THE UNIVERSITY OF WARWICK

Department of Computer Science

Third Year Project

Issues in Source Location Privacy for Wireless Sensor Networks

Author:
Matthew Bradbury

Supervisor:
Dr. Arshad Jhumka

Abstract

Wireless sensor networks are becoming increasingly prevalent and feature rich, enabling novel and diverse applications of them. When they are used to monitor valuable assets, it is possible for an attacker to find the location of these assets. Source location privacy provides a way to prevent the attacker from finding the source's location. This report builds upon previous work in the area by introducing a Template for implementing an algorithm to provide source location privacy for different network configurations not previously investigated. It also introduces an Adaptive algorithm that uses network knowledge to reduce energy usage and the number of parameters needed to give to the algorithm.

Keywords - Source Location Privacy; Wireless Sensor Network; Fake Source; Security;

26th April 2012

Acknowledgements

I would like to thank Arshad Jhumka for taking me on as his URSS student during the summer of 2011, introducing me to the fascinating problem of source location privacy and for all the support given during this project. I would also like to thank Matthew Leeke for helping me during my URSS project and for all the times I have pestered him for help since.

Those that also need a mention are those that helped me with the cluster: Simon Hammond for his initial assistance, and Oliver Perks for helping me with latter issues.

Contents

1 Introduction

1.1 What is a Wireless Sensor Network

A wireless sensor network is a collection of nodes that can communicate wirelessly with their neighbours and are used to sense their surrounding environment [15]. To communicate wirelessly each node is equipped with a radio that allows them to communicate with other nodes within a given range. They contain a very basic computing system to control the radio, handle messages being sent and received, process input from sensors and perform other calculations. As wireless sensor nodes are indeed wireless, it means they have no access to the mains power supply so they must run off batteries. This last point is hugely important as it means that each of the nodes must run off a finite energy source, which means that the software running on the node and the node's hardware needs to be designed with energy usage in mind [9].

While a limited energy source is the predominant characteristic of a wireless sensor node, there are numerous other traits or issues that can be considered. For instance it is possible for these nodes to be mobile (for example an ad-hoc network of PDAs) [24] which leads to very interesting behaviour in handling communication between these nodes. Nodes are usually equipped with the same features (such as the same wireless radio), need to withstand harsh outdoor conditions (as that is where they are typically deployed) [34, 39] and are used in an unattended fashion [21].

As energy usage is limited wireless sensor nodes tend not to use expensive broadcasting protocols such as IEEE 802.11 [15], but instead use much simpler alternatives to save energy. For example wireless protocols such as IEEE 802.15.4 ZigBee [10, 20] are designed to be used by wireless sensor networks and have a lower energy usage associated with them. Using these simple protocols unfortunately has the downside of meaning that broadcasts are subject to several types of collisions and message losses. So it is very important that the software running on the nodes is designed to handle these cases.

As wireless sensor nodes operate in harsh outdoors condition, there is a high probability of them failing. These faults can range from hardware damage caused by environmental conditions or tampering, software bugs, or simply a denial of service caused by nodes running out of power. So algorithms and software are often designed to handle these potential failures.

Wireless sensor networks can be used for a huge variety of tasks to solve many problems. They can be used to detect forest fires [14, 33], monitor pollution [4, 19] and anything that might require sensing over a large area. The only limit are the nodes, their ability to survive the environment and the availability of the required sensors.

1.2 The Problem of Source Location Privacy

The problem of source location privacy originates from privacy concerns within asset monitoring. Imagine there is a wireless sensor network which is monitoring valuable assets, any node that detects a valuable asset will broadcast information about that asset (known as a source). An attacker in the system can follow the messages being sent from the source and then through the network in order to capture that asset. In providing source location privacy the attacker is confused and thus doesn't capture the source. There are various ways to provide source location privacy and various types of attackers to consider when trying to provide it [5].

An attacker can follow a message from its current location to the source using a simple procedure. Any time the attacker is within range of a node broadcasting a message the attacker can find the direction that the message originated from. This is done by analysing the strength of the signal and then finding the direction in which the signal strength is strongest [8]. If the attacker keeps moving in the direction of the node sending a message it will eventually move into the range of another node that is broadcasting and repeat the procedure. By continuing to do this the attacker can travel between the nodes that are forwarding on the message from the source and eventually reach the source.

The problem is usually described in terms of the Panda-Hunter Game, which is as follows. A large array of panda-detecting sensor nodes have been deployed over a large area inhabited by pandas in order to monitor their position. When a panda is sensed by one of these nodes it will broadcast a message at a given rate. This message will be propagated through the network to a sink node. The game includes a hunter in the role of an attacker that is trying to capture pandas by back-tracing the path taken by the messages propagated from the message source. The aim is to introduce a privacy-cautious algorithm to prevent the hunter from reaching the source, while delivering the data to the sink. It is assumed that there is only a single panda and thus only a single source. We also assume that the source encrypts its messages such that an attacker can still receive them, but it is impossible for them to read the message's contents. [18, 25]

1.3 Related Work

Wireless Sensor Networks are currently a very hot topic of research, with many varying avenues of research being investigated. In this section the works that relate to the problem of source location privacy rather than wireless sensor networks as a whole will be covered. Many related works will also be introduced.

1.3.1 Ozturk *et al.*'s Seminal Paper

Source location privacy as a research topic took off with the seminal paper from Ozturk *et al.*. In the paper the authors first introduce the Panda-Hunter Game which was described in the previous section. They also classify the privacy threats, splitting them into two classes: context and content. Where content involves ensuring the message contents is not breached and context is where information about the network is not discovered. Source location privacy is a context type of privacy. Existing means of privacy in networks are discussed and deemed to be insufficient or irrelevant to the specifics of wireless sensor networks either because the methods are inapplicable to sensor networks, or they are solving a subtle different problem.

Initially baseline flooding is investigated as a means to propagate messages throughout the network, but also as a means to provide source location privacy. They found that flooding can never provide source location privacy, so other methods were investigated. Probabilistic flooding was found to improve on baseline flooding's energy usage and also to increase the time it took for an attacker to capture the source. In order to further increase the time it took for an attacker to capture the source the authors investigated two final techniques: fake sources and phantom flooding (which involved a random walk).

The fake source technique involved creating sources that broadcasted messages that were of the same length and also encrypted, so the attacker could not distinguish them from the real messages. They used 'naive injection' strategies to decide where to place the fake sources where a node became a temporary fake source with a given probability. They also investigated these temporary fake nodes becoming permanent fake nodes with a given probability. It was found that these techniques increased the time it took for an attacker to capture the real source.

The phantom routing technique involved the message taking a walk (by unicasting, not flooding) for a given number of hops, then flooding the network. This technique can increase the time it takes for the attacker to discover the source because each message will go on a different walk before flooding the network. However, it is not trivial to implement the random walk. It was also found the random walks can cancel each other out, so a directed random walk was used instead.

1.3.2 Kamat *et al.*'s Seminal Paper

Following Ozturk *et al.*'s paper was another seminal paper written by Kamat *et al.* which extended many of the ideas in the first paper covering baseline flooding, probabilistic flooding, flooding with fake messages and phantom flooding. They also introduce a model of a wireless sensor network and two possible ways of measuring the performance of a source location privacy algorithm. As part of their algorithm, the authors do not assume extra enabling hardware such as GPS so make do with the knowledge they can gain from the network. They do however, use a configuration phase where the sink broadcasts messages, which are then used by the rest of the network to set certain values.

1.3.3 Improving on the Seminal Work

Within the two pieces of seminal work the phantom routing algorithm was the preferred type of algorithm due to the lower cost and lower complexity compared to others such as probabilistic flooding and fake source flooding. The problem was that these papers make poor assumptions regarding when to create fake sources, how long they last and how often they broadcast. Jhumka *et al.* in [16] developed an algorithm that does not need to know the location of fake sources a priori. As part of this investigation they make the argument that there will always be a trade-off between privacy and energy. Both single and multiple attackers were investigated, with their algorithm achieving a less than 20% capture ratio.

Following that paper Jhumka *et al.* continued their investigations in [17]. Here the problem was formulated differently than Kamat *et al.*'s as it includes two new parameters: (i) message rates and (ii) fake source duration. Using this new model the authors were able to prove that the source location privacy problem is NP complete. They also proposed a heuristic based on the two aforementioned parameters to act as the algorithm to provide source location privacy. Overall their algorithm was able to achieve a less than 10% capture ratio with many instances reaching 5% or less.

1.3.4 Energy Usage

One of the most important aspects of wireless sensor networks is their energy consumption, every paper that related to wireless sensor network will most likely mention it as it is one of the defining characteristics of a wireless sensor network. Energy, or the lack of it, influences the software, the hardware and how the two interact. Whatever the wireless sensor network is doing the energy usage of that service or application will need to be taken into account. This is all because wireless sensor nodes run on a finite supply of energy.

Because energy is very important it is useful to be able to simulate how much power an algorithm will end up using. This was done by Shnayder *et al.* in [32] where they developed a simulation environment as an extension to TOSSIM which simulates TinyOS applications. Apart from this simulation environment the authors also measured the most energy intensive tasks, radio transmit was by far the most energy intensive task with radio listening the second most expensive. This is a very useful result, because instead of calculating how much energy is used, in many cases we can simply use the figures for messages sent and messages received.

1.3.5 Different Types of Attacks

There are numerous different types of attacks that wireless sensor network can undergo. Papers that have been mentioned previously mostly focus on local eavesdroppers when trying to provide source location privacy, but is is possible that the attacker may have some sort of global knowledge

[23].

There are also a whole host of physical attacks that a wireless sensor node can also undergo. The nodes could be disabled by having the batteries removed, their memory could be accessed to look for information such as cryptographic keys, debug hardware could be used to reprogram the device or sensors could be replaced with ones that produce faulty results. These are just a few things that an attacker could do to a node it has physical access to [5]. There are ways to mitigate the possible damage by: making the devices tamper proof, by only storing algorithm data in a subset of nodes and favouring symmetric key encryption.

When deploying in the outdoors where anyone may access your hardware it is incredibly important to consider these issues. One someone has access to your hardware they have access to a lot more knowledge available to them, making thwarting future attacks much more difficult.

1.3.6 Content Privacy

Source location privacy focuses on providing context privacy, where the context is the source's location. To be able to provide this context privacy, the algorithms developed require content privacy so the messages are unreadable by the attacker. This content privacy is incredibly important because the attacker may be able to gain information about the source's position from the contents of the message.

The problem of content privacy is well known with many techniques available to provide some level of privacy. At this point in time many methods of encryption are being employed in all manner of computing system to provide context privacy in computer networks [35, Ch. 8]. However, wireless sensor networks are in a different situation to many modern computers, in that they are energy limited and have low computational power. These two factors are incredibly important to consider because encryption is typically computationally expensive and can take millions of CPU cycles per byte needed to be encrypted [13, p. 207-208]. So encryption methods for wireless sensor networks must take these issues into account. [27]

1.3.7 General Traffic Analysis

The problem of source location privacy is a special case of subverting traffic analysis. As part of source location privacy we want to confuse the eavesdropping attacker to prevent them from finding the source's position. Circumventing general traffic analysis could be done to protect the sink instead of the source, or protect other important nodes. Deng *et al.* developed several schemes to prevent general traffic analysis. They noted that to provide this protection the number of messages that needed to be sent doubled or tripled, leading to a much higher energy cost, reinforcing the fact that energy is very important to consider in wireless sensor networks.

1.3.8 Energy Attacks

There are many other types of attacks that a wireless sensor network can undergo. One of them is the denial of sleep attack [7], where by broadcasting messages at certain times, can prevent the radio hardware in the wireless sensor node from going into its low power sleep state. As the radio is kept active, it means that the node runs out of power very quickly and leads to a denial of service when the node finally runs out of power altogether. This is a very different attack to source location privacy because the attacker is actually interfering with the network, whereas the attacker modelled in source location privacy is a simple observer.

1.3.9 Energy Scavenging

Because energy usage is so important there has been much work on attempting to scavenge energy from the surrounding environment [26, 29]. In taking energy from the environment, it is possible to increase the lifetime of the battery by recharging it, or reducing the power draw from it. In some cases it is even possible to eliminate the battery [38]. While this is not directly relevant to the problem of source location privacy, it is important that extra energy can be made available to mitigate the inevitable cost of providing source location privacy.

1.3.10 Real World Sensor Network Behaviour

It is incredibly useful to study wireless sensor networks in the safe environment of abstract problems and simulation, however, to ensure that the applications and protocols developed actually work it is necessary to test them on a real sensor network. Szewczyk *et al.*'s Great Duck Island and Werner-Allen *et al.*'s Reventador Volcano Monitoring deployments set up a wireless sensor network in the outdoors to monitor habitats and seismic activity respectively. They both focused on the hardware they were using but the main contributions of both papers were to analyse how reliable the sensor network were and what patterns of failure were observed. They both concluded that there is a very high likelihood of a sensor node failing and that the network must be able to handle the loss of nodes.

1.3.11 Conclusion of Related Work

Wireless sensor networks are currently a hot topic of research with a wide range of research being devoted to them. Because many problems are related to each other it makes doing work in one field very beneficial to those researching in a different field.A wide range of sources has been draw upon in an attempt to make the contribution as relevant as possible.

1.4 Contribution

There has been substantial work on developing algorithms to provide source location privacy, as shown previously. The first aim is to show that a common template exists for algorithms that use fake sources to provide source location privacy. An issue with previous work is that it only focused on grid networks with the sink and source in certain positions [16, 17], so this work will also investigate how to customise the template for networks in different configurations.

