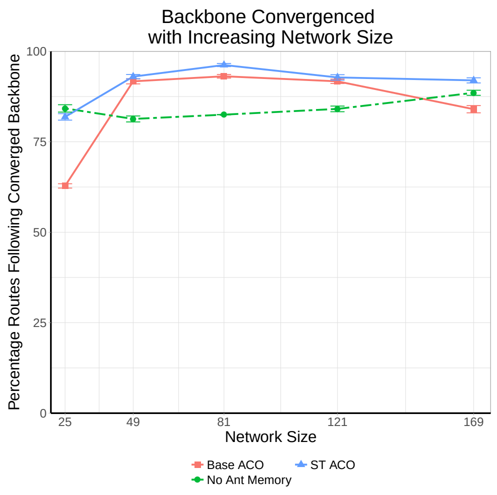
(d) Backbone length comparison

Figure 6.4: Backbone Analysis in Steiner Tree Routing

with previous versions of the ACO protocol. The Steiner Tree ACO protocol has more total nodes involved than other versions of the protocol with a larger difference for larger network sizes. The nodes involved on route metric is largely similar to the other versions of the protocol. This indicates that the Steiner Trees version of the protocol may not be quite as scaleable as a cost for the ability to form Steiner Trees. It is worth noting that the nodes involved figure may reduced by varying the split factor, however this may be at the cost of building less successful trees. Figure 6.3d compares the nodes involved as a percentage of network size, and it can be observed that both versions of the protocol see a decrease in the percentage of nodes involved as network size increases, however the base ACO protocol generally has fewer nodes involved in total. This is consistent with previous figures, and reinforces the need to strike a balance between splitting of the backbone and forming efficient Steiner trees.



(a) Percentage of cycles where a backbone was created



(b) Percentage of cycles where the converged backbone was being followed by ants.

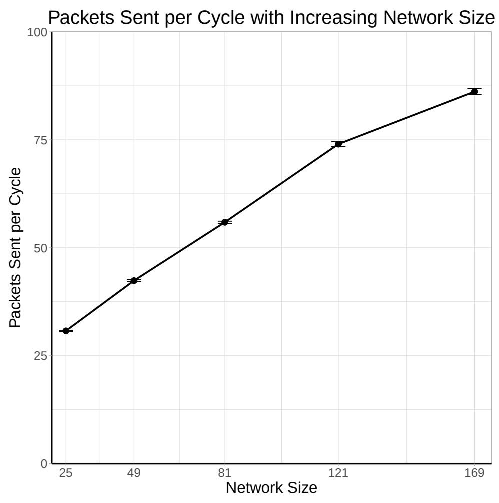
Figure 6.5: Analysis of Backbone Convergence in Steiner Tree Routing

The backbone length that formed in the network with two sources and two sinks is shown in figure 6.4a. As there are only two sources and two sinks, there will only be one backbone creation node and one backbone split node, which means only one measure of backbone length is possible, and this is shown here. The backbone length increases as network size increases, as the minimum distance that it is possible to travel to get to the sinks increases at the network gets bigger. There is virtually no difference between the base metric and the all sinks metric, indicating that when a backbone forms then it is very likely that both sinks will receive a message from it. The backbone length as a percentage of network size is represented in figure 6.4b. The backbone gets proportionally shorter with increased network size, however not linearly. This could indicate that the backbone becomes less effective as a data aggregator as the network size increases, however when the backbone length as a percentage of maximum network diameter is investigated (figure 6.4c), it can be shown that backbone length remains constant. It can be argued that the positive aggregation effects of the backbone is just as effective at larger network sizes, as the backbone length remains a similar percentage of the network diameter.

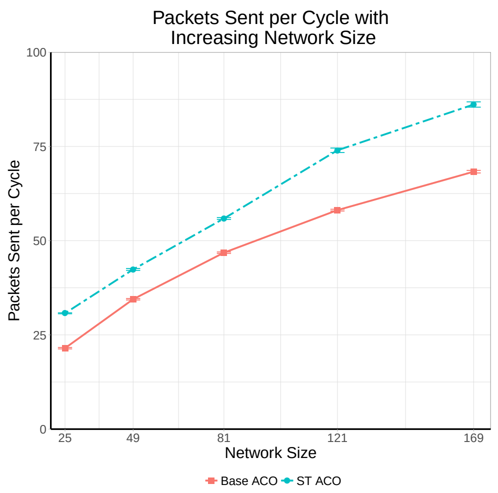
Figure 6.4d compares the backbone length formed by the Steiner Trees ACO protocol with the other versions of the protocol. The Steiner trees backbone length follows a similar trend to the other versions of the protocol, with backbone length increasing with network size. The smallest network sizes see a slightly smaller mean backbone length, possibly indicating that the backbone formation is not as successful for smaller network sizes with the Steiner Tree protocol. The largest network size sees an increase in backbone length over the other version of the protocols, indicating that this version of the protocol has an advantage in larger network sizes when it comes to backbone formation.

The percentage of cycles where a backbone is created over the course of the simulation is shown in figure 6.5a. A backbone is consistently created for most cycles with very little variation across network sizes. The Steiner Tree version of the protocol is arguably better at forming backbones than other versions of the protocol, as there is no drop at the smallest network size, and it is unlikely for multiple backbones to be formed during the same cycle when there are only two sources.

Backbone Convergence is investigated in figure 6.5b. This figure shows the percentage of cycles that follow the 'converged' backbone. This is defined as the backbone that is followed by the majority of the routes over the course of the experiment, and more routes following this same route shows better route convergence. The Steiner Tree ACO protocol has lower route convergence for the smallest network size of 25 nodes, but other sizes shows consistent convergence on a route. This drop for the smallest network size could indicate that parameters should be amended for smaller networks, or perhaps timers set for backbone creation may not be optimised for this network size. A similar trend can be seen for the base ACO protocol, except there is also a small drop with



(a) Packets sent per cycle



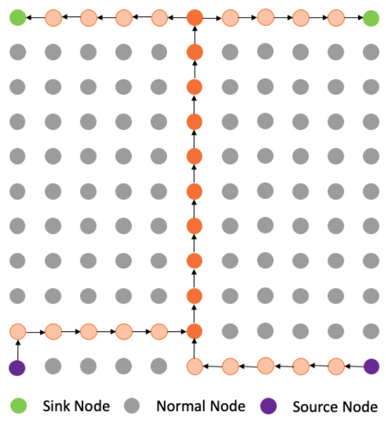
(b) Packets sent per cycle compared with other variation of ACO protocol

Figure 6.6: Packets Sent in Steiner Tree Routing

the largest network size. This could show that the Steiner Tree ACO protocol is more scaleable and performs better for larger networks. Compared to the no ant memory ACO, the Steiner Tree ACO protocol has better backbone convergence for most network sizes, and similar convergence for the smallest network size.

Mean packets sent per cycle (figure 6.6a) increases as network size increases, which is expected as the number of nodes involved in routing also increasing. Comparing the Steiner Trees ACO protocol with the base ACO protocol (figure 6.6b) the packets sent is higher, consistent with the number of nodes involved. This emphasises the necessity of balancing the number of packets sent with the ability to form Steiner Trees connecting multiple sources and sinks.

The network diagram in figure 6.7a shows the most common converged route followed by forward ants in a network of size 121 nodes. It can be seen that the forward ants travelling from the sources travel in a way such that they meet on a node to form a backbone. Following this, in early cycles the nodes will likely split all the way along the backbone until it gets to a node that is no longer splittable, at which point the ants travel towards the sinks. As the



(a) Most common route found for network of size 121 with 2 sources and 2 sinks in the outermost corners of the networks.

| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 98.6 ±0.3 | 98.3 ±0.4 |
| Nodes Involved | 45.7 ±0.6 | 46.0 ±0.6 |
| Nodes Involved on Route | 34.8 ±0.2 | 35.2 ±0.2 |
| Backbone Length | 12.0 ±0.1 | 11.8 ±0.1 |
| Backbone Len. (% Size) | 9.89 ±0.09 | 9.8 ±0.1 |
| Backbone Len. (% Diameter) | 59.9 ±0.5 | 59.1 ±0.6 |
| Packets per Cycle | 74.0 ±0.6 | - |

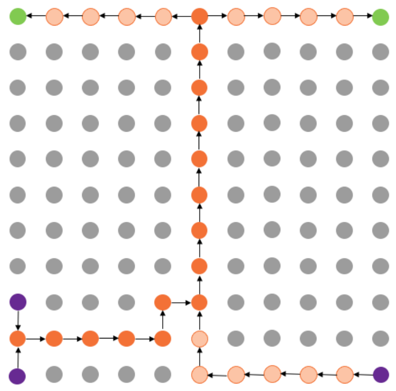
(b) Performance metric for 2 sources, 2 sinks

Figure 6.7: Network of 121 nodes with 2 sources and 2 sinks

simulation runs for longer, the routes tend to converge on a low number of splits.

The most common converged route for a network topology with three sources is shown in figure 6.8a. Two ants are launched from the sources in the bottom left corner, and combine on the node between them. This combined ant then travels to meet the third ant from the bottom right corner, forming a backbone of ants from all three sources. This then travels along the backbone before splitting to reach both sinks. The delivery ratio remains high, implying that the protocol still enables the successful delivery of data from all sources to all sinks. The delivery ratio remains high with the additional source, even when considering the all sinks delivery ratio measure.

The route shown in figure 6.9a is the most common converged route with four sources. The route seems efficient, with ants meeting early to form a



(a) Most common route found for network of size 121 with 3 sources and 2 sinks in the outermost corners of the networks.

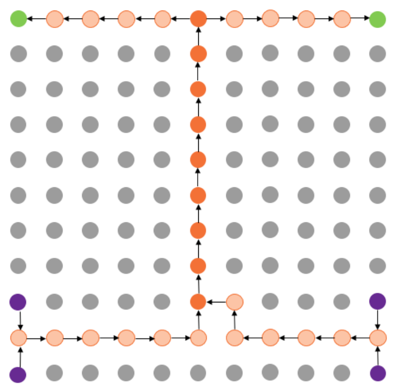
| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 95.1 ±0.9 | 90 ±2 |
| Nodes Involved | 59.0 ±0.7 | 61 ±1 |
| Nodes Involved on Route | 31.5 ±0.3 | 31.3 ±0.5 |
| Backbone Length | 10.7 ±0.1 | 8.6 ±0.2 |
| Backbone Len. (% Size) | 8.9 ±0.1 | 7.1 ±0.2 |
| Backbone Len. (% Diameter) | 53.7 ±0.7 | 43 ±1 |
| Packets per Cycle | 98 ±1 | - |

(b) Performance metric for 3 sources, 2 sinks

Figure 6.8: Network of 121 nodes with 3 sources and 2 sinks

backbone of two ants. The delivery ratios remain consistently high, indicating the ability of the protocol to consistently deliver data. A network with 5 sources is shown in figure 6.10a and still shows consistent delivery ratio, however the all sinks delivery ratio has dropped. This shows that this is the point where the consistent backbone formation is less.

For the highest number of sources tested, six sources, the converged routes formed were less consistent between simulations. This means that there was not one route that could be considered the converged route over the course of all simulations. Additionally, a larger proportion of experiments had no converged route consisting of ants from all 6 sources merging to form a backbone. It was found that 30.9% of simulations successfully formed a backbone consisting of ants from all 6 sources and sent messages to both sinks from this backbone. This suggests that this is the point where the technique starts to become less successful of forming routes. However, there were still a number of experiments



(a) Most common route found for network of size 121 with 4 sources and 2 sinks in the outermost corners of the networks.

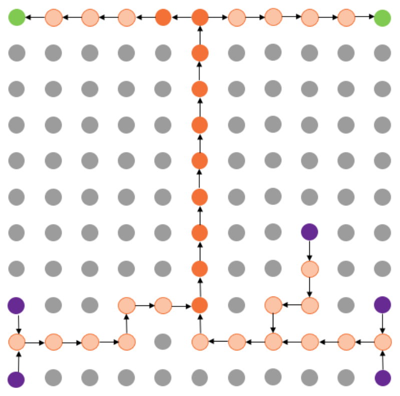
| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 96.3 ±0.8 | 81 ±2 |
| Nodes Involved | 45.7 ±0.7 | 43 ±1 |
| Nodes Involved on Route | 31.0 ±0.3 | 28.3 ±0.6 |
| Backbone Length | 9.0 ±0.2 | 7.8 ±0.2 |
| Backbone Len. (% Size) | 7.4 ±0.2 | 6.5 ±0.2 |
| Backbone Len. (% Diameter) | 45 ±1 | 39 ±1 |
| Packets per Cycle | 82.3 ±0.9 | - |

(b) Performance metric for 4 sources, 2 sinks

Figure 6.9: Network of 121 nodes with 4 sources and 2 sinks

were a Steiner Tree was formed corrected. An example of a successful route, that is all sinks consistently received messages from all sources, is shown in figure 6.11a. The route shows a reasonably efficient route, with most ants joining the backbone in few nodes, albeit not optimally in all cases. The reason for this loss in efficiency in joining the backbone is that the ants tended to want to join together symmetrically; routes to the backbone tended to form in pairs of nodes, at a point equidistant to the two nodes. This is due to issues with synchronising the wait timer on each node, leading to a preference for symmetrical routes. This choice meant that routes prioritised the most number of ants possible to form a backbone, however input parameters could be changed for example to encourage shorter paths to the backbone but fewer ants.

In addition to varying the number of sources in the network, the number of sinks were also varied. The most common converged route for a network with three sinks is shown in figure 6.12a. The backbone forms using the same



(a) Most common route found for network of size 121 with 5 sources and 2 sinks in the outermost corners of the networks.

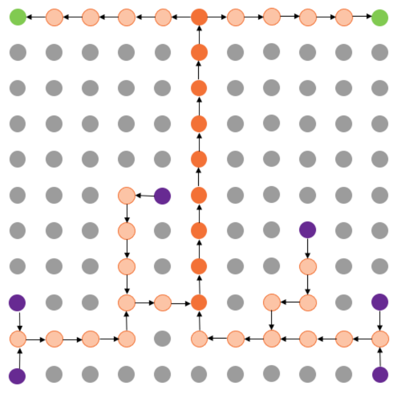
| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 95 ±1 | 59 ±3 |
| Nodes Involved | 68.9 ±0.8 | 61 ±3 |
| Nodes Involved on Route | 33.7 ±0.4 | 28 ±1 |
| Backbone Length | 11.2 ±0.2 | 8.4 ±0.3 |
| Backbone Len. (% Size) | 9.2 ±0.1 | 7.0 ±0.3 |
| Backbone Len. (% Diameter) | 56.0 ±0.9 | 42 ±2 |
| Packets per Cycle | 110 ±1 | - |

(b) Performance metric for 5 sources, 2 sinks

Figure 6.10: Network of 121 nodes with 5 sources and 2 sinks

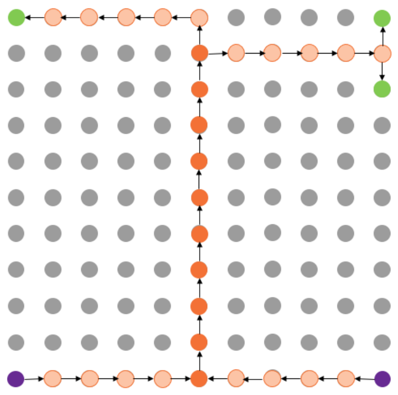
route that previous experiments with 2 sources uses, in a minimal number of hops. The ant then splits at the latest possible point, and delivers messages to all sinks in a minimal number of hops

The most commonly followed converged route for 4 sinks between simulations is shown in figure 6.13a. The delivery ratio is good, with the mean value being 92%, indicating that generally messages were successfully delivered from all sources to all sinks. It can be observed that the route is not optimal after the initial split, as two parallel paths travel to two sinks, when it would be more efficient to have one path that later splits again. This is likely due to the hop count of the lower parallel route being smaller than the upper route, as the lower route has travelled one node less along the shared backbone. The means that the lower sink prefers the route from the lower parallel route. Each ant is not aware about the “true” end of the backbone, as it will not know which of the split ants actually get to the sinks.



(a) Example of a route found for network of size 121 with 6 sources and 2 sinks in the outermost corners of the networks.

At 5 sinks, the protocol starts to be less consistent with forming routes to all sinks, with 41.5% of simulations successfully delivering messages from all sources to all sinks. An example of a successful route is shown in figure 6.14a. This experiment had a delivery ratio of 99.2%, showing that when a route does initially form the protocol is successful at delivery messages to the sinks. The nodes involved in the route for this particular experiment is on average 38.6 and the average backbone length forms a reasonable portion of this, depending on the definition of the backbone. Similarly, with 6 sinks the protocol also did not consistently create successful routes, however an example where the tree did form is shown in 6.14b.

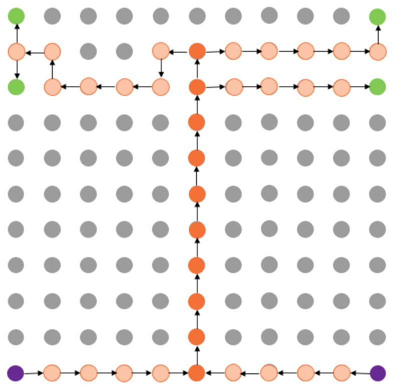


(a) Most common converged route for network of size 121 with 2 sources and 3 sinks in the outermost corners of the networks.

| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 94.9 ±0.7 | 87 ±2 |
| Nodes Involved | 51.9 ±0.8 | 65 ±3 |
| Nodes Involved on Route | 34.4 ±0.3 | 35.1 ±0.4 |
| Backbone Length | 15.0 ±0.1 | 8.28 ±0.1 |
| Backbone Len. (% Size) | 12.4 ±0.1 | 6.8 ±0.1 |
| Backbone Len. (% Diameter) | 74.8 ±0.7 | 41.4 ±0.6 |
| Packets per Cycle | 96 ±1 | - |

(b) Performance metric for 2 sources, 3 sinks

Figure 6.12: Network of 121 nodes with 2 sources and 3 sinks

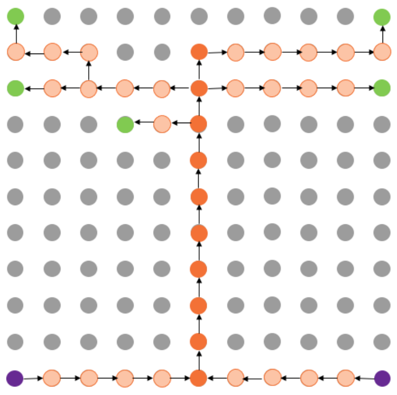


(a) Most common converged route for network of size 121 with 2 sources and 4 sinks in the outermost corners of the networks.

| Performance Metric | Base Value | All Sinks Value |
|---|---|---|
| Delivery Ratio | 92 ±1 | 85 ±2 |
| Nodes Involved | 51.1 ±0.7 | 49.8 ±0.3 |
| Nodes Involved on Route | 36.5 ±0.3 | 33 ±2 |
| Backbone Length | 13.0 ±0.2 | 12.2 ±0.3 |
| Backbone Len. (% Size) | 10.8 ±0.1 | 10.0 ±0.3 |
| Backbone Len. (% Diameter) | 65 ±1 | 61 ±2 |
| Packets per Cycle | 101 ±1 | - |

(b) Performance metric for 2 sources, 4 sinks

Figure 6.13: Network of 121 nodes with 2 sources and 4 sinks


(a) Example of a route found for network of size 121 with 2 sources and 5 sinks.