Secondly, previous works have mostly taken a random walk through the parameter space in attempt to find 'good' parameters which provide a low capture ratio and low energy usage. An adaptive algorithm will be developed that takes advantage of network knowledge to reduce the number of parameters a network administrator would need to set. As part of this algorithm another aim is to bring down the energy usage of the algorithm in comparison to the template results.

2 Model

The model used is based on the same model proposed by Kamat *et al.*, where the simulation game is a six-tuple $(\mathcal{N}, S, A, \mathcal{R}, \mathcal{H}, \mathcal{M})$, where

$\mathcal{N} = \{n_j\}_{j \in I}$ This is the network of indexed sensor nodes n_j, which are indexed using an index set $I \subseteq \mathbb{N}$. j is the node's unique identifier.

$S \in \mathcal{N}$ is the network sink, to which all communication is ultimately routed to.

$A \in \mathcal{N}$ is a sensor node that has detected an asset. Assets are characterised by a mobility pattern M_o.

$\mathcal{R}$ is the routing policy employed to protect the asset from being acquired or tracked by the hunter.

$\mathcal{H}$ is the hunter, or attacker, who seeks to acquire or capture A through a set of movement rules $\mathcal{M}$.

The following sections will detail the configuration of the simulator used, the attacker's behaviour ($\mathcal{H}$ and $\mathcal{M}$) and how distributed algorithms will be defined in this report. Defining the structure of $\mathcal{N}$ and the positions of S and A will be left until section 3. The definitions of the different algorithms that implement $\mathcal{R}$ will be presented in section 3 and section 4.

2.1 Simulating Wireless Sensor Networks

To ensure that the algorithms developed work under the situations defined they will need to be tested. While one may think that running the algorithms on an actual wireless sensor network would be best, in this case it is not. A real network has the downside of requiring considerations of other issues (such as hardware failures) that are not being considered. It also requires a real person to act as the attacker and takes real time to run the algorithm. Using a simulator provides an automated way to test and repeat simulations, which allows a faster turnaround of results with less effort. For this project JProwler [2] was used to simulate the algorithms developed.

2.2 Simulation Radio Model

JProwler has two radio models that can be used: Rayleigh and Gaussian. The Rayleigh radio model is designed to be used when nodes are moving very often and the Gaussian model is for the case when nodes do not move as often. It has been very important to decide which one to use, because the majority of our network is stationary, but one node (the attacker) will be very mobile. As one our our nodes is very mobile, the Rayleigh radio model will be used for every node. The radio models have not been mixed and matched because the different models have different signal strengths and ranges, and as part of the attacker model it will be assumed that the attacker has the same abilities as the rest of the nodes in the wireless sensor network. So the model used by every node must be the same.

2.3 JProwler Configuration

Each wireless sensor node can be configured to have different protocol timings, our simulator (JProwler) simulates the Mica2 platform and its parameters. The following are the constants (taken from the source code) used to determine how long it takes to send a message. Values such as noise and signal are calculated during runtime by the simulator. [2]

Send Minimum Waiting Time: 5 ms This is the minimum amount of time that a node will wait for after being asked to send a message.

Send Random Waiting Time: 3.2 ms This is a random amount of time that will be added to the minimum wait time to decide how long to wait before sending a message. The minimum amount is 0, the maximum is the time given.

Send Minimum Back Off Time: 2.5 ms This is the minimum amount of time that a node will back off for when another message is sensed as being broadcasted.

Send Random Back Off Time: 0.75 ms This is a random amount of time that will be added to the minimum back off time to decide how long to wait before sending a message. The minimum amount is 0, the maximum is the time given.

Send Transmission Time: 24 ms This is the amount of time it takes to send a message. It is constant, no matter the length of the message.

This means that without sensing the channel for a message being broadcasted by neighbouring nodes, it will take a node between 24.5 and 24.82 ms to broadcast a message.

2.4 JProwler Communication Simulation

When simulating the Mica2 platform JProwler allows messages to be broadcasted in two ways that are useful. The first is to simply send the message. This is useful when the information just needs to be sent and the algorithm doesn't care about avoiding collisions. The second is to first sense the channel and detect if any neighbours are sending message, the node then waits until the channel is clear and sends the message. This case is useful when trying to avoid collisions, but not when a large volume of messages is trying to be sent very quickly.

There is a problem in that even when sensing the channel collisions can occur because the protocol does not implement any methods to detect the hidden terminal problem. This problem occurs when a node wants to broadcast and a node two hops away is currently broadcasting, when the first node starts broadcasting the middle node will not receive a message as the signals from the two surrounding nodes have collided [6]. To prevent this from happening it would require a more complicated protocol and more messages to discover who is broadcasting and avoid the collisions (these exist many of these protocols, [28] is one example), so it is not implemented in JProwler or the Mica2 platform. This means that collisions are impossible to avoid when simulating networks using JProwler.

2.5 Communication Protocol

There are various kinds of routing protocols that can be used in wireless sensor networks [3]. The flooding protocol which is used is outlined as follows:

1. A source node will generate and broadcast a message.
2. The neighbours will receive this message.
3. If the receiving node has not previously seen the message then the message is rebroadcasted.

It has been shown by Ozturk *et al.* in [25] that "flooding provides the least possible privacy protection since it allows the adversary to track and reach the source location within the minimum safety period". So, the algorithms are developed on top of flooding to improve on this worst case level of privacy protection.

2.6 Defining Distributed Algorithms

To define the algorithms developed, a custom pseudo-language is used. Each box represents a process running on a wireless sensor node (although it is possible for one of these boxes to be split up to represent the algorithm better). Within this box there are three sections **variables**, **constants** and **actions**. The **variables** section details modifiable variables that exist within the program. The **constants** section details network knowledge or network parameters that cannot be changed by the program, these constants will be the same for all nodes that have them.

The **actions** section contains the methods which depending on the type will be called when a message is received or after a timer has timed out. The first type of function (receiving a message) is demonstrated in Figure 1, where '*MessageContents...*' is the list of variables contained within the message.

```
process $j$
**actions**
    % Receiving Message
    *RcvFunction*:: **rcv**$\langle MessageContents... \rangle \rightarrow$
        % Function Contents
```

Figure 1: Example Receive Message Algorithm

The second type of function (timer timeout) is demonstrated in Figure 2. The function **set** is used to restart the timer, so it is called again once the timer times out again. If **set** is not called then this function will not be called again.

```
process $j$
**variables**
    *period*: timer init $\alpha$;

**constants**
    % How often the message is broadcasted
    $\alpha$: time;

**actions**
    *SendFunction*:: **timeout**(*period*) $\rightarrow$
        % Function Contents
        **set**(*rate*, $\alpha$);
```

Figure 2: Example Send Message Algorithm

Variables and constants are given types and can optionally be initialised to a given constant. They can be initialised with $\perp$ to indicate that their value has not been set. Also anything after a '%' is considered a comment.

```
        Normal broadcast
        **BCAST**$\langle Message \rangle$;

        Delayed Broadcast - The broadcast occurs after $T$ time units
        **BCAST**$\langle Message \rangle$ **in** $T$;

        Sense Broadcast - The broadcast occurs when the node cannot detect a
                          message being sent by its 1-hop neighbours
        **BCAST**$\langle Message \rangle$ **sense**;

        Repeated Broadcast - The message is broadcasted N times
        **BCAST**$\langle Message \rangle$ **repeat**(N);
```

Figure 3: Types of Message Broadcast

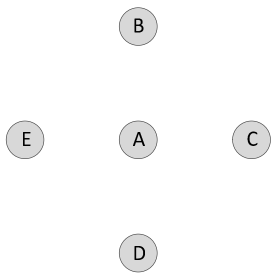
To send messages we use **BCAST**, of which there are several varieties required as shown in Figure 3. These can also be combined, for instance it is possible to sense the channel for messages and repeat the broadcast N times (sensing each time) using '**BCAST**$\langle Message \rangle$ **sense repeat**(N)'.

2.6.1 Implicit Steps in BCAST

In the algorithms defined a set of hashes is maintained so messages received can be checked against it to ensure that messages are not processed twice. However, there is an implicit part of these algorithms that is not shown. This is that every time a node generates and broadcasts a message that hash of that message is also added to the set of messages seen. This is important to do because it prevents the following behaviour which generates more messages and leads to more collisions.

On the left there is a very small wireless sensor network where each of the outer nodes can communicate with A and vice-vera, but no other nodes are within communication range of each other. The table on the right shows the number of steps taken when A broadcasts a single message. With the column on the left being the case when a broadcast doesn't record the generated message's hash and the column on the right doing so.

What can be seen is that not recording the hash means that the node generating the message has a chance to receive the message and thus process the message it generated. This is not needed because A already has knowledge of the message, so by not recording the hash it leads to increased energy usage by performing extra steps.



	Without Recording Hash	With Recording Hash
1	A BCASTs	A BCASTs
2	B, C, D, E Receive	B, C, D, E Receive
3	B BCASTs	B BCASTs
4	A Receives	A Ignores
5	A BCASTs	
6	B, C, D, E Ignores	

From here on every time an algorithm contains a **BCAST** it will use this behaviour of putting the message's hash it is sending into the *messages* set.

2.7 Message Encryption and Fake Message Format

All messages sent by the source node are assumed to be encrypted. This ensures that the attacker cannot read the messages to gain an insight into where the source by be by violating content privacy. This can be done through various mechanisms that are out of the scope of this project and were covered in the related work.

To ensure that the attacker cannot distinguish between fake and real messages it is assumed that the fake messages are encrypted and that they are the same length as real messages. These two properties should mean that fake and real messages are indistinguishable to each other, so the attacker should not be able to work out which is real and which is fake.

2.8 Asset Mobility Pattern

As part of the formal definition the asset detecting node A need to have a mobility pattern M_o defined. For these algorithms it is assumed that the asset will never move, and thus never change the asset-detecting-node that is generating messages. To be able to handle movement it is intended for the algorithms to simply be restarted every time it has been detected that A changed. This can be done by recording A's unique identifier j and then restarting the algorithm when the current identifier does not equal the previous identifier stored.

2.9 Attacker Model

So far the nature of the model of the network has been detailed. In this section, the model of the attacker $\mathcal{H}$ and how it behaves $\mathcal{M}$ is covered.

2.9.1 Attacker Complexity

There are many ways that an attacker can approach the problem they are trying to solve, that is finding the location of the source of messages being sent through the network. For example, one attacker could use a brute force search to solve the problem in linear time (with respect to the number of nodes in the system). The next level up would be to have that node contain memory, and use it to evaluate what direction would be best to explore in. We can complicate this more by having multiple coordinated attackers.

To simplify the problem, the attacker will be implemented as the simplest intelligent agent (as set out by Russell and Norvig), which is a simple reflex agent. This will allow a focus on developing the best way to provide source location privacy for this initial case, without the development being complicated by trying to thwart more complicated behaviour.

2.9.2 Attacker Ability

When considering the attacker in the system, one must also understand what it is actually able to do. Benenson *et al.* in [5] makes the point that *presence* and *intervention* are the two most relevant parameters for basic security analysis. So they will be focused upon define the attacker's ability.

Presence defines where the attacker acts in the system, whether it is *local*, *distributed* or *global*. Local means that the attacker is only able to act on a small subsection of the network, probably only equipped with a radio capable of broadcasting several metres. A distributed attacker is similar to a local attacker except that it is either mobile, or has access to many small subsections of the network. A global attacker has access to the entire network. Here we have a total order of ability, where a global attacker is the most powerful, then a distributed attacker and finally a local attacker is the least powerful.

Intervention defines what the attacker can do within the network, this ranges from simply *eavesdropping* to *crashing* nodes and finally to *reprogramming* them. Eavesdropping means the attacker only listens to messages, no messages will be sent. Crashing means that the attacker causes a node to crash, this could be achieved by a physical or software based attack. Finally reprogramming means that the attacker can access the node and run arbitrary programs on that node. Once again there is a total order of power with reprogramming being the most powerful, then crashing and finally an eavesdropping attacker is the least powerful. These are only a small number of the many vectors an attacker can take to intervene in the network.

2.9.3 Attacker Properties

The attacker in this model is essentially a mobile wireless sensor node. It has the same capabilities as any other wireless sensor node in the network, such as radio range, except it also has the ability to move. Assuming the radio range is the same for the attacker as every other node in the network is done for simplicities sake. The attacker has the following properties:

Distributed Presence The attacker has a limited listening range, so can only listen in on a few surrounding nodes. To simplify the simulation we assume that it has the same radio range as the rest of the nodes in the network.

Mobile The attacker is mobile. This is because it needs to move through the network to find the source.

Eavesdropper The attacker only listens for messages, it does not broadcast under any circumstance. This means that our network cannot know where the attacker is and then react to that knowledge.

Direction Sensing The attacker knows from what direction messages have come from.

Aphasic This means that the attacker cannot understand the content of messages. It is however, able to record the contents and tell if a message had been previously received.

2.9.4 Attacker Algorithm

```
process $j$
variables
    % Messages seen
    $messages$: set of int init $\emptyset$

    % Has the source been found
    $sourceFound$: boolean init False

actions
    % Receiving Message
    $receive$:: $\mathbf{rcv}\langle Message, hash\rangle \rightarrow$
        if $(\neg sourceFound \wedge hash \notin messages)$ then
            $messages := messages \cup \{hash\}$;
            Move(PositionOfSender(Message));
            if (OurPosition = SourcePosition) then
                $sourceFound := True$;
            fi;
        fi;
```

Figure 4: Attacker Algorithm

This algorithm details an attacker's movement rules $\mathcal{M}$, whereby it will move 1-hop to a neighbouring node that sent a message the attacker has not previously received. An assumption is made that the attacker can tell when it is receiving a message that it has received before, but that content privacy of the message is ensured. Another part of this algorithm is to detect and record that the attacker has found the source node (A).