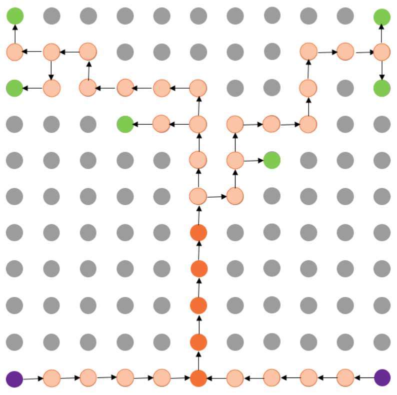
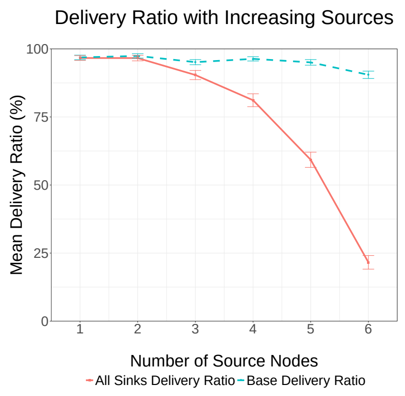
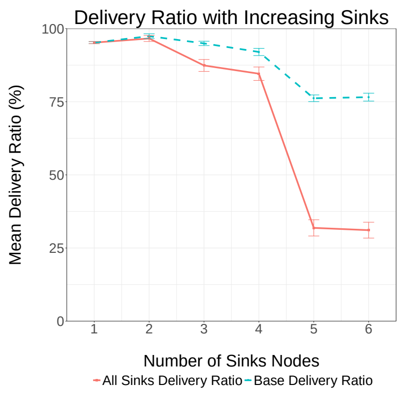

(b) Example of a route found for network of size 121 with 2 sources and 6 sinks.

Figure 6.14: Routes for 5 and 6 sinks

An overview of how the delivery ratio varies with increased sources or sinks is shown in figures 6.15a and 6.15b. It can be seen with increasing number of sources, the all sinks version of delivery ratio decreases with increasing number of sources, indicating that larger number of sources makes it more difficult for successful delivery of packets. However, the base delivery ratio remains high, indicating that at least some ants have delivered data to some sinks. This may be useful in scenarios where multiple sinks are used for fault tolerance. A similar pattern is shown for increasing number of sinks.



(a) Delivery Ratio of networks with varying number of sources



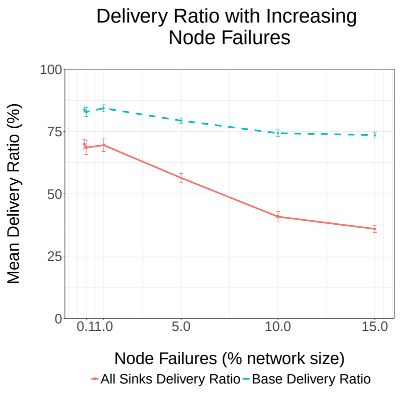
(b) Delivery Ratio of networks with varying number of sinks

Figure 6.15: A comparison of delivery ratios when varying sources and sinks

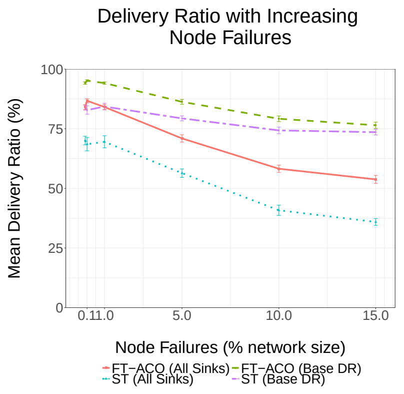
### 6.4.1 Fault Tolerant Steiner Trees using ACO

An extended version of the protocol was implemented that attempts to develop fault tolerant Steiner Trees in wireless sensor networks with multiple sources and multiple sinks. The delivery ratio of this protocol with increasing node failures is shown in figure 6.16a. First random node failures are investigated, then a selected number of patterned failures.

It can be seen in figure 6.16a that the base delivery ratio of the protocol remains relatively consistent as random node failures increases. This means that regardless of failures, the protocol is able to deliver messages from a source to a sink, though not necessarily all of them. The all sinks delivery ratio figure drops as more failures occur, seeing its first significant drop at 5% node failures, or approximately 6 failures throughout the network. When comparing with the fault tolerant ACO based protocol in figure 6.16b, the Steiner Tree protocol follows a similar trend. Both versions of the protocol have a consistent base delivery ratio with increasing node failures, however the All Sinks delivery ratio measure decreases with more node failures.



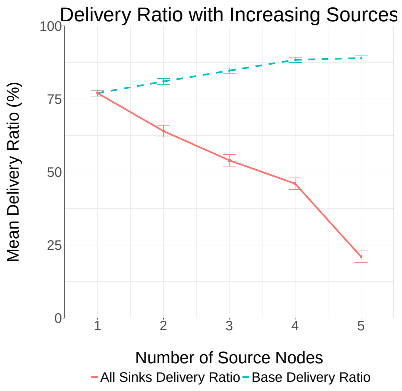
(a) Delivery Ratio of protocol with increasing node failures in a network with 2 sources and 2 sinks.



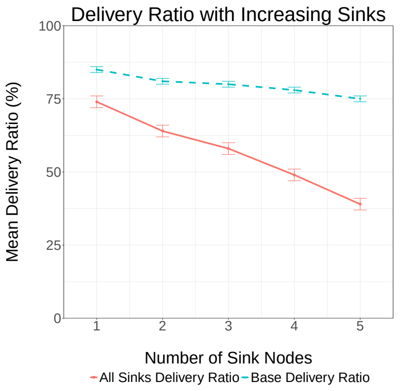
(b) Delivery Ratio of protocol with increasing node failures compared with previous fault tolerant ACO based protocols in a network with 2 sources and 2 sinks

Figure 6.16: Delivery Ratio for Fault Tolerant Steiner Trees

In order to investigate how the protocol performs with a similar number of node failures but with varying topology, figures 6.17a and 6.17b show delivery ratio with increasing sources and sinks when node failures remains at 3%. As the number of sources increases, the base delivery ratio remains high, and tends to increase with more sources. This could be explained by the fact that this figure includes delivery of data on sinks when the full backbone has not been formed, and so it may be easier for some closer sources to reach the sink without forming a full backbone. It could be possible that two main trees are created, and the all sinks delivery ratio is not capturing this behaviour. As with previous results, the all sinks delivery ratio decreases with the number of sources.



(a) Delivery Ratio of networks with 3% node failures with increasing sources.



(b) Delivery Ratio of networks with 3% node failures with increasing sinks.

Figure 6.17: Delivery Ratio for 3% node failures

### Patterned Node Failures

A series of patterned failures were also tested with the fault tolerant Steiner Tree protocol, all with two sources and two sinks. Firstly, a patterned failure where nodes failed on the backbone. This was inconsistent between experiments, however an example of a recovered route is shown in figure 6.18. Likely due to the process of multiple splits, the ants have preferred a route with a short backbone, and were less successful at routing around a failure to the fault tolerant protocol in chapter two. The mean delivery ratio for all experiment for this pattern is 75% $\pm$ 1 and the all sinks delivery ratio is 59% $\pm$ 2. Again this is less successful than the previous fault tolerant protocol.

A pattern where node failures occur between the backbone split node and a sink is shown in figure 6.19. This example route is more efficient, using fewer nodes and having a longer backbone. The protocol performed different routes on different experiments, however the mean delivery ratio was 77% $\pm$ 1 and the all sinks delivery ratio 58% $\pm$ 2. This shows that the Steiner Tree ACO protocol was less effective than the base fault tolerant ACO protocol at delivering to all sinks in the case of failure.

The final patterned failure investigated a group of nodes failing that do not

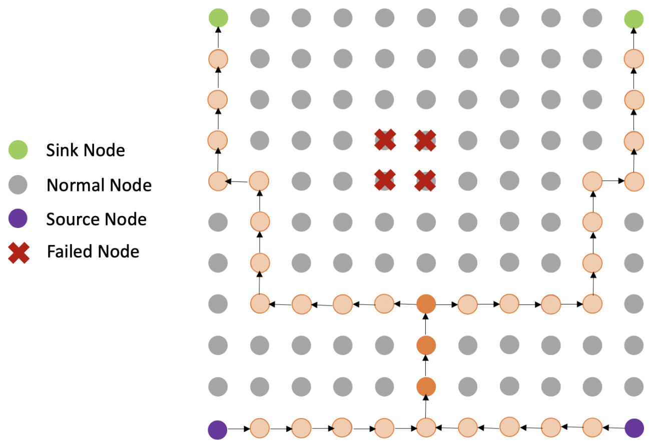


Figure 6.18: Patterned Failure on the backbone and an example route.