2.9.5 An Issue with Attacker Behaviour

One of the big issues with source location privacy is that from the attacker's viewpoint the problem they are trying to solve is a simple search problem: given a set of sensor nodes which one has detected the valuable asset? This can simply be solved with a brute force search where the attacker visits every node. To work around this it is assumed that the attacker doesn't perform this kind of behaviour and is trying to do something clever by listening in on network traffic and only using that as the means to find the valuable asset.

2.9.6 Starting Position

Being in line with previous work [17, 16, 18, 25, 40], the attacker will start at the same position as the sink. There has been work with the attacker starting at a random position [31], but a fixed starting position has been chosen for a number of reasons. The first is that in all simulations it is know where the attacker will start, so firmer conclusions can be made about the results. The second is that the sink is a reasonable place for the attacker to start as it is most likely a well known base station and it is unlikely to be disconnected from the rest of the network. Finally, it gives a position that is not to far, but not too close to the source. For example if the attacker was randomly place one hop from the source many times, there is little the algorithm can do to prevent the attacker from finding the source. Placing it at the sink is a reasonable compromise against the potential randomness of the attacker's position and likely distance from the source.

2.10 Testing

Testing algorithms developed for wireless sensor networks is very difficult, due to the inherent non-determinism [12] in the wireless sensor network. One cannot just test if a certain set of events happen, because there are many sets of events that lead to a desirable outcome. For example due to collisions the time at which a node becomes a fake source might change, but as long as the capture ratio is low and the energy usage is low then the algorithm is 'good'. However there is another problem here, that we are trying to minimise both energy usage and capture ratio and these metrics do not often decrease with each other. Usually it takes an increase in energy to provide a lower capture ratio and vice versa. So defining what is correct behaviour entirely depends on what the user of the wireless sensor network wants to optimise more, capture ratio or energy usage.

Because testing is very difficult, the algorithms developed were tested by actually running them and gathering results. If the capture ratio is low and the energy usage is also relatively low then the algorithm will be deemed as acceptably good. This is also problematic because deciding what 'low' capture ratio is good enough is subjective, adding energy usage and message receive ratio of the sink also makes it more complicated to decide what results are good. So, there will be focus on obtaining a low capture ratio for the Template algorithm, and then a focus on reducing the energy usage in the Adaptive algorithm. If the capture ratio is near 0% and the Adaptive algorithm has a lower energy usage than the Template algorithm then the algorithms will be considered good.

3 Template Algorithm

3.1 Introduction

The aim of this project is to investigate different configurations of networks and develop an algorithm to provide source location privacy for these configurations. In previous work, there has been a large focus on grid networks, so as part of this project configurations other than grids will be investigated. The overriding aim of this part of the project is to come up with a new method to allocate fake nodes at run time that will be effective on different networks.

3.2 Network Setup

The configuration used to set up the network varies depending on the structure of the network used, be it grid, ring or circle. All of them are two dimensional networks of varying size. All of their nodes are the same distance away from their North, South, West and East neighbours (expect those on the edge of the network that lack those neighbours). Each network was simulated at different sizes $\Sigma \in \{11, 15, 21, 25\}$, these sizes lead to different sizes for different network structures.

For grid networks it means that there were Σ^2 nodes in a grid shape, so the sizes $\{11, 15, 21, 25\}$ respectively had $\{121, 225, 441, 625\}$ nodes in the network. Ring networks were created by first creating a grid network then removing all nodes that were not along the edge of the network. Circle networks were created by first creating a grid network then removing all nodes that were not within the circle that had its centre point at the centre node and a radius of $\frac{\Sigma}{2}$.

Nodes are initially assigned unique sequential node identifiers which are used to determine where to place the source and the sink. Once these nodes have been placed the node identifiers are randomized, so that every node still have a unique identifier, but its identifier has no relation to the identifiers of the node's neighbours.

3.3 Configurations

The following configurations are those that have been identified because they should be interesting and have different issues associated with them. These are only a small subset of possible configurations, as it would be infeasible to test them all, this is why the following representative subset of configurations have been chosen.

3.3.1 Source Corner

The source in the top left corner of the grid network and the sink is at the dead centre. This configuration is important because the source is at a point that is furthest from the sink and there are nodes that are further from the source than the sink is. This configuration also has the useful property of having a node that is furthest from the source at a point whose distance from the source is the same as the source's distance from the sink, this can be capitalised easily by the algorithm.

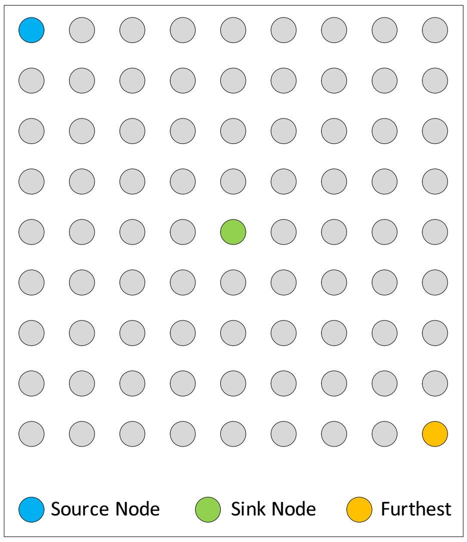


Figure 5: Source Corner for a size 9 grid network

The distance between the sink and the source is equal to the size of the network minus 1 $(\Sigma - 1)$.

This configuration could be rotated such that the source node is in any of the other corners. Given this rotation the algorithm should behave in a similar fashion to the example given.

3.3.2 Sink Corner

The sink in the bottom right corner of the grid network and the source is at the centre. This configuration is important because the sink is at a point that is furthest from the source. There is nowhere further from the source than the sink is. Meaning that in order to lure the attacker to one of the furthest nodes it will have to lure it closer to the source first. This must be done with care so the algorithm doesn't just lead the attacker to the source.

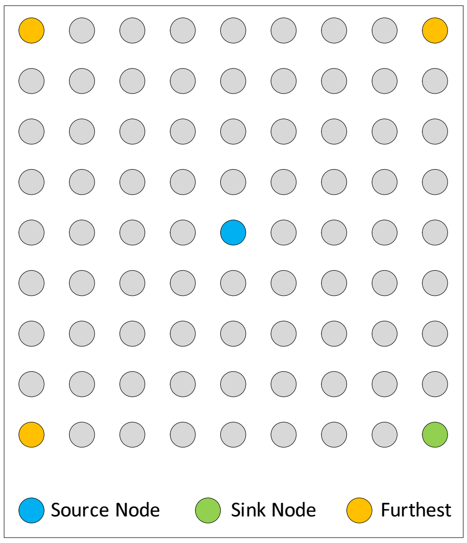


Figure 6: Sink Corner for a size 9 grid network

The distance between the sink and the source is equal to the size of the network minus 1 $(\Sigma - 1)$.

Once again this configuration could be rotated so that the sink is in any of the corners while the source remains in the centre. Again the behaviour when rotated should be similar to that of this original configuration.

3.3.3 Further Sink Corner

The sink is in the bottom right corner of the grid network and the source is 3 hops to the east and 3 hops south of the top left corner. This configuration is important because the sink is at a point that is furthest from the source. This is different from Sink Corner, because instead of working out how to escape from the corner, fake sources have to lead the attacker away, but not in the direction of the source. The other problem is that the sink node is the node that is furthest from the source, the algorithm will need to handle this as the sink node cannot be a fake source as well.

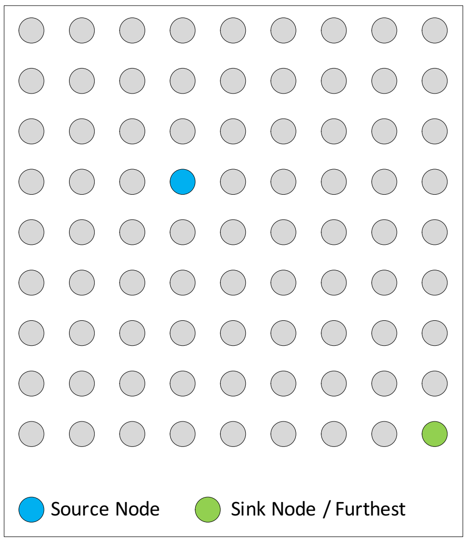


Figure 7: Further Sink Corner for a size 9 grid network

The distance between the sink and the source is equal to $2\Sigma - 8$.

As with the previous configurations rotation should not affect the running of the algorithm.

3.3.4 Generic 1

In this configuration the sink is very close to both the edge of the network and the source. Fake sources must be created diagonally behind the sink to lure the attacker away. Using the same algorithm as in previous papers [17] fake sources would be created closer to the source. The algorithm designed should be use a different methodology to avoid this problem. This configuration is interesting because the sink and source are very close, the sink is close to the edge and the furthest points are at steep diagonals to the sink.

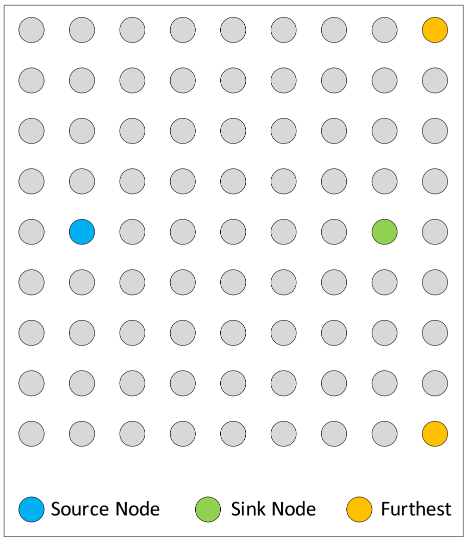


Figure 8: Generic 1 for a size 9 grid network

The distance between the sink and the source is equal to $2\lfloor \frac{\Sigma}{3} \rfloor$.

As with the previous configurations rotation should not affect the running of the algorithm.

3.3.5 Generic 2

This configuration is similar to Generic 1, but is on a diagonal slant rather than a horizontal. This puts more distance between the source and sink, but also changes the positions of where the furthest node is. This is an interesting configuration because there is very little space behind the sink to create fake sources. So the algorithm should be able to allocate in this space, or in a space that is further from the sink, but not too close to the source.

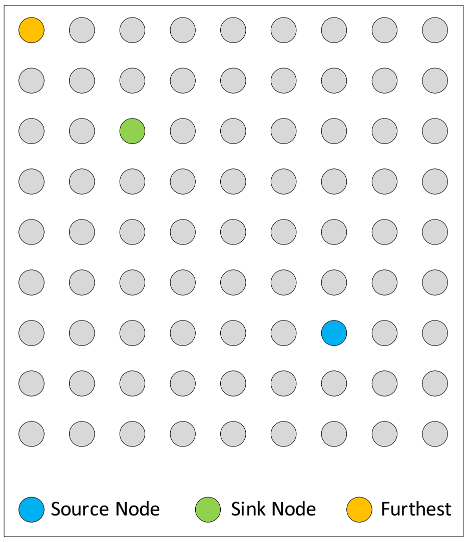


Figure 9: Generic 2 for a size 9 grid network

The distance between the sink and the source is equal to $2\Sigma - 10$.

As with the previous configurations rotation should not affect the running of the algorithm.

3.3.6 Ring Top

While ring shaped networks are perhaps not the most realistic of network types that will be deployed in real life scenarios, they represent an interesting basic network shape to study. In any configuration there are only every two paths from the source to the sink. So it will be important to see what effect fake messages have on the number of messages received by the sink.

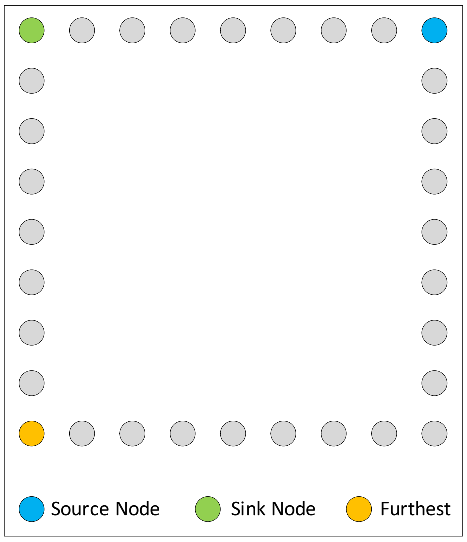


Figure 10: Ring Top for a size 9 ring network

In this configuration the sink is in the top left corner and the source in the top right. This configuration is interesting because there is space behind the sink for a fake source to be created at a node that is further from the source than the sink is.

The distance between the sink and the source is equal to $\Sigma - 1$.

3.3.7 Ring Middle

For the Ring Middle configuration the sink and source are in the middle on opposite sides of the network. In this configuration the sink node is also the furthest node from the source, so like the Further Sink Corner no matter what direction fake sources are created in they will be created in the direction of the source. Here it will be very important to see where permanent fake sources are created to ensure that neither path to the source is blocked, making sure the receive rate of the sink doesn't decrease by too much.

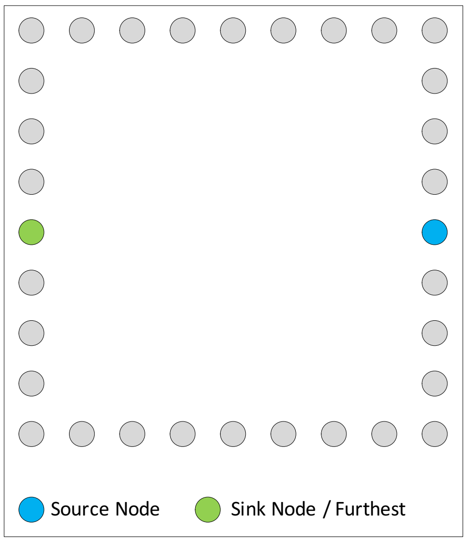


Figure 11: Ring Middle for a size 9 ring network

The distance between the sink and the source is equal to $2\Sigma - 2$.

3.3.8 Ring Opposite

For the Ring Opposite configuration the sink and source are in opposite corners. This configuration should be equivalent to the Ring Middle configuration, but is being analysed to see if behaviour is any different. As the configuration is essentially the same the sink is also the furthest point from the source. This configuration also needs to consider if one of its paths will be blocked by fake sources.

The distance between the sink and the source is equal to $2\Sigma - 2$.

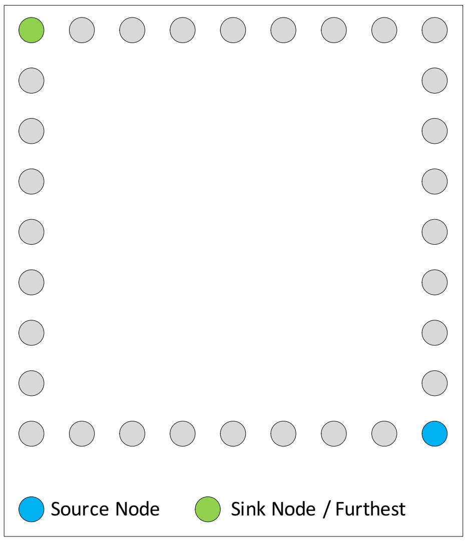


Figure 12: Ring Opposite for a size 9 ring network

3.3.9 Circle Edges

This configuration involves a circle, which should produce behaviour than that of a ring or a grid. The property that each node is the same distance from its surrounding nodes is maintained. Meaning that not all nodes on the edge are the same distance from the centre, so this is not a true circle.

The sink and source are on opposite edges of the network. The sink node is at one of the points furthest from the source.

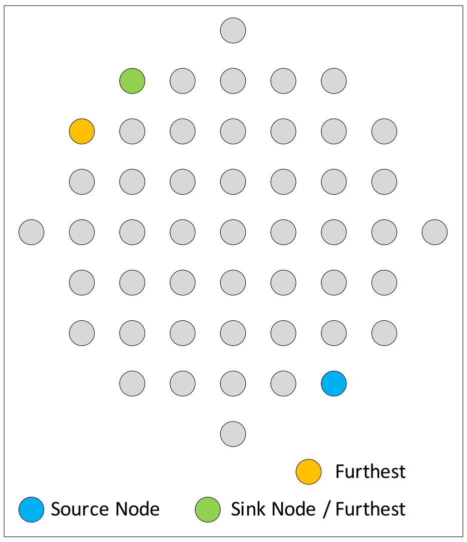


Figure 13: Circle Edges for a size 9 circle network

3.3.10 Circle Sink Centre

In this configuration the sink is in the centre of the circle and the source is on the edge. There exists space behind the sink for fake sources to be created in. This configuration is expected to behave similarly to the Source Corner configuration.

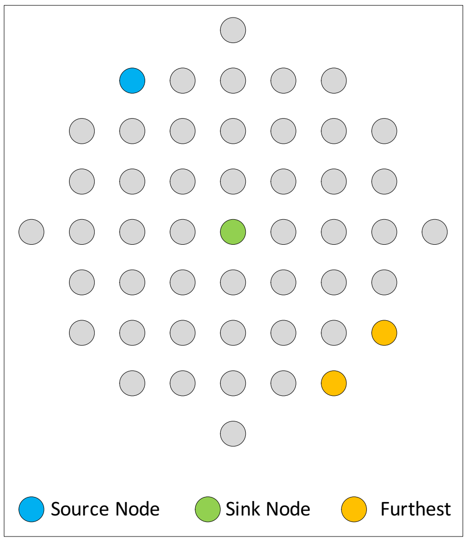


Figure 14: Circle Sink Centre for a size 9 circle network

3.3.11 Circle Source Centre

This configuration has the source node in the centre and the sink on the edge of the network. The sink is at a point furthest from the source, but there are also other points furthest from the source on the edge of the network. This configuration is expected to behave like the Sink Corner configuration.

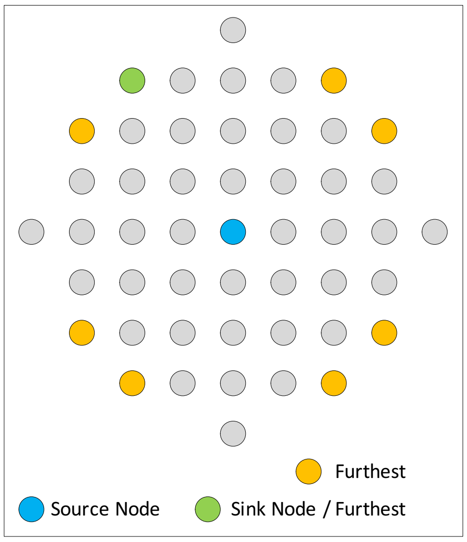


Figure 15: Circle Source Centre for a size 9 circle network

3.4 Safety Period

3.4.1 Calculating

In the problem of source location privacy, one thing that is difficult to define is the notion of when the attacker has been defeated. Therefore, a safety period (as defined in [16]) will be used. The safety period is defined such that an attacker has failed to capture the source once the time in the simulation reaches twice the amount of time it takes for the attacker to capture the source when there is no protection.

3.4.2 Algorithm to Calculate Safety Periods

The algorithm running on the nodes when there is no protection is defined below. Simply the source node creates and sends messages, normal nodes forward the messages onwards and the sink simply listens for the messages. The attacker follows the algorithm given in Figure 4.

```
process $j$ - If type is Normal
variables
    % Messages seen
    $messages$: set of int init $\emptyset$

actions
    % Receiving Normal Message
    $receiveNormal$:: $\mathbf{rcv}\langle Normal, hash\rangle \rightarrow$
        if ($hash \notin messages$) then
            $messages := messages \cup \{hash\}$;
            $\mathbf{BCAST}\langle Normal, hash\rangle$;
        fi;
```

Figure 16: No Protection Algorithm - Normal Node

```
process $j$ - If type is Source
variables
    % The timer used to send message
    $period$: timer init $P_{source}$;

constants
    % How often messages are sent
    $P_{source}$: time;

actions
    % Sending Normal Messages
    $sendNormal$:: $\mathbf{timeout}(period) \rightarrow$
        $\mathbf{BCAST}\langle Normal, \mathbf{hash}(Normal)\rangle$;
        $\mathbf{set}(period,\ P_{source})$;
```

Figure 17: No Protection Algorithm - Source Node

```
process $j$ - If type is Sink
**variables**
    % Messages seen
    *messages*: set of int init $\emptyset$

**actions**
    % Receiving ⟨NORMAL⟩ Message
    *receiveNormal*:: $\mathbf{rcv}\langle Normal, hash \rangle \rightarrow$
        **if** ($hash \notin messages$) **then**
            $messages := messages \cup \{hash\}$;
            **ProcessMessage**($Normal$);
        **fi**;
```

Figure 18: No Protection Algorithm - Sink Node

When gathering the results the safety period algorithm was run for each of parameter combinations, not related to creating fake sources, that will be used in the Template algorithm. Previous work used the slowest message rate (1 message per second) to get the longest safety period [16], however, this does not take faster source rates into account. So when these safety periods were calculated they were run for each combination of network size, network configuration and source rate. The simulations were repeated 10,000 times and the average of the time taken to capture the source was recorded. To calculate the safety period for a given configuration, size and rate, the time it took for the attacker to find the source was doubled.

3.4.3 Selected Results

The following are the safety periods for two configurations, the rest can be found in Appendix A along with greater detail about the network.

Table 1: Safety Periods for the Source Corner configuration on grid networks

Network Size	**Safety Period**			
	1/sec	**2/sec**	**4/sec**	**8/sec**
11×11	33.58	16.90	8.99	9.41
15×15	49.63	24.85	13.29	14.47
21×21	73.52	36.74	19.78	22.90
25×25	89.80	44.68	24.34	28.52

Table 2: Safety Periods for the Sink Corner configuration on grid networks

Network Size	**Safety Period**			
	1/sec	**2/sec**	**4/sec**	**8/sec**
11×11	31.53	15.87	8.29	9.16
15×15	47.29	23.91	12.39	14.92
21×21	71.00	35.69	18.85	24.65
25×25	87.08	43.73	23.19	31.87

3.4.4 Safety Period Analysis - Sink-Source Distance Discrepancies

It is interesting to see that different values for the Source Corner (Table 6) and Sink Corner (Table 7) configurations were obtained, even though the sink-source distance is the same. What appears to be happening is that it is easier for an attacker to get from the corner to the centre rather than the other way round. This is likely to be due to collisions occurring in a different pattern.

3.4.5 Safety Period Analysis - Receive Ratio

The receive ratios seen show that with just basic flooding perfect message transmission is not obtained. However, this was to be expected as energy expensive protocols to ensure the sink receives every message are not used. These results also match the findings of Werner-Allen *et al.* and Szewczyk *et al.* who deployed wireless sensor networks in real world scenarios and found that their wireless sensor networks didn't see a 100% receive ratio.

Some configurations have better receive ratios than others, this is also to be expected. The different paths that messages will need to take from source to sink will lead to different patterns of collisions, which will lead to different receive ratios.

3.4.6 Safety Period Analysis - Increasing Safety Period

For grid networks there was some unexpected behaviour. Between the messages rates 4 and 8 per second the safety period increases. This is probably because more messages are being sent, leading to higher collisions, so fewer messages reach the attacker to lure it towards the source. Or due to collisions they reach the attacker from the opposite direction and lead it away from the source. This is backed up by the fact there is a drop in the receive ratio over this same period. Figure 19 shows this pattern for size 25 Sink Corner networks, but from the safety period tables (Tables 6, 7, 8, 9, 10) this pattern is also observed for the rest of the grid configurations.

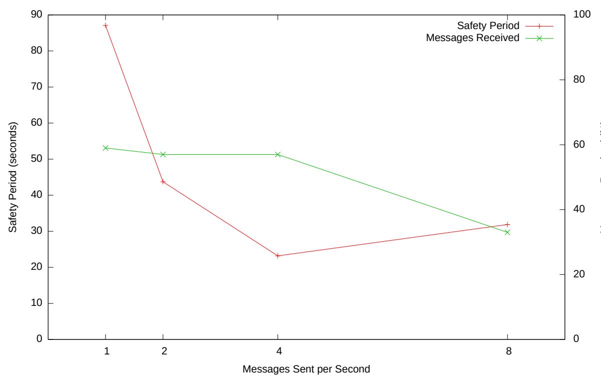


Figure 19: Safety Period and Receive Ratio for a size 25 Sink Corner network

3.5 Method

3.5.1 Sink as Fake Source

When considering how to implement this algorithm, it seemed that it would be easiest to simply have the sink act as the fake source. The idea is that the attacker starts at this position so it should require a minimal amount of energy to keep it there and away from the real source. However, this method is not acceptable for several reasons. Primarily because this means that the algorithm becomes more dependant on the attacker starting at the sink. Secondly, because this has not occurred existing literature.

3.5.2 Pre-configuration Phase

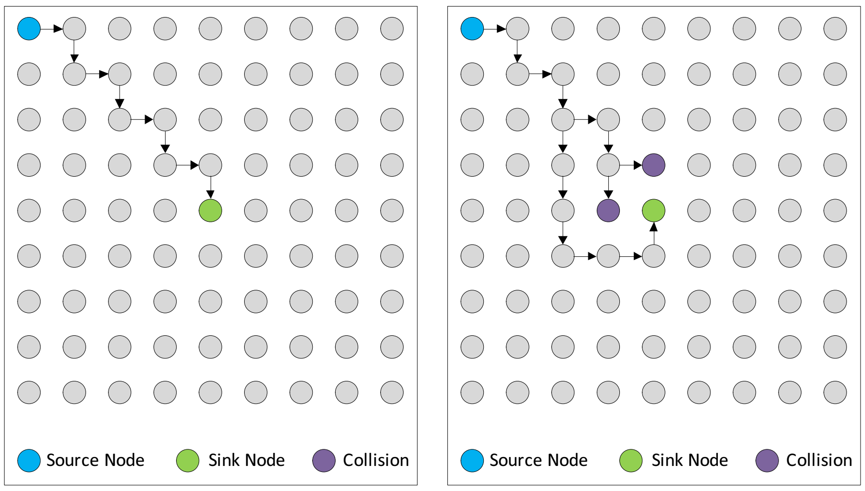
Because the sink cannot act as the fake source, it means there is a need to be able to calculate which nodes in the network should become fake sources. Using a pre-configuration phase to set this up was incredibly tempting, because it allowed better calculation of values due to two issues when calculating during runtime of the algorithm. First, certain sections of the network are unlikely to be able to communicate with others, for example, a message forwarded by one corner is unlikely to be received by the opposite corner. This is because nodes between the opposite corner nodes are likely to have already forwarded this message and the algorithm specifies that they do not forward the same message twice. Secondly, it allowed better calculation of values that may be different than expected or absent due to collisions. Examples of this is the sink-source distance being calculated as greater than it is due to collisions.