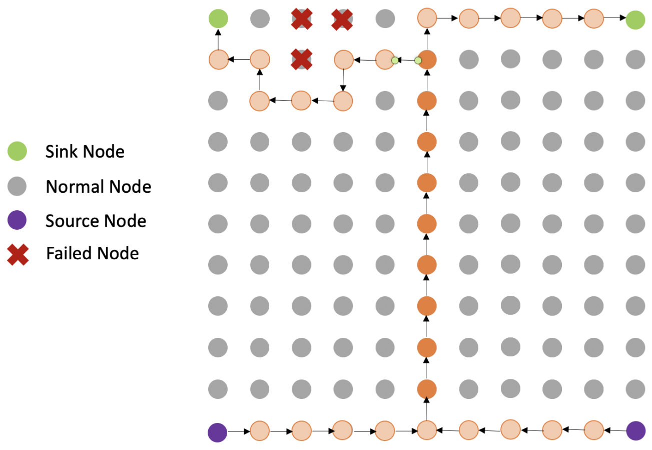


Figure 6.19: Patterned Failure after the backbone split and an example route.

form the expected route, figure 6.20. The mean delivery ratio is lower than that with no failures, 85% $\pm$ 0.9 and 73% $\pm$ 2 for base and all sinks respectively. This could indicate that the fault detecting beacon ants may be affecting the performance of the protocol. Figure 6.20 shows an example route, however this is not consistent between experiments.

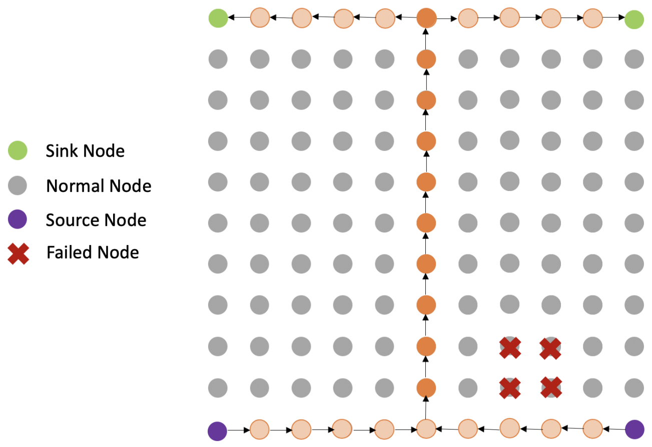


Figure 6.20: Patterned Failure not on the main route and an example route.

## 6.5 Conclusion

This chapter has presented an alternate version of the ACO based routing protocol for many-to-many wireless sensor networks described in the previous two chapters in order to create Steiner Trees in grid based networks. Steiner Trees have many potential uses in WSN, included the connection of an arbitrary number of sources and sinks. In this chapter, the terminal points of Steiner Trees were considered to be the multiple sources and sinks in the network, and were connected with a routing path created by ants. In order to do this, a novel pheromone trail initialisation process was developed, which initialised pheromone trail values between nodes before data delivery took place in order to encourage trees to be created. This was successful for various numbers of sources and sinks whilst still keeping the requirement that each node had only local knowledge of the global network topology. The protocol still is able to function without ant memory, and each node only has to store a bloom filter of consistent size. This has the advantage of needing to store a constant amount of information, regardless of the number of sources and sinks, showing the scaleability of the protocol. The protocol is also able to be used for networks with multiple sinks without repeating the protocol for each sink.

In addition to the ACO based Steiner Tree protocol, a fault tolerant variant of the protocol was developed, which shows potential in dealing with node failures in the network while still being able to deliver from sources to sinks. The protocol was less successful when considering only routes where all sources and all sinks were included on route, however high base delivery ratio shows that most of the time at least some data was still being successfully delivered. This could indicate that the use of a Steiner Tree as a whole could be a fault tolerant technique, as it enables that at least some packets were consistently delivered from at least one source to one sink regardless of node failures.

# Chapter 7

# Discussion, Evaluation, and Future Work

Ant Colony Optimisation can be used as the basis for a framework that can generate routing protocols for solving many routing problems in Wireless Sensor Networks. It has been shown that the ACO framework is successful in grid based networks in the routing of messages from multiple sources to multiple sinks, a problem that is little researched by existing work despite having the potential for many situation as WSN become more sophisticated. The protocol has successful continuous data delivery with high delivery ratios, whilst also remaining efficient and scaleable. The ACO framework can also be used for fault tolerance routing protocol, being able to successfully recovery from faults with the novel introduction of "beacon ants", as well as the use of targeted pheromone evaporation. Finally ACO framework for routing protocols was adapted in order to find Steiner Trees in WSN using the same basic concepts, successfully forming Steiner trees in a distributed fashion. These variations of routing protocol based on the same ACO based framework show how the meta heuristic can be adapted to a number of different problems, without needed a specific set of starting conditions. The flexibility and scalebility of the protocol leads it to many potential uses and applications.

This chapter presents a discussion of the ACO based routing protocol as a framework for varied WSN based routing problems, including its limitations. Observations that were made through the course of development are noted here that may lead to future applications of the protocol being more successful.

## 7.1 Observations

This section will discuss various observations through the course of the work around the concept of ACO in routing in many-to-many WSN.

Generally, it was found that the protocol was more successful when the network topology was more symmetrical with how the sources and sinks were laid out. For instance, if the sources and sinks are all in the corners of the network, it was easier for ants to combine and form a backbone. This is likely due to synchronisation of ants; when they are travelling approximately the same distance to get to the backbone, it is easier to time the combine timer to wait for the backbone to form. This could be helpful in deciding where to place sources and sinks in the network topology when setting up the network. It must be noted that the protocol was tested only on a grid topology, and so the observations of symmetrically laid out networks may not apply if the network itself is not symmetrical.

In the course of this work, it was found that the choice of input parameters was very important for the success of the protocol. For instance, changing the evaporation constant meant a balance between consistency in routes, choosing nodes that led to a previously successful path, and making new choices in order to improve the path. In this course of this work, input parameters were largely chosen through experimentation, with the most successful values being used throughout. It is important to note that the variation of input parameters has a great deal of impact of the overall success of the routing protocols, and also has an impact on how adaptable the framework is for different scenarios. Again looking at evaporation constant, this could be varied based on the desired outcomes for a particular applications. If the goal is consistency and quickly forming routes, less evaporation should occur.

One of the goals of this work was to create an ACO based framework that could be extended and adjusted for the use of solving many problems. Though most aspects of the protocol were kept the same, the base concepts remained throughout. For instance, the probabilistic choice of which node to travel to next by the ants was a consistent factor in all protocols, but the exact equation used to decide this was changed. Different problems required different priorities, so the equation is a way of adapting to these without making large changes to the base algorithm. This shows how the framework is adaptable to new problems. Other important aspects that could be considered for a change for a new problem include; which nodes combine and split, how much information each node holds about its neighbours, the equation for backward ant trail updates. There is also the potential for the reintroduction of ant memory for certain problems, especially if payload is not a concern.

Many lessons were learnt in the course of developing the fault tolerant extension of the ACO protocol for successful data delivery. Generally, a balance needed to be struck between encouraging ants to avoid failed nodes, and completely disregarding a node when it was detected as a potential failure. It was important to set the trail evaporation to be high enough for an ant to be more likely to choose and alternative route, whilst also not leading to the complete break down of routes leading to entirely random choices. Often nodes were detected as failed when they were fine, and it could be conceivable in real world scenarios where a node may be temporarily unavailable, so it was desired to not discount the node completely. This also means that the fault detection using the beacon ants required experimentation to determine input parameters such as window size. A larger window size meant that it would take longer for faults to be detected, but a small window would lead to more false positives.

It was found that with larger network sizes, the benefits of using ACO became greater. For instance, the larger networks led to a larger proportion of the total route length consisting of the backbone, and so also relatively fewer nodes involved. This means fewer energy expanded in sending messages and so a greater network lifetime. This scaleability of the protocol is a large advantage, as it performs better with larger networks, not worse as with much previous work in WSN routing. This means that the ACO based framework for routing in WSN is well suited for large networks, making them durable and reliable.

## 7.2 Limitations

There were some limitations in synchronisation of the forward ants, i.e., when combining to form a backbone, if a source is very close to where the backbone will be, it may tend to travel an artificially long route in order to arrive at the backbone at the same time as ants originating further away. This could lead to inefficiency in the routes used, however it was shown that for most part delivery ratio remained high when these synchronisation issues occurs. This could be improved by changing the timer length, or with more experimentation of input parameters.

There are also some limitation in the protocol when it comes to splitting the backbone, as sometimes it may split early, leading to inefficient routes. This is a consequence of the distributed nature of the protocol, and is hard to avoid as sometimes a route may look more efficient to an ant when it is not.

The requirement of using backward ants could also be considered a limitation of the protocol as it increases the total number of messages needing to be sent. Many existing works that make use of ACO in WSN also use backward ants, as it is necessary to use them to update pheromone trail after completion of the routes, as there is no way of knowing how successful the route is until it has finished. Future work could be carried out to lessen the impact of backward ants, for instance reducing the frequency of backward ant launches as the algorithm runs. This could take the form of, for instance, after $x$ cycles, backward ants will only be launched from the sinks every other cycle.

## 7.3 Evaluation

This work presented an ACO based framework for routing in many-to-many wireless sensor networks, that could be used to generate various routing protocols for a number of different problems. These ACO based routing protocols are effective in successfully delivering messages from sources to sinks in a wide range of network sizes and topologies in an efficient and scaleable way. The use of the ACO meta heuristic allows the protocol to be applied to a number of different problems, without needed a specific set of starting conditions. This chapter both summarises the work presented, and also explores future directions that could follow from the work.

### 7.3.1 Conclusion

As WSN are more commonly used, a larger number of applications will emerge that will require more complicated starting conditions and topologies, for instance multiple sources and multiple sinks. For this reason, it is necessary for routing protocols that can be applied to a wide range of starting conditions. For this reason, meta heuristics are chosen as the focus of this work, and so an Ant Colony Optimisation based framework that can be applied to wide ranging and more complicated applications. The ACO based framework has been used to develop a number of routing protocols for WSN, while the base concepts of ACO remain constant.

The routing protocols developed in this work have been shown to be successful in delivering data in grid based networks with varied number of sources and sinks. Results also show that the protocol scales very well in larger network size in terms of minimising the number of messages sent and overall path length, while maintaining a high delivery ratio. To expand on this, as the network size gets larger, the protocol does not used proportionally larger resources. If the network size increases, the number of messages sent and the path length of routes between sources and sinks does not increase proportionally, but is efficient in terms of path length and messages sent. It was also found that as the network got larger, certain benefits such as the ability to aggregate data improved, as the backbone forms a larger part of the total route. Due to these factors, the protocol can be considered scaleable. Additionally, the fault tolerant variant is able to maintain this high delivery ratio while recovering from a relatively large amount of faults in the network. The ACO protocol was also shown to be able to create Steiner Trees in a WSN and successfully deliver between Steiner points on the tree.

In ACO, agents known as ants travel between nodes in a graph laying "pheromone trail" between the nodes. The amount of pheromone trail laid is proportional to the success of the route, and is also periodically evaporated. Ants travel between nodes choosing the next node to travel to using a probability function based on a measure of distance to the goal, and the amount of trail between the nodes. Through the use of pheromone trail, successful routes form over time. The ACO protocol developed for WSN took this concept and used it to create short, efficient routes. The protocol used the concept of a shared path, or "backbone" in order to make the most of data aggregation to improve scalebility; more nodes on the backbone meant that fewer messages were needed to be sent with increased network size. Both forward and backward ants were used, forward to move data from sources to sinks, and backward travel from sink to source to update pheromone trails. The protocol was successful at delivering all data from all sources to all sinks with a minimal number of nodes and messages. It was scaleable and energy efficient, getting more efficient with larger networks. The scalebility comes from the fact that the protocol forms short routes between sources and sinks, using proportionally fewer nodes compared to the total network size.

The first version of the protocol required the use of a concept called “ant memory”. This is where each ant travelling through the network kept track of the nodes it had visited so far. This was used to ensure no nodes were repeated in the route, as well as by backward ants to know the route they needed to travel along to update pheromone trail. This had the advantage of preventing repeated nodes, but at the cost of having to carry more information in the payload. This is a very commonly used concept in most ACO protocols presented in previous works, however has its limitations as it increases the size of the payload of each ant reducing scaleability. A variation on the protocol was produced that had no ant memory, instead each node on being visited by an ant kept track of the previously visited node by the ant, and the node the ant chose to travel to next. This means meant the payload was smaller, increasing the effectiveness of the protocol with larger networks as the amount of information each node and ant needs to keep track of remains the same regardless of the route length. This enables backward ants to travel the correct route, however did not necessarily prevented repeat visits. It was found in experiments that the removal of ant memory did not lead to repeat visits, and in many ways improved performance in terms of delivery ratio and other metrics.

Due to their limited energy supply and the types of environments that WSNs are often used in, it is not uncommon and often expected for sensor nodes to fail over the course of the lifetime of the network. In these situations, it is advantageous for a routing protocol to recovery from these faults and still be able to deliver data from sources to sinks. A fault tolerant variant of the ACO based routing protocol for many-to-many sensor networks is therefore introduced. This protocol introduces the concepts of “beacon ants”. Beacon ants are periodically broadcast between nodes in order to detect faulty nodes. A window of the number of beacon ants received from each neighbours is kept by each node, and if no beacon ant has been received for a predetermined time (dictated by the window size), the neighbour may have failed. The pheromone trail between the detecting node and the potential faulty node is then evaporated proportionally to the likelihood that the node has failed using targeted pheromone evaporation. This discourages ants from travelling to the node in the future, however does not entirely discount it in case the node is actually not faulty. The protocol remains scaleable and efficient, but also is able to successfully recover from faults in the network for both random and patterned failures. The scaleability remains in terms of the route length and messages sent from the base ACO protocol. As the fault tolerance emerges locally, that is each node only needs to keep track of the faults of its immediate neighbours, the protocol will not become less efficient with larger networks, only more neighbours. Recovery time is often low, partially because routes formed by the ACO routing protocol use a low proportion of the nodes to begin with, but also due to the ants only making local decisions about routes, they can easily find a path to travel around most failed nodes.

A variation on the protocol was developed for creating Steiner Trees between nodes in a grid based wireless sensor network. Steiner Trees are helpful in a number of scenarios, as they are a cost effective way of connecting sources and sinks. This takes the concepts from the previous ACO based protocols and introduces an pheromone initialisation step. Before data delivery takes place, pheromone initialisation occurs in order to encourage Steiner Trees to form in the network. This consists of simultaneous flooding from sources and sinks, with pheromone being laid between nodes on routes to where separate floods meet. This version of the protocol has no measure of distance in the probability function used by ants to travel between nodes, only pheromone trail. This means that the information necessary for each node to store is much lower, and is the same size regardless of how many sources and sinks it needs to keep track of. Additionally changes were made to how the backbone was created and split, in order to facilitate more sources and sinks. This version of the protocol was able to create Steiner Trees between arbitrary number of sources and sinks in grid based networks while maintaining performance metrics such as delivery ratio and packets sent. Additionally, a fault tolerant variant of this protocol was developed that used beacon ants and targeted pheromone evaporation, that showed some level of tolerance to faults, though less successfully than the previous fault tolerant protocol.

To summarise, an ACO-based framework was developed for routing in many-to-many WSN, tested in a variety of grid based networks. The framework makes use of meta heuristics in order to solve varied problems without the need for specific starting conditions. The protocols are all distributed, with only local decisions being made by nodes. Results have shown the routing protocols developed from the ACO based framework are successful at continuous data delivery, achieving high delivery ratios even in the presence of faults. The protocol is very scaleable, often become more efficient as network size increases due to the positive effects of aggregation along the backbone. The fault tolerant routing protocol could recover from faults quick and maintain this high delivery ratio whilst also still following efficient routes. Finally, the ACO framework was used to create a protocol for Steiner Tree routing, which meant that routes could be formed for a large range of grid based network topologies. Overall this shows the flexibility of the ACO framework, in its ability to successfully routing messages in many-to-many networks for a wide range of starting conditions and network topologies.

### 7.3.2 Future Work

The work presented in this thesis used Ant Colony Optimisation in order to create efficient and scaleable routing protocols for many-to-many wireless sensor networks that are tolerant to faults. However, the work my be extended and applied in many different possible ways, and so some of the potential future directions for the work are presented in this section.

**Optimisation Variables**

The work here focuses on messages sent and nodes involved as an optimisation variable for efficient routes, however there are other possible variables to consider. For instance, energy remaining on the nodes could form an interesting path of future research. The protocol could be adapted to take energy remaining of the nodes into account when selecting the next node to travel to, and so increasing the lifetime of the network in this way. This could form an energy aware variation of the ACO based routing protocol, based on the main concepts of the ACO framework. This would be implemented through changes to the equations used by ants to select the next node to travel to. In the current work the probability function is based on hop counts and pheromone value, an energy aware probability function would also include a factor considering the expected energy cost to travel to this node, and/or the energy remaining on that node. The goal here would be to encourage ants to travel paths that require less energy, or to avoid nodes with less energy remaining. The energy required for paths could be found in the set up phase, whereas the energy remaining on nodes could be found using the beacon ants.

**Improved Fault Tolerant Steiner Tree Routing**

The fault tolerant variant of the ACO based routing protocol for creating Steiner Trees in WSN showed promise, however was not entirely successful. It appears that the repeated broadcasting of the beacon ants interfered with the routing process, and so led to less successful routes. Future work should be carried out to improve on this aspect of the protocol, perhaps through more targeted broadcast of beacon ants. For instance, beacon ants could only be launched if a node expects backward ants but has not received any, or as part of the backward ant route in order to only detect failed nodes on the route rather than in the whole network. This would reduce the amount of beacon ants, and therefore have less chance of interupting the path of forward ants. Another method of improving fault tolerant Steiner tree routing would be to investigate changes in the input parameters. Input parameters such as evaporation constant remained constant in the investigation, however lowering this figure, and so increasing the evaporation, could lead to better results. This is because ants will be forced to find alternative routes earlier, as the pheromone will evaporate more quickly. To improve fault tolerance, there is also the potential of rethinking the fault tolerance mechanism entirely, and instead of using beacon ants utilise backward ants more in the fault detection process. Backward ants could be sent along previously successful routes if sinks have not received forward ants in a while in order to detect where along that route the fault is.