However, this calculation phase is undesirable because it adds extra burden on the network administrator and means that algorithm is less able to handle changes such as nodes moving. It is also one of the criticisms of previous papers [16], that nodes were preselected to be fake sources. So the algorithms are designed to be run on a network without a pre-configuration phase to set up network values. In doing so the certain knowledge that the pre-configuration phase offered was sacrificed and the algorithms were designed to handle the potentially unreliable knowledge of the network.

3.5.3 Collision Difficulties

In wireless sensor networks, it is often the case that simple protocols such as CSMA are used instead of more reliable protocols such as IEEE 802.11 [15]. Using a simpler protocol it means that energy is saved, however, the simple protocol cannot avoid message collisions in all cases. Using this simpler protocol that cannot prevent all collisions leads to many difficulties.

For example, one of the most fundamental pieces of information that need to be set up in the wireless sensor network is the distance of a node from the source. Due to collisions it is entirely possible (and quite probable) that this value will not be the minimum source distance. Figure 20 illustrates how collisions could cause the path of a message to change from having a length of 8 to a length of 10, this then changes the source distance that would be calculated. This problem is compounded further because of the large number of messages that will be sending and receiving in the system at a given time, leading to a high number of collisions.



(a) A path with no collisions (b) A path with two collision

Figure 20: Different paths from a source to a sink due to collisions

So with all algorithms developed it was necessary to ensure that the algorithm is resilient to the problems that are caused by collisions, by making the algorithm able to handle unreliable information.

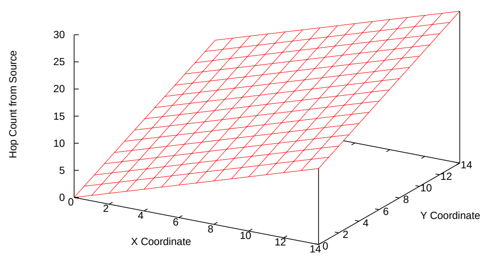
3.5.4 Aims

One of the overall aims when designing the algorithm was to have the permanent fake sources created at the point in the wireless sensor network that was the furthest away from the source and have temporary fake sources created in the in-between space. This is desirable because it leads the attacker a long way from the real source and also because those nodes should be plausible. Whereas alternative methods such as allocating the permanent fake nodes adjacent to the sink as fake sources may use less energy, but do not draw the attacker very far and their proximity to the sink may reduce the number of messages it receives.

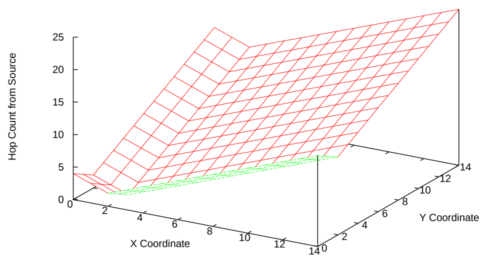
The permanent fake source allocation is a different method to previous papers because this algorithm is intended to work well on any configuration. Previous papers such as [17] created permanent fake sources at nodes that were the same sink-source distance from the source. This was an acceptable decision because they authors were only working on grid networks and the sink and source were positioned such that at this distance a permanent fake source could be created at one of my target points. For the algorithm its use is not desirable because fewer assumptions are made about the network and its configuration. So their method cannot be to detect the position of fake sources.

Detecting these furthest points is a difficult problem, because it is global and would involve nodes in the corner communicating with each other. This is the kind of calculation a pre-configuration phase would allow this to be set-up correctly, because every node would need to know the position of every other node to calculate if it is the furthest from the source. As the use of a pre-configuration phase is being avoided, other methods needed to be investigated to allocate nodes as fake sources.

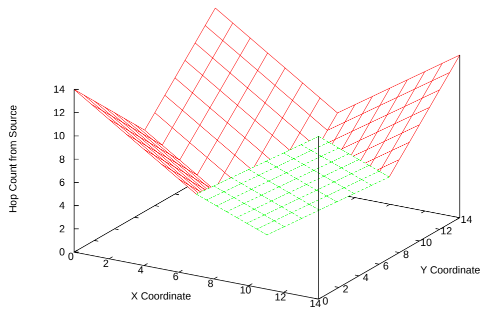
The first solution explored was to specify the corners of the network as the locations that fake sources should be created in. These are local maxima of source distanc) and at least one must be a global maxima. Figure 21 shows several graphs each with a source at a different location. With each of these graphs the highest point is the furthest from the source and thus probably the best location to have a permanent fake source. For example in Figure 21(a) the best place to put the permanent fake source would be at (14,14) which is the furthest point from the source (at the top right of the grid).



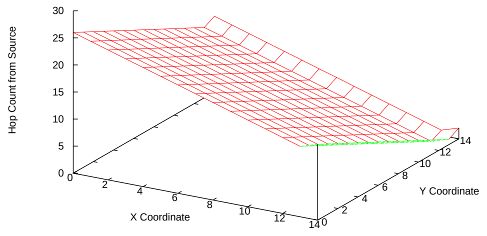
(a) Source at (0,0)



(b) Source at (2,2)



(c) Source at (7,7) - The centre node



(d) Source at (13,13)

Figure 21: Distance of each node from the source for different source positions

However, this method had several problems. The first being that by using the corners it limits the algorithm to perfect grid networks which in unrealistic in real world scenarios and goes against the purpose of this project. The second is that detecting corner nodes is difficult. What can be done instead is to refocus on not finding a perfect global maximum, but instead settle for several local maxima. Detecting these using the node's source distance is much easier than detecting corners.

When attempting to place the fake source in the position that is furthest from the source using its source distance, the following process was used: Every time a node receives a message it would record the minimum of the previously known source distance and the distance the message t has received had travelled from the source. Each time a node would broadcast a message it would include this knowledge. Whenever a node received a message with a higher distance then its own source distance it would unset a flag saying it was furthest.

This turned out to be problematic because the source distance could be lowered from a higher distance caused by collisions, to the lower distance that had been calculated. So nodes that were a local maximum might have believed there was a further node due to that node receiving a higher source distance due to a collision. To work around this the source distance the first time a node received a message from the source is recorded. The algorithm then includes the information on this first source distance it knows of and forwards this information in messages. When a node receives a message with a maximum known source distance that is greater than itself, it unsets a flag that indicates it is a permanent fake source candidate. Only nodes with this flag set can become permanent fake sources.

Switching to using the first source distance instead of updating it throughout the execution of the algorithm solved big collision related problem. Yet there was still another case to consider, what if a node is a permanent fake source and the flag is becomes unset? This is a common event as nodes many not receive the maximum source distance information due to collisions. The answer to this question is to unset the flag and revert the node to being a normal node when it receives a message with a larger maximum known source distance. Before it becomes a normal node it broadcasts a choose message so that more temporary fake sources can be created and then a permanent fake source at the location of this node that is further from the source.

So the procedure of the algorithm is to create temporary fake sources until a node is asked to be come a fake source and has a permanent flag set, in which case they become a permanent fake source. The nodes that have this permanent flag set are chosen by having nodes compare their source distance that was recorded the first time they received a message. This specialisation is called the ”Generic” specialisation.

3.5.5 Algorithm Specialisations

The above method worked very well on the Source Corner configuration of networks, which was the base configuration that was developed for. It didn’t however work well for the Sink Corner configuration. This was because a permanent fake source would be allocated, then deallocated once a further permanent fake source was created. This led to the fake sources attempting to draw the attacker through the source’s messages with just another permanent fake source. It simply didn’t work well for that configuration. So a new specialisation was designed for the Sink Corner network configuration.

As part of this specialisation, the algorithm would not longer try to allocate the node that is furthest from the source as a permanent fake source, it would simply allocate the first possible node as the permanent fake source. This would be done by that node setting a flag in the fake messages it broadcasted telling other nodes not to become a fake source. This specialisation is called the "Further" specialisation.

Another feature that needed to be changed was the ability for permanent fake sources to revert to normal nodes. This is because that is the enabling feature of the original idea that allowed permanent fake sources to be created at the furthest point and remove the closer fake sources. As this specialisation wants the fake sources allocated to remain allocated this feature was not desired, so it was removed for this specialisation.

Specialisation	Generic	Further
Permanent fake source Position	Furthest Point	First Point
Can revert PFS to normal node	Yes	No

Table 3: Algorithm specialisations features

3.5.6 Telling Messages Apart

In order to record if messages have been seen before, a hash is calculated at various stages. This hash function is intended to give a globally unique identifier to that message. It is possible that in the algorithm the content of the message may change, but the hash stays the same. This is because the message being sent by the source still has the same actual content, only the extra variables required by the algorithm have changed. This hash is calculated for all types of messages, every messages generated from it's source node will have a unique hash.

3.6 Algorithm

The Template algorithm is given in Appendix B, the contents of this section cover relevant functions required for the algorithm and well as an explanation of certain features and caveats of the algorithm.

3.6.1 Algorithm Overview

The following is an overview of the process of the algorithm:

1. The source node generates a ⟨NORMAL⟩ message, which is flooded through the network. This is repeated after given period until the simulation stops.

2. As normal nodes receive their first ⟨NORMAL⟩ message they set their source distance as the number of hops that message had travelled.

3. When the sink node receives their first ⟨NORMAL⟩ message, it broadcasts an ⟨AWAY⟩ message.

4. This ⟨AWAY⟩ message is flooded to the entire network and contains network knowledge (such as the sink-source distance).
5. The nodes that receive the ⟨AWAY⟩ message that are one hop from the sink become temporary fake sources.
6. These temporary fake sources broadcast ⟨FAKE⟩ messages at a given rate for a given duration.
7. Once that duration is up the temporary fake sources broadcast a ⟨CHOOSE⟩ message and become normal nodes again.
8. When a normal node receives a ⟨CHOOSE⟩ message it either becomes a temporary or permanent fake source, depending on if the permanent flag is set.
9. This fake source will start broadcasting ⟨FAKE⟩ message, if it is permanent it will not stop doing so.

3.6.2 Node Types

The following is an overview of the function of the different types of nodes:

Sink Receives ⟨NORMAL⟩ message sent from the Source.

Source Sends ⟨NORMAL⟩ messages when an asset is detected.

Normal Forwards on messages received, they provide a way for the source to communicate with the sink when those two nodes are not 1-hop neighbours.

Fake Source Generate ⟨FAKE⟩ messages to lure the attacker away from the source.

3.6.3 Message Types

The following is an overview of the function of the different types of messages:

NORMAL Contains information sent from the source.

AWAY Used to disseminate information from the sink.

CHOOSE Use to allocate temporary and permanent fake sources.

FAKE Same length as ⟨NORMAL⟩ messages, used to lure the attacker away from the source.

3.6.4 Functions

One aspect of the algorithm that would have been preferable, would have been the runtime detection of which specialisation to use. The way that is was intended to detect which specialisation to use was to check how close the sink was to the edge of the network. However, like detecting corners this is a difficult and global problem. So instead of calculating the specialisation to use at runtime, the specialisation needs to be given at set-up time by the network administrator, this is done by implementing the function IsNearEdgeOfNetwork.

The function GetAlgorithm is only ever called by the sink node, and calls the implemented IsNearEdgeOfNetwork to choose which specialisation to use. For example the configurations Source Corner and Generic 1 return false, the other configurations return true. So the network maintainer will need to decide which configuration is best represented by the selected configurations in order to choose their implementation.

In general, if there is space behind the sink for fake sources to be created then IsNearEdgeOfNetwork should return False, leading to the Generic specialisation being chosen. If the sink is in a position that fake sources will have to be created in the direction of source then returning True and using the Further specialisation is the better option.

Function 1 Calculate the specialisation to use in the Template algorithm

```
function GetAlgorithm($j$)
    if IsNearEdgeOfNetwork($j$) then
        return Further
    else
        return Generic
    end if
end function
```

Function 2 Indicates if the node is near the edge of the network - for Source Corner and Generic 1 configurations

```
function IsNearEdgeOfNetwork($j$)
    return False
end function
```

Function 3 Indicates if the node is near the edge of the network - for Sink Corner, Further Source Corner and Generic 2 configurations

```
function IsNearEdgeOfNetwork($j$)
    return True
end function
```

3.6.5 Ignore Choose

The algorithm proposed by [17] implemented an ignore choose optimisation in their algorithm and it has also been implemented in this algorithm, albeit with some modifications to handle different configurations.

The ignore choose optimisation partitions the network into an area of nodes that ignore choose messages and an area that doesn't. The area in which choose messages are ignored is an area close to the source in order to prevent fake sources from being created in the direction of the source. The benefit is two-fold, firstly fewer fake sources mean fewer fake messages and a lower energy usage, secondly the created fake sources do not lead the attacker to the source. What needs to be decided upon is the area of nodes that ignore the choose messages.