### Reduction In Backward Ants Sent

Backward ants form a substantial amount of total packages sent throughout the running of the protocol, however when the forward ants have converged on a route for long enough, backward ants often end up being redundant. A future path of research could be to find ways to reduce the number of backward ants needed, for instance backward ants could be sent every cycle at the start of the continuous data delivery, but the rate of backward ants would be reduced over time. For instance, after 10 successful cycles of data delivery, backward ants would only be launched every other cycle with the evaporation rate being reduced in order to maintain pheromone values. Alternatively, backward ants could also only be launched by the sink the route is changing, for instance if the route length is different, in order to update pheromone trails based on this new route. Reducing the amount of backward ants sent may also require some level of reduction of pheromone evaporation, such that routes are still maintained. This would have the effect of reducing the amount of messages sent, and so decreasing energy used and increasing network lifetime.

### Adaptation for non grid topologies

It would be advantageous to adapt the protocol for no grid based topologies, as the testing carried out in this work was on grid networks. This would likely not involve too many changes of the protocol, though perhaps would need some experimentation in the appropriate input parameters. For instance, it may be advantageous to increase the evaporation rate, as it may require more experimentation from the ants in order to find the optimal route. It would be useful to investigate how the efficiency of the protocol scales with the number of neighbours nodes have, as this would evaluatate scaleability for both total network size and the connectivity of the network.

**Investigation into Mobile Nodes**

An interesting branch of research would be to experiment with ACO based routing protocols in networks with a number of mobile nodes. This was beyond the scope of this work due to resources, however could be developed into a successful variant of the protocol without too many changes outside of input parameters. An advantage of the protocol is it's adaptability, and so investigating this network setup would help to prove this.

**Genetic Algorithms**

The use of genetic algorithms for optimal parameter selection could be a possible extension area. In the work presented in this thesis, a number of input parameters are required to be selected, such as evaporation rate. These parameters were chosen based on a combination of logical reasoning based on desired behaviours, and tested with repeated simulations. However, the application of genetic algorithms could both speed up this process as well as finding more effective input parameters, allowing the protocols to be more effective in a wide range of environments, without having the repeated test for differing figures. This allows the ACO based framework to be more easily adapted for future problems.

**Open Source Protocol Suite**

A potential area of future work is in the release of an open source suite of the ACO based routing protocols for WSN presented this thesis. Such a suite would be adapted in such as a way as to enable easy use of the protocol for anyone wishing to set up a WSN. This work would involve changes to the ContikiOS implementation such that the protocol could be changed by the end user without needing to change the source code, such as through a user interface. This user interface would potentially set up the protocol for range of sources and sinks, and allow the user to change input parameters. This could also be combined with the future work using genetic algorithms, in order to optimise settings for a particular WSN setup.

# Bibliography

[1] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2003.09.010. URL `http://www.sciencedirect.com/science/article/pii/S1570870503000738`.

[2] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8): 102–114, Aug 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1024422.

[3] Ian F. Akyildiz and IsWireless Sensor Networks for Habitat Monitoringmail H. Kasimoglu. Wireless sensor and actorâwe refer to entities that can act on the network as actors they are sometimes referred to as actuators in related literature.â networks: research challenges. *Ad Hoc Networks*, 2(4):351 – 367, 2004. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2004.04.003. URL `http://www.sciencedirect.com/science/article/pii/S1570870504000319`.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002. ISSN 1389-1286. doi: https://doi.org/10.1016/S1389-1286(01)00302-4. URL `http://www.sciencedirect.com/science/article/pii/S1389128601003024`.

[5] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor

networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, 2004. doi: 10.1109/MWC.2004.1368893.

[6] S. Ali, A. Fakoorian, and H. Taheri. Optimum reed-solomon erasure coding in fault tolerant sensor networks. In *2007 4th International Symposium on Wireless Communication Systems*, pages 6–10, 2007. doi: 10.1109/ISWCS.2007.4392291.

[7] H. Alwan and A. Agarwal. A survey on fault tolerant routing techniques in wireless sensor networks. In *2009 Third International Conference on Sensor Technologies and Applications*, pages 366–371, 2009. doi: 10.1109/SENSORCOMM.2009.62.

[8] Isaac Amundson and Xenofon D. Koutsoukos. A survey on localization for mobile wireless sensor networks. In Richard Fuller and Xenofon D. Koutsoukos, editors, *Mobile Entity Localization and Tracking in GPS-less Environnments*, pages 235–254, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04385-7.

[9] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt. Riot os: Towards an os for the internet of things. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 79–80, 2013. doi: 10.1109/INFCOMW.2013.6970748.

[10] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*, 8(4):34:1–34:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240123. URL `http://doi.acm.org/10.1145/2240116.2240123`.

[11] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Out-of-the-box environmental monitoring. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 332–343, 2008. doi: 10.1109/IPSN.2008.28.

[12] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[13] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 22–31, New York, NY, USA, 2002. ACM. ISBN 1-58113-589-0. doi: 10.1145/570738.570742. URL `http://doi.acm.org/10.1145/570738.570742`.

[14] Wenyu Cai, Xinyu Jin, Yu Zhang, Kangsheng Chen, and Rui Wang. Aco based qos routing algorithm for wireless sensor networks. In Jianhua Ma, Hai Jin, Laurence T. Yang, and Jeffrey J.-P. Tsai, editors, *Ubiquitous Intelligence and Computing*, pages 419–428, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[15] Tiago Camilo, Carlos Carreto, Jorge Sá Silva, and Fernando Boavida. An energy-efficient ant-based routing algorithm for wireless sensor networks. In Marco Dorigo, Luca Maria Gambardella, Mauro Birattari, Alcherio Martinoli, Riccardo Poli, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, pages 49–59, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-38483-0.

[16] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *2009 International Conference on Information Processing in Sensor Networks*, pages 277–288, 2009.

[17] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Proceedings of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, DIWANS '06, page 65–72, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595934715. doi: 10.1145/1160972.1160985. URL `https://doi.org/10.1145/1160972.1160985`.

[18] Yuequan Chen, Edward Chan, and Song Han. Energy efficient multipath routing in large scale sensor networks with multiple sink nodes. In Jiannong Cao, Wolfgang Nejdl, and Ming Xu, editors, *Advanced Parallel Processing Technologies*, pages 390–399, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32107-1.

[19] T. Clouqueur, K. K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. *IEEE Transactions on Computers*, 53(3):320–333, 2004. doi: 10.1109/TC.2004.1261838.

[20] D Costa and A Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48(3):295–305, 1997. doi: 10.1057/palgrave.jors.2600357.

[21] B. Deb, S. Bhatnagar, and B. Nath. Reinform: reliable information forwarding using multiple paths in sensor networks. In *28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN '03. Proceedings.*, pages 406–415, 2003. doi: 10.1109/LCN.2003.1243166.

[22] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. A topology discovery algorithm for sensor networks with applications to network management, 2002.

[23] D. S. Deif and Y. Gadallah. An ant colony optimization approach for the deployment of reliable wireless sensor networks. *IEEE Access*, 5: 10744–10756, 2017. doi: 10.1109/ACCESS.2017.2711484.

[24] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *J. Artif. Int. Res.*, 9(1):317–365, December 1998. ISSN 1076-9757. URL `http://dl.acm.org/citation.cfm?id=1622797.1622806`.

[25] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications*

*Societies.*, volume 2, pages 902–913 vol. 2, 2005. doi: 10.1109/INFCOM.2005.1498320.

[26] Niannian Ding, P. X. Liu, and Chao Hu. Data gathering communication in wireless sensor networks using ant colony optimization. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 697–702, Aug 2005. doi: 10.1109/IROS.2005.1545067.

[27] P. Djukic and S. Valaee. Minimum energy fault tolerant sensor networks. In *IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004.*, pages 22–26, 2004. doi: 10.1109/GLOCOMW.2004.1417543.

[28] P. Djukic and S. Valaee. Maximum network lifetime in fault tolerant sensor networks. In *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, volume 5, pages 5 pp.–3106, 2005. doi: 10.1109/GLOCOM.2005.1578328.

[29] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477 Vol. 2, 1999. doi: 10.1109/CEC.1999.782657.

[30] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997. doi: 10.1109/4235.585892.

[31] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, Feb 1996. ISSN 1083-4419. doi: 10.1109/3477.484436.

[32] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE*

*Computational Intelligence Magazine*, 1(4):28–39, 2006. doi: 10.1109/MCI.2006.329691.

[33] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy. Technical report, 1991.

[34] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 3, pages 1918–1922 vol.3, 2003. doi: 10.1109/WCNC.2003.1200680.

[35] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN '04, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2260-2. doi: 10.1109/LCN.2004.38. URL `https://doi.org/10.1109/LCN.2004.38`.

[36] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29$^{th}$ Annual IEEE International Conference on*, pages 455–462, 2004. doi: 10.1109/LCN.2004.38.

[37] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 4, pages 2033–2036 vol.4, 2001. doi: 10.1109/ICASSP.2001.940390.

[38] K. Fathima and Kumar Sindhanaiselvan. Ant colony optimization based routing in wireless sensor networks. *Int. J. Adv. Netw. Appl.*, 4:1686–1689, 02 2013.

[39] E. Felemban, Chang-Gun Lee, and E. Ekici. Mmspeed: multipath multi-

speed protocol for qos guarantee of reliability and. timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006. doi: 10.1109/TMC.2006.79.

[40] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric tsps by ant colonies. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 622–627, 1996. doi: 10.1109/ICEC.1996.542672.

[41] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

[42] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, October 2001. ISSN 1559-1662. doi: 10.1145/509506.509514. URL `https://doi.org/10.1145/509506.509514`.

[43] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977. doi: 10.1137/0132072. URL `https://doi.org/10.1137/0132072`.

[44] Chen Gen-Huey, Michael E. Houle, and Kuo Ming-Ter. The steiner problem in distributed computing systems. *Information Sciences*, 74(1):73 – 96, 1993. ISSN 0020-0255. doi: https://doi.org/10.1016/0020-0255(93)90128-9. URL `http://www.sciencedirect.com/science/article/pii/0020025593901289`.

[45] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968. doi: 10.1137/0116001. URL `https://doi.org/10.1137/0116001`.

[46] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss,

and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-519-2. doi: 10.1145/1644038.1644040. URL `http://doi.acm.org/10.1145/1644038.1644040`.

[47] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[48] J. Grosso, A. Jhumka, and M. Bradbury. Reliable many-to-many routing in wireless sensor networks using ant colony optimisation. In *2019 15th European Dependable Computing Conference (EDCC)*, pages 111–118, 2019. doi: 10.1109/EDCC.2019.00030.

[49] Jasmine Grosso and Arshad Jhumka. Fault-tolerant ant colony based-routing in many-to-many iot sensor networks. pages 1–10, 11 2021. doi: 10.1109/NCA53618.2021.9685935.

[50] J. Gutiérrez, J. F. Villa-Medina, A. Nieto-Garibay, and M. Á. Porta-Gándara. Automated irrigation system using a wireless sensor network and gprs module. *IEEE Transactions on Instrumentation and Measurement*, 63(1):166–176, 2014. doi: 10.1109/TIM.2013.2276487.

[51] M. J. Handy, M. Haase, and D. Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *4th International Workshop on Mobile and Wireless Communications Network*, pages 368–372, 2002. doi: 10.1109/MWCN.2002.1045790.

[52] H. Hassanein and Jing Luo. Reliable energy aware routing in wireless sensor networks. In *Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pages 54–64, 2006. doi: 10.1109/DSSNS.2006.10.

[53] Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liest-

man. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988. doi: https://doi.org/10.1002/net.3230180406. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230180406`.

[54] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, page 174–185, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131429. doi: 10.1145/313451.313529. URL `https://doi.org/10.1145/313451.313529`.

[55] Chih-fan Hsin and Mingyan Liu. Self-monitoring of wireless sensor networks. *Computer Communications*, 29, 07 2004. doi: 10.1016/j.comcom.2004.12.031.

[56] Yu Hu, Tong Jing, Zhe Feng, Xian-Long Hong, Xiao-Dong Hu, and Gui-Ying Yan. Aco-steiner: Ant colony optimization based rectilinear steiner minimal tree algorithm. *Journal of Computer Science and Technology*, 21(1):147–152, 2006.

[57] F. K. Hwang and Dana S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992. doi: 10.1002/net.3230220105. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230220105`.

[58] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003. doi: 10.1109/TNET.2002.808417.

[59] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, February 2003. ISSN 1063-6692. doi: 10.1109/TNET.2002.808417. URL `http://dx.doi.org/10.1109/TNET.2002.808417`.

[60] V. Isler, S. Kannan, and K. Daniilidis. Sampling based sensor-network deployment. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1780–1785 vol.2, 2004. doi: 10.1109/IROS.2004.1389654.

[61] Won-Suk Jang, William M. Healy, and MirosÅaw J. Skibniewski. Wireless sensor networks as part of a web-based building environmental monitoring system. *Automation in Construction*, 17(6):729 – 736, 2008. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2008.02.001. URL `http://www.sciencedirect.com/science/article/pii/S0926580508000174`.

[62] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *J. of Global Optimization*, 39(3):459–471, November 2007. ISSN 0925-5001. doi: 10.1007/s10898-007-9149-x. URL `https://doi.org/10.1007/s10898-007-9149-x`.

[63] Dervis Karaboga, Selcuk Okdem, and Celal Ozturk. Cluster based wireless sensor network routing using artificial bee colony algorithm. *Wireless Networks*, 18(7):847–860, 2012.

[64] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL `https://doi.org/10.1007/978-1-4684-2001-2_9`.

[65] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995. doi: 10.1109/ICNN.1995.488968.

[66] Michael Kintner-Meyer, Michael Brambley, Teresa Carlon, and Nathan Bauman. Wireless sensors: Technology and cost-savings for commercial buildings. *2002 ACEEE Summer Study on Energy Efficiency in Buildings*, 11, 01 2002.

[67] Y. Kiri, M. Sugano, and M. Murata. Self-organized data-gathering scheme for multi-sink sensor networks inspired by swarm intelligence. In *First International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, pages 161–172, 2007. doi: 10.1109/SASO.2007.52.

[68] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1 (3):286–292, 1993. doi: 10.1109/90.234851.

[69] Sookyoung Lee and Mohamed Younis. Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree. *Journal of Parallel and Distributed Computing*, 70(5):525 – 536, 2010. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2009.12.004. URL `http://www.sciencedirect.com/science/article/pii/S0743731509002433`.

[70] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. doi: 10.1002/net.3230110211.

[71] Lucas Lessing, Irina Dumitrescu, and Thomas Stützle. A comparison between aco algorithms for the set covering problem. In Marco Dorigo, Mauro Birattari, Christian Blum, Luca Maria Gambardella, Francesco Mondada, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, pages 1–12, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-28646-2.

[72] L. Li and J. Y. Halpern. Minimum-energy mobile wireless networks revisited. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 1, pages 278–283 vol.1, 2001. doi: 10.1109/ICC.2001.936317.

[73] Wen-Hwa Liao, Yucheng Kao, and Ying-Shan Li. A sensor deployment approach using glowworm swarm optimization algorithm in wireless sensor networks. *Expert Systems with Applications*, 38(10):12180 –

12188, 2011. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2011.03.053. URL `http://www.sciencedirect.com/science/article/pii/S0957417411004611`.

[74] S. Lindsey and C. S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Proceedings, IEEE Aerospace Conference*, volume 3, pages 3–1125–3–1130 vol.3, 2002. doi: 10.1109/AERO.2002.1035242.

[75] Bing-Hong Liu, Ngoc-Tu Nguyen, Van-Trung Pham, and Wei-Sheng Wang. Constrained node-weighted steiner tree based algorithms for constructing a wireless sensor network to cover maximum weighted critical square grids. *Computer Communications*, 81:52 – 60, 2016. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2015.07.027. URL `http://www.sciencedirect.com/science/article/pii/S0140366415002844`.

[76] L. Liu, Y. Song, H. Zhang, H. Ma, and A. V. Vasilakos. Physarum optimization: A biology-inspired algorithm for the steiner tree problem in networks. *IEEE Transactions on Computers*, 64(3):818–831, 2015. doi: 10.1109/TC.2013.229.

[77] E. L. Lloyd and G. Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, 2007. doi: 10.1109/TC.2007.250629.

[78] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang. Ttdd: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks*, 11(1):161–175, 2005.

[79] J. Lynch. An overview of wireless structural health monitoring for civil structures. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365:345 – 372, 2006.

[80] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996. ISBN 9780080504704.

[81] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, page 88–97, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135890. doi: 10.1145/570738.570751. URL `https://doi.org/10.1145/570738.570751`.

[82] I. I. Mandoiu, V. V. Vazirani, and J. L. Ganley. A new heuristic for rectilinear steiner trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1129–1139, Oct 2000. ISSN 1937-4151. doi: 10.1109/43.875292.

[83] Kamalrulnizam Bin Abu Bakar Malrey Lee Marjan Radi, Behnam Dezfouli. Multipath routing in wireless sensor networks: Survey and research challenges. *Sensors*, 12(1):650–685, 2012. ISSN 1424-8220. doi: 10.3390/s120100650. URL `https://www.mdpi.com/1424-8220/12/1/650/pdf`.

[84] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, page 255–265, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581131976. doi: 10.1145/345910.345955. URL `https://doi.org/10.1145/345910.345955`.

[85] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1380–1387 vol.3, 2001. doi: 10.1109/INFCOM.2001.916633.

[86] Vinod Menaria, S.C. Jain, and Nagaraju Autha. A fault tolerance based route optimisation and data aggregation using artificial intelligence to enhance performance in wireless sensor networks. *International Journal of Wireless and Mobile Computing*, 14:123, 01 2018. doi: 10.1504/IJWMC.2018.091139.

[87] Manki Min, Hongwei Du, Xiaohua Jia, Christina Xiao Huang, Scott C. H. Huang, and Weili Wu. Improving construction for connected dominating set with steiner tree in wireless sensor networks. *Journal of Global Optimization*, 35(1):111–119, 2006.

[88] Raquel A.F. Mini, Antonio A.F. Loureiro, and Badri Nath. The distinctive design characteristic of a wireless sensor network: the energy map. *Computer Communications*, 27(10):935 – 945, 2004. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2004.01.004. URL `http://www.sciencedirect.com/science/article/pii/S0140366404000143`. Protocol Engineering for Wired and Wireless Networks.

[89] L. Mottola and G. P. Picco. Muster: Adaptive energy-aware multisink routing in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 10(12):1694–1709, Dec 2011. ISSN 1536-1233. doi: 10.1109/TMC.2010.250.

[90] T. Naumowicz, R. Freeman, H. Kirk, B. Dean, M. Calsyn, A. Liers, A. Braendle, T. Guilford, and J. Schiller. Wireless sensor network for habitat monitoring on skomer island. In *IEEE Local Computer Network Conference*, pages 882–889, 2010. doi: 10.1109/LCN.2010.5735827.