To begin with let us explore the case when there is room behind the sink for fake sources to be created, this is targeted at the Generic configuration. Here, the area of nodes from the source to just behind the sink ignore choose messages using Equation 1. However, due to collisions the path taken from the sink to the source may be longer than the minimum possible. This means that if the first source distance recorded is used to calculate the ignore choose range and the first distance is greater than the minimum source distance, then the ignore choose area will cover the sink. This will prevent any temporary fake nodes from being created. In order to allow for this issue a smaller ignore choose area is used Equation 2, this means there is less chance of the sink being in the ignore choose area when the source distance is longer than expected.

$$H_{source} < \Delta_{sink-source} \tag{1}$$

$$H_{source} \leq \frac{3}{4}\Delta_{sink-source} \tag{2}$$

The second case that needs to considered is when there is no room behind the sink for the temporary fake sources to be created. This specialisation needs to be considered separately because the previous predicates would not allow the creation of temporary fake sources towards the source, as might be needed. So Equation 3 is used to define the ignore choose area. This allows temporary fake sources to be created towards the source, but not too close. It also allows them to escape from the corner the sink is in, this is done by ensuring that the ignore choose area never reaches the network edge, ensuring a choose message can always go around the ignore choose area.

$$H_{source} \leq \frac{\Delta_{sink-source}}{2} - 1 \tag{3}$$

3.6.6 Disseminating Network Knowledge

As has been have previously mentioned, collisions make disseminating knowledge quite tricky. The algorithm needs to transmit a variety of information that is calculated at the sink to all nodes the first time it receives a normal message. Among this information is the $\Delta_{sink-source}$, Δ_{sink} the and the algorithm $\mathcal{A}$ to use. To ensure that as many nodes as possible receive this very important information, once the sink has received it's first normal message, it calculates the required information then broadcasts it in the form of an away message. In order to increase the likelihood of this message being received by as many nodes as possible, nodes first sense the channel before broadcasting and then rebroadcast several times.

3.7 Results

Simulations were run for every grid configuration, varying the network size, source rate, the fake source rate, the temporary fake source duration and the temporary fake source probability. The permanent fake source probability was fixed at 100%. Simulations were repeated 5000 times for each combination of parameters and the results were averaged to give the results included in this report. To work out the capture rate the number of simulations that reported the source as captured within the safety period was divided by the number of simulation runs (5000).

3.7.1 Selective Configurations

The graphs (and the results tables) only include results for the grid configuration because that was the only configuration the Template algorithm was run on. This was due to the large number of simulations that would need to be run if the Template algorithm was tested on ring and circle configurations. So the Template algorithm is only analysed for these grid configurations. It is left to the Adaptive algorithm to show that the general principles work. The Adaptive algorithm should have fewer parameters than this Template algorithm, so it should be more feasible to test it and analyse the results.

3.7.2 Functions Required

One of the main aims of the Template algorithm was to investigate what functions are required to be implemented for each type of node. Table 4 specifies which functions are required and which functions are optional for every node type. The lists that follow the table explain why certain functions are required or optional for given node types. They are designated ‘Req’ if they are required and ‘Opt’ if there are optional.

	Sink	Source	Normal	Fake
timeout(Send Normal)		Yes		
timeout(Send Fake)				Yes
timeout(Stop Sending Fake)				Optional
rcv(Normal)	Yes		Yes	Yes
rcv(Fake)	Yes	Optional	Yes	Yes
rcv(Away)		Yes	Yes	
rcv(Choose)			Yes	

Table 4: Functions required by the node types

Sink

Req rcv(Normal): as that is its function to act as the ⟨NORMAL⟩l message destination

Req rcv(Fake): to forward it as that is part of the flooding protocol to lure the attacker away from the source

Source

Req timeout(Send Normal): as that is its function to send ⟨NORMAL⟩ messages

Req rcv(Away): to know the algorithm running and other sink information

Opt rcv(Fake): to learn more network information

Normal

Req rcv(Normal) to forward it as that is part of the flooding protocol

Req rcv(Fake) to forward it as that is part of the flooding protocol to lure the attacker away from the source

Req rcv(Away) to learn the algorithm and to forward it onwards

Req rcv(Choose) to potentially be chosen as a fake source

Fake

Req timeout(Send Fake): as that is its function

Opt timeout(Stop Sending Fake): needed for temporary fake sources, not for permanent

Req rcv(Normal): to forward it as that is part of the flooding protocol

Opt rcv(Fake): to forward it as that is part of the flooding protocol to lure the attacker away from the source

Of the functions not mentioned for these node types there are functions that should never be implemented for various reasons. First is that the sink should never receive an away or choose, as it is the one that generates the initial message so would be pointless. Second is that the source should never receive a normal as it has generated it. Third is that the source should never receive a choose as it will never become a fake source. Fourth is that fake nodes should never receive an away because they will not exist when the first away wave is sent. Finally fake nodes should never receive a choose message as they have already decided to become a fake source.

3.7.3 Table of Results

The results in Appendix D contain results with the strictly worse data removed, this is data that has a worse capture rate and a worse number of fake messages sent. The percentage received is included in the results because it is an important metric, but it is not considered when removing the strictly worse results as it leads to too few results being removed. Each table contains the results for a given configuration and network size. Each table is split up into three vertical sections as designated by double vertical lines. The section on the left contains parameters that need to be known. The middle section are the parameters being varied to see which are best for this configuration. The section on the right contains the results of the simulations.

To pick the parameters to set up a network with, simply find the table that corresponds to your network configuration and size. On that table find the section that deals with the source period being used in the network. The middle values are the parameters needed to be given, the values on the right show our results when using them. Choose these parameters on the basis of whether optimising for capture ratio or energy usage is desired.

3.7.4 Table of Bad Results

The results in Appendix E are results for the Source Corner and Sink Corner configurations where the incorrect algorithm specialisation was used (Generic for Sink Corner and Further for Source Corner). To make reading the results tables easier they have been coloured such that when the incorrect algorithm has done worse than the correct algorithm the row is coloured green, otherwise the row is coloured orange. Each result cell contains three values, the first is the result of the configuration and algorithm, the second is the difference between these results and the results that use the correct specialisation and the final value is the percentage difference between these results and the results for the correct specialisation.

The Sink Corner results show that the capture rates are good, but the incorrect algorithm-configuration combination has better energy usage and better receive rates. The better capture rate is down to the fact that the 'ignore choose' area covers any possible permanent fake source nodes, so no permanent fake sources and very few temporary fake sources are created. This means fewer messages luring the attack from the source, increasing the capture rate. The worse energy usage and receive rates are caused by two factors. The first is due to fewer fake sources being created that leads to fewer fake messages being sent, the second is caused by the capture rate being higher which leads to the simulation terminating early, this decreases the amount of time fake messages have to be generated in.

The Source Corner results are very confused. It was expected for them to show worse capture rates with fewer fake messages and a better receive rate. This would have been due to the Further having a small 'ignore choose' range, meaning more temporary fake sources are generated and also more fake messages sent. However, this pattern is not clearly observed clearly, as a lot more results show that using what was considered to be the wrong algorithm to be better. This result is most likely caused by the Further specialisation trying to allocate a permanent fake source at the first possible node, not the furthest node from the source. So the algorithm would end up creating fewer temporary fake sources and thus sending fewer messages. Overall though the differences between the algorithms is so small that it is difficult to say conclusively that one specialisation is better than another for this configuration. So the Generic specialisation for the Source Corner configuration will continue to be used because it gives better misdirection by creating a permanent fake source at the furthest point.

3.7.5 Graphical Analysis

A selection of graphs can be found in Appendix C, these show capture rates and the number of fake messages sent when varying different parameters and network configurations. These graphs show that patterns that emerge when varying parameters.

The patterns that were observed are similar to previous work's results (with regard to the Source Corner configuration on a grid network [17]). In general larger networks have a lower capture ratio and a high fake source duration, a higher fake source rate and a higher temporary fake source probability all lead to a lower capture ratio. These results make sense because in larger networks there is a greater distance for the attacker to travel, meaning a greater time for them to be lured away by the fake sources. The other 3 parameters cause a higher number of fake messages to be generated, leading to a larger pull on the fake source, producing a lower capture ratio.

However, the graphs that show the results when varying the rate of normal messages show some contradictory behaviour. When the fastest broadcast rate leads to a higher capture ratio. What is most likely happening is that as more and more messages are being generated the number of collisions also increases. As there are more collisions, fewer fake messages reach the attacker, so there is less of a pull towards the fake source.

A separate interesting trend is that a faster source rate leads to a lower number of messages sent. One would expect the opposite to be true, where a faster source rate leads to more messages sent. However, due to the way the safety period is calculated it means that simulations where the source is broadcasting at a faster rate have a shorter safety period. As the safety period is shorter for simulations with faster source broadcast rates it means there is less time for nodes to broadcast compared to simulations with a slower source rates, and thus this is why this behaviour is observed.

Overall, for each configuration the algorithm provides at least one parameter set with a low capture rate of less than 5%. Although the parameters to achieve this capture rate are not the same for different network configurations. There also exists some leeway in choosing the parameters depending if one wishes to optimise for energy usage or capture ratio.

3.8 Summary

As part of the Template section of this project, lots has been achieved. The configurations that will be used have been identified and their safety period have been calculated. Most importantly two specialisations to handle different sink and source positions have been developed and the Template algorithm has been developed on top of them. Finally the results from the Template algorithm appears to be very good, with many parameter sets that give very good capture ratios.

4 Adaptive Algorithm

4.1 Introduction

One of the most important concerns with wireless sensor networks is power usage due to the fact that wireless sensor nodes have a finite source of energy. Therefore, it is important to minimise energy usage to maximise their lifetime. The most energy consuming task is having the radio active [32], this occurs when a message is being sent or received. So anything that can be done to decrease the number of messages sent, will lead to a longer lifetime of wireless nodes making them more useful for a longer period of time.

The aim of this algorithm is twofold. Firstly, the aim is to decrease the number of parameters that the algorithm needs to be provided with by determining them on-line. Secondly, extra on-line knowledge of the network should be used to decrease the number of messages that needs to be sent.

4.2 Template Similarities

As this algorithm is building on the previous work of the Template algorithm much of what has been discussed also applies here. The only main difference is in how the algorithm generates fake messages, but there are also other optimisation that take advantage of network knowledge. The configurations identified and the safety periods (in Appendix A) for those configurations will be reused in this algorithm.

4.3 Algorithm

The aim of decreasing energy usage can be achieved in numerous way, however, in the context of this project it means one thing - decreasing the number of messages sent. This is because using the radio is the most energy expensive task and also because it would require to much effort to calculate energy usage. The following are the optimisations implemented on top of the Template algorithm, that are used to decrease the number of messages sent. As well as reducing the number of messages sent, these optimisations were designed to reduce the number of parameters as well.

4.3.1 Range Limited Messages

In order to save energy how far some messages travel can be limited. To work out how far the messages should travel, the configuration selected and other network knowledge can be utilised. There are however bounds on the minimum distance these messages need to travel. This section will discuss how these bounds were calculated.

Normal messages that are generated by the source are only ever generated with the intention of reaching the sink node. This means that those messages should only every need to travel a distance of $\Delta_{sink-source}$ hops from the source. To handle collisions and the randomness of the path messages can take I limit the normal messages to only be forwarded by nodes that have a Δ_{source} that is less than or equal to $1.125 \times \Delta_{sink-source}$. The extra distance is to allow the message to approach the sink from multiple directions to mitigate the effects of collisions.

Another type of messages that it makes sense to limit the range of is fake messages. However, as they have no real destination in mind no firm limit can be given on how far they need to travel. Because of this an arbitrary limit has been chosen, that from the results, helps achieve better results. The method is to slowly decrease the range of the fake messages as fake sources generate those messages. A fake source distance cannot be used to limit the range like was done for normal messages, so the number of hops a message can travel is limited instead. This means that the message is vulnerable to the randomness of the path, but as it has no destination in mind this is not an issue. The reason that the range decreases over time that it is assumed that previous messages have reached the attacker and lured it towards the fake source. As the attacker should now be closer there should be less of a need for a fake message to travel further.

In the case of the Generic configuration, the temporary fake sources should have led the attacker behind the sink and away from the source. So by the time a permanent fake source is created it should be out of range of the normal messages. Because the permanent fake source can be created in a position that is outside the range of normal messages the range can decrease from Δ_{source} to $0.4 \times \Delta_{source}$. The aim here was to get the coefficient for the minimum down as low as possible to reduce the number of fake messages that would potentially collide with normal messages, with the intention of this increasing the receive rate.

In the Further configuration the attacker probably won't have been lured out of the normal range. So instead of decreasing the fake range to the low level used for the Generic configuration, the range decreases to a higher level from Δ_{source} to $0.75 \times \Delta_{source}$. This means that the fake messages still have a large minimum area of influence and leave out an area that is unlikely to be occupied by the attacker.

These two optimisations should help improve both the capture rate and the number of fake messages sent. The number of fake messages sent should be decreased, as fewer are being generated and they are not travelling as far. Normal messages are travelling for a shorter distance, so should have less of an influence on the attacker, meaning the capture rate should decrease.

4.3.2 Ignore Choose

As with the Template algorithm (and the algorithm developed for [17]) it is advantageous avoid fake sources being created in the direction of the source. So I used the same method described in subsubsection 3.6.5 in the Adaptive algorithm.

4.3.3 Temporary Fake Source Message Generation

One of the biggest focuses of the algorithm is the creation of the temporary fake sources and their ability to lure the attacker away from the source. Their importance is also reflected in the Template algorithm where 3 parameters are devoted to them (probability, rate and duration). From the Template algorithm's results it was observed that decreasing the probability leads to a worse capture ratio with a negligible decrease in the number of messages sent, so it was decided to leave the probability of temporary fake sources at 100% for the Adaptive algorithm. This leaves the rate and duration to be configured using network knowledge.

4.3.4 Temporary Fake Source: Number of Fake Messages to Send

Choosing how many fake messages to broadcast depends on the strategy being used. Initially, the number of fake messages to broadcast was set to the approximate distance of the attacker, but this produced poor results because too few fake messages were generated. Finally, for the Generic specialisation, the method that was settled was Equation 4, this method tries to pull the attacker back to the current node assuming it is still at the sink. For the Further specialisation Equation 5 is used, this method generates enough messages to pull an attacker away from the source node.

$$max(\Delta_{source} - \Delta_{sink-source}, 1) \tag{4}$$

Figure 22: How many fake messages to send for the Generic Specialisation

$$max(\Delta_{source}, 1) \tag{5}$$

Figure 23: How many fake messages to send for the Further Specialisation

4.3.5 Temporary Fake Source: Time Until Next Fake Message Broadcast

When a temporary fake source is sending messages, when the next fake message is sent needs to be decided upon. There are numerous strategies to handle this, but the aim of this algorithm is to take advantage of network specific knowledge. With this knowledge very important constant is known, which is the maximum time it will take to send a message from one node to another (T_{send}). Using this, exactly when the temporary fake source should broadcast the next message can be calculated, so it will (i) be broadcasted as soon as possible and (ii) broadcasted without colliding with the last message broadcasted. To do this the temporary fake source must wait for the original message to have travelled three hops. If it waits for the previously sent message to only travel one or two hops then collisions will occur as shown in Figure 24. This means the temporary fake source must wait for $3 \times T_{send}$ time units before sending the next message.

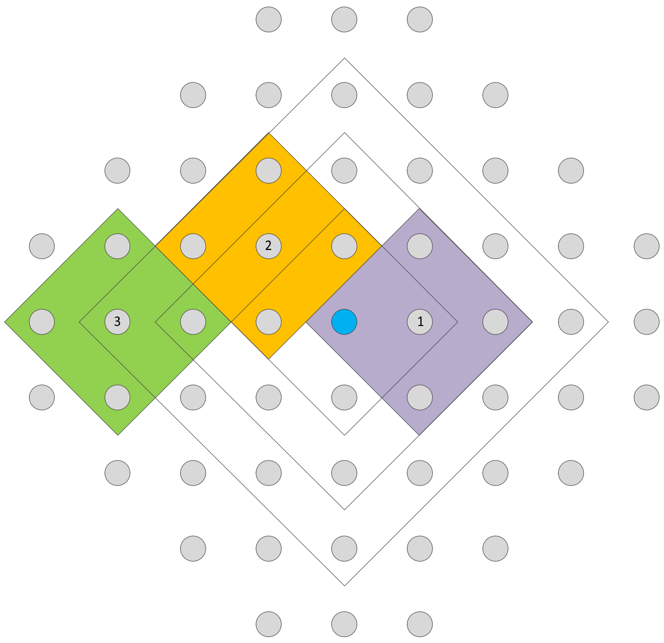


Figure 24: Collision graph for nodes broadcasting at 1, 2 and 3 hops from the source

In Figure 24 nodes that have a broadcast range that includes nodes in the sources broadcast range will cause collisions if either attempts to broadcast. So the minimum distance from the source when a broadcast will not collide with a source broadcast required is three hops.

4.3.6 Permanent Fake Source Message Generation

Once a permanent fake source is created it can be assumed that the temporary fake sources have lured the attacker towards the position of the current permanent fake source. This assumption is not that the attacker is very near, just that it is in the general area. As it is close the fake messages can continue to be range limited, but as the attacker is closer a slower rate can also be used. From the Template results it can be observed that a rate that is faster than the source rate will produce low capture rates, so the rate that is chosen should be faster than the source rate, but not as fast as it broadcasted in the Template algorithm.

$$0.85 \times P_{source} \tag{6}$$

Figure 25: How often to generate a Fake message for the Generic Specialisation

$$0.55 \times P_{source} \tag{7}$$

Figure 26: How often to generate a Fake message for the Further Specialisation

Equation 6 is used to decide the period of a permanent fake source for the Generic specialisation and Equation 7 is for the Further specialisation. A higher period (and thus a slower rate) can be used for the Generic specialisation as the fake sources should have led the attacker behind the sink which should be outside the normal message's range. For the Further specialisation the permanent fake source is likely to be created inside the normal message's range, so a much higher rate is required to effectively keep pulling the message towards the fake source rather than the real source.

4.3.7 Choosing a Permanent Fake Source

For this algorithm the same technique as described in subsubsection 3.5.4 for the Template algorithm was used. However, instead of having the duration and rate provided as parameters they are calculated at run-time as previously described.

4.4 Issues

With the Adaptive algorithm there were several issues that were faced, many were the same as when developing the Template algorithm. But developing the algorithm to be better than the Template algorithm was harder as network knowledge had to be used in new ways to improve upon the Template results. This was difficult because there was no set way to decide how to use the information, so time had to spend investigating different ways of using the knowledge.

An important problems to consider was that many of the optimisation ideas that were developed for the Adaptive algorithm could easily apply to the Template algorithm. The problem is defining the scope of 'network knowledge' and from there deciding which optimisations are applicable to the Template algorithm. Network knowledge would certainly include T_{send} (the time it takes a node to send a message to another node), meaning the way that fake messages are generated cannot be used for the Template algorithm. But with other optimisations, such as Ignore Choose or limiting message range, the decision is less clear. They definitely do take advantage of a kind of network knowledge - both the previous examples require knowledge of the $\Delta_{sink-source}$ (sink-source distance) - but is this the kind of network knowledge that was the focus for the Adaptive algorithm? If it is should the Ignore Choose optimisation be removed from the Template algorithm?

I think that the answers to these questions can be answered decisively. Firstly, the Ignore Choose optimisation should remain in the Template algorithm because it was in-place in previous work [17]. The other optimisations do take advantage of network knowledge, although different types of network knowledge. Hardware-based knowledge (e.g. T_{send}) and configuration-based knowledge (e.g. $\Delta_{sink-source}$) are the two types that should have been identified and focused on in the Adaptive algorithm. So the solution is to leave the Template algorithm as implemented and leave the Adaptive algorithm with its newly introduced optimisations based on this knowledge.

4.5 Results

Simulations were run for every grid, ring and circle configuration, varying the source rate and network size. Simulations were repeated 5000 times for each combination of parameters and the results were averaged to give the results included in this report. To work out the capture rate the number of simulations that reported the source as captured within the safety period was divided by the number of repeats.

The results are presented in a number of ways. In Appendix G the raw results and graphs for each configuration are presented. In Appendix H these raw results are compared with the results for the Template algorithm running on the same configuration with the same parameters. The tables that contain the comparison results are formatted such that the first column is a parameter that is shared between algorithms, the next 4 are specific to the Template algorithm and then the final three are the compared results. In each cell the compared results have two values the first is the difference between the Adaptive algorithm and the Template algorithm, the second is the percentage difference calculated using Equation 8.

$$\%\text{ Diff} = \frac{Adaptive - Template}{(Adaptive + Template)/2} \times 100 \tag{8}$$

The Adaptive algorithm was run on all the configurations previous identified, but because the Template algorithm was not, it has not been possible to compares Adaptive and Template results for the ring or circle configurations. It was simply done to show that this algorithm is capable to handling non-grid network structures.

4.6 Analysis

From the results in Appendix G the following patterns are observed for each of the configurations. When the size of the network increases the capture ratio decreases and the number of fake messages sent increases. The first observation is due to there being a greater sink-source distance meaning the attacker has a further distance to travel and a greater time for it to be lured away by fake sources. The second observation can be explained by the number of nodes being greater in a larger network, leading to more nodes flooding fake messages onwards.

When varying the source rate the pattern emerging is not as clear as varying the network size. What can be generally said that that with a slower source rate a higher number of fake messages is sent. This is due to the way the safety period is defined, leading a faster source rate having a smaller safety period, causing a shorter simulation time. As the simulation is not run as longer fewer messages were sent. This pattern was also explained within the Template results in subsubsection 3.7.5.

The graphs also show that in general slower source rates lead to lower capture ratios. This is most likely because at higher rates the number of collisions becomes very high, meaning lots of the fake messages are lost. As fewer fake messages are propagated there is less of a lure towards the fake sources.

4.6.1 Comparison with Template Results

In Appendix H are tables that compare the results of the Adaptive algorithm to the strictly better results of the Template algorithm. All results here are the same as the strictly better results in the Template tables, this is done to eliminate the obviously bad parameter sets.

The tables show the difference between the Template algorithm's values and the Adaptive algorithm's values. If the value was negative then the Adaptive algorithm either sent fewer messages or had a lower capture ratio, if the value is positive then the reverse is true. The aim is for the Adaptive to be better than the Template algorithm for the parameters that provide a good balance between capture ratio and number of fake messages sent for a given set of parameters (size and source period).

One pattern that does occur is that when the Template algorithm produced a high capture ratio, the Adaptive algorithm is shown to send many more fake messages. The reason for this is that when the source is captured in a simulation, that simulation is terminated. This means that the messages that would have been sent up to the safety period are not sent, leading to a low level of fake messages sent being recorded. So when this is compared against the Adaptive algorithm's low capture ratio and low number of messages sent, it appears as if the Adaptive algorithm requires more energy. In a sense this is true, but only because the Adaptive algorithm runs for longer in the simulator.

4.6.2 Comparison with Template Results

In Appendix H are the results tables that compare the Adaptive results with the Template results for the grid configurations. In the majority of results the Adaptive algorithm sees a reduced number of fake messages sent and a higher number of messages received by the sink. This is contrasted with a general increase in the percentage captured. The increase in the percentage captured is a maximum of 11%, but is usually much smaller and in some cases the Adaptive algorithm has a lower capture rate.

There are very few results where the Template algorithm produces better results in all three categories of capture rate, number of fake messages sent and receive rate. But the majority of results show that the Adaptive algorithm will usually lead to a lower energy usage. Overall there has been a trade-off between energy usage and the capture rate. The Adaptive algorithm has allowed the capture rate to be slightly higher, whilst reducing energy usage compared to the Template algorithm.

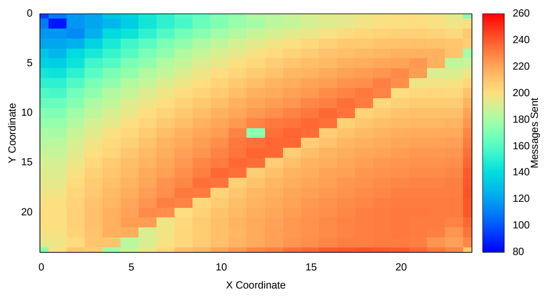
4.6.3 Similar Patterns to Template Algorithm

Overall most of the game general patterns observed when the Template algorithm was run on grid networks also occur and they occur for the same reasons. The capture rates decrease as the size of the network increases. A slower source rate leads to a higher number of messages sent due to the decreased safety period. This is except for the 0.125 source period exception, which still occurs because the same safety periods are used. The percentage of messages receives remains fairly constant when size varies, but decreases when the source rate increases.

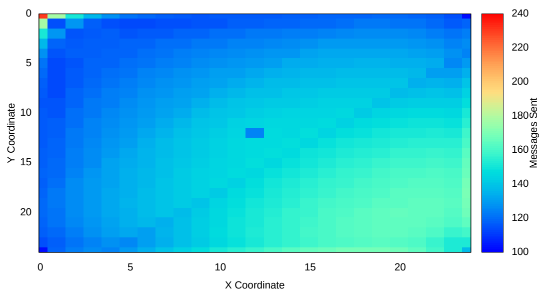
4.6.4 Spread of Energy Usage

One of the things that was potentially worrying was that that algorithm's aim was to end up with one node allocated as the permanent fake source and then have that node send a large number of fake messages into the system. Generating all these fake messages might cause a considerable energy drain, it is possible that using more fake sources that broadcast at a slower rate may use less energy.

The graphs in Figure 89 in Appendix I show heat maps for networks running the Adaptive algorithm for different source periods on grid networks of size 25. Size 25 was chosen because it provides a better resolution than smaller sized networks.



(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.125 seconds

Figure 27: Messages sent while running the Adaptive algorithm for the SourceCorner configuration

In these graphs there are a few things that can be very obviously seen. Firstly, the sink in the centre of the heat map stands out as it doesn't forward on normal messages and only forwards fake messages, so has a low overall number of messages sent. Secondly, there is a diagonal line near the sink that indicates the point at which normal messages stop being forwarded. It can also be observed where permanent fake sources exist as there is a very warm area where they generate fake messages in the bottom right hand corner. Finally, the source can be seen in the top left corner.

An interesting feature is that the corners in the top right and bottom left send fewer messages than their neighbours. This is likely caused by messages frequently colliding when corner nodes are trying to receive them, and thus the nodes do not receive as many messages that they would then forward onwards. A lack of collisions is probably responsible for a greater number of messages being sent along the bottom and right edges of the network.

Overall it seems that the energy usage around the permanent fake source is spread out, this is because all the surrounding nodes need to forward every unique fake message they receive from the permanent fake source. This means there is no high energy hotspot where the permanent fake source is and a low energy usage around it as was the worry.

4.7 Results for Other Structures

4.7.1 Reasoning

The only structures of wireless sensor networks that the Template algorithm has been tested on are grids. While it is possible that these networks will be set up in a grid-like fashion, it is also possible that they will not be. As it is infeasible to run simulations for all networks with different layouts, the two that were described earlier will be used: a circle and a ring.

These structures will only have data gathered for the Adaptive algorithm for several reasons: (i) less data is produced making it easier to analyse, (ii) the Adaptive algorithm incorporates the template aspect of the Template algorithm and (iii) it should give better results (for energy usage) than the Template algorithm would.