[91] Roman Novak, Jože Rugelj, and Gorazd Kandus. "steiner tree based distributed multicast routing in networks," steiner trees in industries. 11, 01 2000. doi: 10.1007/978-1-4613-0255-1_10.

[92] Luis Oliveira and Joel Rodrigues. Wireless sensor networks: A survey on environmental monitoring. *JCM*, 6:143–151, 04 2011. doi: 10.4304/jcm.6.2.143-151.

[93] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *31st IEEE Conference on Local Computer Networks*, pages 641–648, November 2006. doi: 10.1109/LCN.2006.322172.

[94] Mohd Fauzi Othman and Khairunnisa Shazali. Wireless sensor network applications: A study in environment monitoring system. *Procedia Engineering*, 41:1204 – 1210, 2012. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2012.07.302. URL `http://www.sciencedirect.com/science/article/pii/S1877705812027026`. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[95] E. I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless sensor networks. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 6, pages 3663–3667 Vol.6, June 2004. doi: 10.1109/ICC.2004.1313226.

[96] Lilia Paradis and Qi Han. A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2): 171–190, 2007. doi: 10.1007/s10922-007-9062-0. URL `https://doi.org/10.1007/s10922-007-9062-0`.

[97] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957. doi: 10.1002/j.1538-7305.1957.tb01515.x.

[98] Markus Prossegger and Abdelhamid Bouchachia. Ant colony optimization for steiner tree problems. CSTST '08, page 331–336, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580463. doi: 10.1145/1456223.1456292. URL `https://doi.org/10.1145/1456223.1456292`.

[99] Rong Qu, Ying Xu, Juan P. Castro, and Dario Landa-Silva. Particle swarm optimization for the steiner tree in graph and delay-constrained multicast routing problems. *Journal of Heuristics*, 19(2):317–342, 2013.

[100] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. SenSys '05, page 255–267, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 159593054X. doi: 10.1145/1098918.1098946. URL https://doi.org/10.1145/1098918.1098946.

[101] Chris R. Reid, David J. T. Sumpter, and Madeleine Beekman. Optimisation in a natural system: Argentine ants solve the towers of hanoi. *Journal of Experimental Biology*, 214(1):50–58, 2011. ISSN 0022-0949. doi: 10.1242/jeb.048173. URL https://jeb.biologists.org/content/214/1/50.

[102] Marc Reimann, Karl Doerner, and Richard F Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers Operations Research*, 31(4):563 – 591, 2004. ISSN 0305-0548. doi: https://doi.org/10.1016/S0305-0548(03)00014-5. URL http://www.sciencedirect.com/science/article/pii/S0305054803000145.

[103] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM computer communication review*, 27(2): 24–36, 1997.

[104] G. Robins and A. Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discrete Math.*, 19:122–134, 01 2005. doi: 10.1137/S0895480101393155.

[105] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, 1999. doi: 10.1109/49.779917.

[106] JoÅŸe Rugelj and Sandi KlavÅŸar. Distributed multicast routing in point-to-point networks. *Computers Operations Research*, 24(6):521 – 527, 1997. ISSN 0305-0548. doi: https://doi.org/10.1016/S0305-0548(96)00074-3. URL http://www.sciencedirect.com/science/article/pii/S0305054896000743.

[107] Muhammad Saleem and Muddassar Farooq. Beesensor: a bee-inspired power aware routing protocol for wireless sensor networks. In *Workshops on Applications of Evolutionary Computation*, pages 81–90. Springer, 2007.

[108] C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. In *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, volume 1, pages 357–361 vol.1, 2001. doi: 10.1109/MILCOM.2001.985819.

[109] Noman Shabbir and Syed Hassan. *Routing Protocols for Wireless Sensor Networks (WSNs)*. 10 2017. ISBN 978-953-51-3561-6. doi: 10.5772/intechopen.70208.

[110] R. C. Shah and J. M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, volume 1, pages 350–355 vol.1, Mar 2002. doi: 10.1109/WCNC.2002.993520.

[111] Faisal Karim Shaikh and Sherali Zeadally. Energy harvesting in wireless sensor networks: A comprehensive review. *Renewable and Sustainable Energy Reviews*, 55:1041–1054, 2016. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2015.11.010. URL `https://www.sciencedirect.com/science/article/pii/S1364032115012629`.

[112] Gurdip Singh and Kusuma Vellanki. A distributed protocol for constructing multicast trees. In *IN PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON PRINCIPLES OF DISTRIBURED SYSTEMS*, 1998.

[113] Gurdip Singh, Sanjoy Das, Shekhar V Gosavi, and Sandeep Pujar. Ant Colony Algorithms for Steiner Trees: An Application to Routing in Sensor Networks. In Leandro Nunes de Castro and Fernando J Von Zuben, editors, *Recent Developments in Biologically In-*

*spired Computing*, pages 181–206. IGI Global, Hershey, PA, USA, 2005. ISBN 9781591403128. doi: 10.4018/978-1-59140-312-8.ch008. URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59140-312-8.ch008`.

[114] Jessica Staddon, Dirk Balfanz, and Glenn Durfee. Efficient tracing of failed nodes in sensor networks. WSNA '02, page 122–130, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135890. doi: 10.1145/570738.570756. URL `https://doi.org/10.1145/570738.570756`.

[115] T. Stutzle and H. Hoos. Max-min ant system and local search for the traveling salesman problem. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pages 309–314, 1997. doi: 10.1109/ICEC.1997.592327.

[116] Thomas StÃŒtzle and Holger H. Hoos. Maxâmin ant system. *Future Generation Computer Systems*, 16(8):889 – 914, 2000. ISSN 0167-739X. doi: https://doi.org/10.1016/S0167-739X(00)00043-1. URL `http://www.sciencedirect.com/science/article/pii/S0167739X00000431`.

[117] S. Surendran and S. Prakash. An aco look-ahead approach to qos enabled fault- tolerant routing in manets. *China Communications*, 12(8):93–110, 2015. doi: 10.1109/CC.2015.7224693.

[118] A. T. Tai, K. S. Tso, and W. H. Sanders. Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *International Conference on Dependable Systems and Networks, 2004*, pages 805–814, 2004. doi: 10.1109/DSN.2004.1311951.

[119] Ankit Tiwari, Prasanna Ballal, and Frank L. Lewis. Energy-efficient wireless sensor network design and implementation for condition-based maintenance. *ACM Trans. Sen. Netw.*, 3(1):1–es, March 2007. ISSN 1550-4859. doi: 10.1145/1210669.1210670. URL `https://doi.org/10.1145/1210669.1210670`.

[120] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A macroscope in the redwoods. SenSys '05, page 51–63, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 159593054X. doi: 10.1145/1098918.1098925. URL `https://doi.org/10.1145/1098918.1098925`.

[121] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, page 28–39, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137079. doi: 10.1145/958491.958496. URL `https://doi.org/10.1145/958491.958496`.

[122] Hakim Weatherspoon and John D. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 328–337, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45748-0.

[123] Wenjing Lou. An efficient n-to-1 multipath routing protocol in wireless sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, pages 8 pp.–672, 2005. doi: 10.1109/MAHSS.2005.1542857.

[124] Wenjing Lou and Younggoo Kwon. H-spread: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 55(4):1320–1330, 2006. doi: 10.1109/TVT.2006.877707.

[125] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006. doi: 10.1109/MIC.2006.26.

[126] Huaming Wu, Yong Cheng, Qiuyue Liu, Jun Wang, Shaohua Wan, and Tariq Umer. Distributed fault detection for wireless sensor networks based on support vector regression. *Wireless Communications and Mobile Computing*, 2018:4349795, 2018. doi: 10.1155/2018/4349795. URL `https://doi.org/10.1155/2018/4349795`.

[127] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, page 70–84, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581134223. doi: 10.1145/381677.381685. URL `https://doi.org/10.1145/381677.381685`.

[128] Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298, 2005. doi: 10.1007/s11276-005-6612-9. URL `https://doi.org/10.1007/s11276-005-6612-9`.

[129] Peng-Yeng Yin, Ray-I. Chang, Chih-Chiang Chao, and Yen-Ting Chu. Niched ant colony optimization with colony guides for qos multicast routing. *Journal of Network and Computer Applications*, 40:61 – 72, 2014. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2013.08.003. URL `http://www.sciencedirect.com/science/article/pii/S1084804513001781`.

[130] M. Younis and R. Waknis. Connectivity restoration in wireless sensor networks using steiner tree approximations. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, 2010. doi: 10.1109/GLOCOM.2010.5683530.

[131] M. Yu, H. Mokhtar, and M. Merabti. Fault management in wireless sensor networks. *IEEE Wireless Communications*, 14(6):13–19, 2007. doi: 10.1109/MWC.2007.4407222.

[132] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. *UCLA Computer Science Department Technical Report*, 463, 10 2001.

[133] M. M. Zanjireh and H. Larijani. A survey on centralised and distributed clustering routing algorithms for wsns. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–6, 2015. doi: 10.1109/VTCSpring.2015.7145650.

[134] Ying Zhang, Lukas D. Kuhn, and Markus P. J. Fromherz. Improvements on ant routing for sensor networks. In Marco Dorigo, Mauro Birattari, Christian Blum, Luca Maria Gambardella, Francesco Mondada, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, pages 154–165, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-28646-2.

[135] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, page 1–13, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137079. doi: 10.1145/958491.958493. URL `https://doi.org/10.1145/958491.958493`.

[136] Y. J. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, volume 1, pages 356–362 vol.1, 2002. doi: 10.1109/WCNC.2002.993521.