4.7.2 Patterns of Non-Grid Configurations

For the non-grid network configurations there are no Template results to compares them to, so the results will simply be compared to the observed patterns. The Circle Sink Centre and Circle Source Centre have results that follow the patterns previously observed. The Circle Edges configuration differs by producing an increase in the capture rate when increasing the size of the network for the two slowest source rates, but follows the usual pattern for the two fastest rates.

For ring configurations behaviour similar to that of the Circle Edges configuration is observed, where when increasing the network size the slowest source rate gives an increase in capture rates. Another problem with ring networks is that while the capture ratio is very low, so is the receive ratio.

All of the issues are most likely caused by the fact that that the specialisations were designed with a grid network in mind, with the aim of it working on any configuration. Because of this focus it means that other specialisations need to be written to handle certain cases. These cases are focused on the ring grids to improve their receive rate, but as ring networks are unlikely to be deployed it has not been worth devoting too much time to them. The results do show that the algorithm achieves an good level of source location privacy for most of the configurations.

4.7.3 Random Network Configuration

As well as the results for grid, ring and circle networks the algorithm has been informally run on a random configuration of nodes evenly distributed over a given area where the sink and source are guaranteed to be connected. From these informal results it was observed that either the simulation finished instantly, or that it was taking a long time for the attacker to capture the source. The first case was most likely because the sink and source were placed very close to each other, leading to a very quick capture. The second case appears to show that given a sufficient space between sink and source the Adaptive algorithm can perform well in a random configuration.

I decided to force the Generic specialisation, because I assumed that a random configuration will likely end up with space behind the sink for fake sources to be created in. I decided not to gather formal results because of the difficulty in working out what the safety period should be and then the issue of deciding whether to choose one random configuration or just use a new one for each individual simulation. Overall there were too many issues to resolve and too little time.

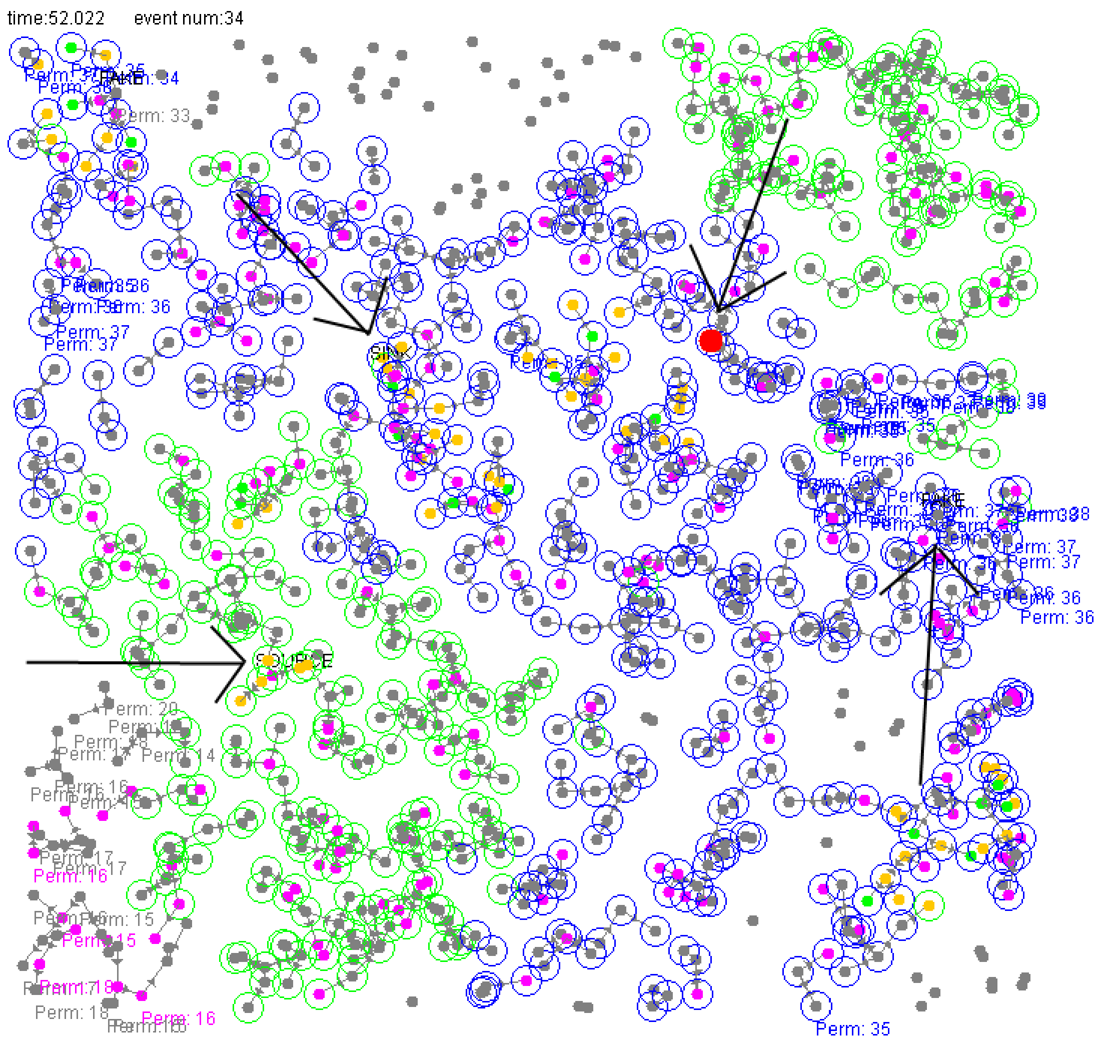


Figure 28: An example random network running the Adaptive algorithm with the Generic specialisation

Figure 28 shows an example of the Adaptive algorithm being simulated for a random configuration with 900 nodes. The source is in the mass of green ringed nodes in the bottom left of the network. The sink is north-east of the source. The attacker is east of the sink and is being drawn towards a permanent fake source to the south-east. In this network as mentioned before, the sink and source are connected, but some portions of the network are disconnected.

4.8 Summary

As part of the Adaptive section of this project, the Adaptive algorithm has been developed, building upon previous work. This section required fewer initial pieces of work to be done, as they had already been achieved as part of the Template algorithm. This section has investigated ways to reduce the number of messages that are sent while maintaining a similar capture rate as compared to the Template algorithm. Finally, results for non-grid configurations have been obtained which, in most cases, show a good capture rate.

5 Project Management

Due to the size and complexity of my third year project is has been very important for me to manage it well, whilst balancing completing it with my studies. Overall I feel that I achieved that goal, however, there were times when my project became my primary focus and this detracted from my studies. Although this didn't happen that often.

I managed to follow the schedule that I revised in my progress report fairly well. I had two problems that caused further delays. The first was that I was unable to use the cluster as I will explain later. Due to this inability to use the cluster I was unable to refine my algorithm based on results for the last few weeks of term. Instead this was done over the Easter holidays.

Activity	Time Allocated
Related reading on source location privacy	1 week
Learning simulation tool (JProwler)	1 week
Implementing template algorithm	5 weeks
Testing algorithm	1 week
Gathering results	1 week
Writing up	1 week
Developing adaptive algorithm	6 weeks
Testing adaptive algorithm	1 week
Gathering results	2 weeks
Writing up	1 week

Table 5: Project Time Allocation

I managed to do a large amount of the write-up during term time as I was proceeding with my project, this helped me understand where I was with my project and what I still needed to do. I did not manage to finish the entire write-up during term time though, as I was focusing on getting results. So the majority of the report writing was left until the Easter holiday, this should have been planned for in my specification.

5.1 Source Control Management

Initially I had asked the department to provide me with a subversion account, as this had been done during the second year group project I had assumed that it would be done fairly quickly. However, after about a month into the first term I had yet to have been provided with a repository. As I desired a place to keep the development history and also a place that would provide a back-up of my project if I lost my main development machine, I decided to stop waiting for the department to provide me with a subversion repository and get my own. I settled with BitBucket as they offer free private repositories. I ensured that my repository was set to private to ensure no one else could access the work I was doing.

5.2 Access to Academic Papers

One of the resources I knew that I was going to need was access to academic papers. I found that by using the Athens or Shibboleth login provided by the University [37], as well as the Department of Computer Science subscribing to some of the major journals, that is was very easy to access academic papers that I needed. I also took advantage of Google Scholar when performing background research, taking advantage of their reference downloading feature. So overall getting the academic papers I needed was not a problem.

5.3 Cluster

The Department's High Performance Cluster was initially non-problematic for me during the first term. Unfortunately this was not when I was trying to gather the majority of my data. During the second term I encountered a number of problems, ranging from the cluster being down during Christmas and then having various issues with it during the Second Term (such as it being used for the High Performance Computing Module CS402 [36]). Initially I looked to implement a simple cluster of my own using BOINC [1], however setting up a server would have taken too long and detracted from my actual project. So once term started I contacted Oliver Perks in attempt to resolve the problems I was having. As the problems were fairly hard to diagnose this period of a partially working cluster continued for some time. Eventually I decided to work around the problem, this was done my modifying my cluster submission scripts to not submit jobs that I had obtained results for. This allowed me to rerun my submission scripts to ensure I gathered all the results required.

5.4 A Research Project

As this was a research project I found that my original estimates ended up being significantly different than from before. I overran the deadline to finish the Template algorithm and finished the Adaptive algorithm very quickly. From this experience I believe that it is possible to give an estimate of the time a task will take, but it is incredibly difficult to give a reliable schedule to a research project. This was because the time taken to come up with new ideas to solve problems (or to research existing solutions) can vary wildly, as it depends on a persons ability to come up with those ideas. Some ideas may be forthcoming, other may not be and there is little control over how innovative someone is.

5.5 Evaluation

I feel that my project went very well which was probably due to my previous experience from the URSS project I undertook during the summer. However, I was much more independent in this project. I think this ended up having both a positive and negative outcome on my project. On one hand I was free to do what I want when I wanted to and to organise my self as I wished rather than working for someone else. This freedom was definitely beneficial to me and allowed me the flexibility I needed to complete my other work. On the other hand because I was independent I needed to come up with my own ideas rather than have someone to guide me through. Coming up with ideas was by far the hardest part of this project. But then I suppose that is true of any work that is trying to come up with an original solution.

5.6 Summary

I encountered a few problems during my project, some of which were not a simple case to resolve and led to wasted time. These were problems that were not possible to easily avoid or resolve and were expected in this sort of project. if undertaking a project similar to this in the future then I would definitely allocate more time to coming up with ideas and gathering results as they ended up being the most time consuming activities.

6 Future Work

With any project there is always more that can be done to improve upon the work completed. The following are several ideas that either directly expand upon the work done within this project, or are related to it in some way.

6.1 Improve Directly on this Work

Due to time constrains I have run out of time to continue optimising my algorithms, the following are some things I would like to do to improve my work. In general they should be straightforward.

- Improve the Adaptive algorithm to make it better than the Template algorithm in all cases
- Identifying and developing for more configurations

6.2 Better Attacker Intelligence

My attacker is a simple reflex agent; when it receives a message that it has not previously processed it moves to the source of that message. In a real world scenario the attacker will be a human equipped with radio receiving equipment that it is using to detect the source of messages. A human is most definitely not a simple reflex agent and is capable of complex behaviour. I would suggest either investigating co-operating attackers or attackers with memory. These are likely to introduce behaviour in the attacker that is harder to handle, whilst still being manageable to solve.

6.3 Spread out Fake Sources

Currently I aim to only create a single permanent fake source. This means that one node is depended upon to provide the fake message to lure the attacker away. As wireless sensor nodes are energy constrained this means that it is possible that a large amount of energy is being drawn from a single node. I began to investigate this issue and it appears that there is no hotspot of large energy usage, but it would be good to investigate this properly.

6.4 Handle Hardware Failures

Wireless sensor nodes are unreliable [34, 39], they work in outdoor conditions where electronics do not usually fare very well. It is very possible for the components to fail or for the node to run out of energy. When this occurs the algorithms to provide source location privacy should be able to handle this failure. However this failure is handled would need to continue to take energy usage into account, for example, periodically broadcasting alive messages may be infeasible due to the high energy cost. This means that other techniques must be carefully considered.

6.5 Comparison with other Techniques

This project has solely investigated using fake sources to provide source location privacy, yet there exist many other techniques to provide it, for example Phantom Routing [25]. It would be very beneficial to undertake a project that investigated the advantages and disadvantages of the different techniques and to see which algorithms perform better in certain situations.

6.6 Use Probabilistic Flooding

In previous work probabilistic flooding has been used to improve the performance of flooding [18, 25], yet in this project it has not been used. It would be useful to investigate the effects that introducing it would have on metrics such as the capture rate, energy usage and the receive rate.

6.7 Iron Triangle

One of the things that I think has been done to simplify the study is to focus on two behaviours: energy usage and capture rates. I think that there should be a third and very important value to consider and that is the delivery ratio. I think that it is likely that these three behaviours form a kind of Iron Triangle (such as Time/Cost/Quality in the world of Project Management [22, p. 84-86]) where it will only be possible for a maximum of two to be optimised and the third will suffer. As part of this is would also be good to investigate other parameters such as message latency.

7 Conclusions

On the technical side, I have achieved my aim of developing a Template algorithm that investigates different network configurations and an Adaptive algorithm that reduces the number of parameters needed and the amount of energy consumed. I am happy with their state, but as always more could be done to improve upon them. I do wish that I had the time to do so, but the fact that there is a hard deadline to meet means this is simply not possible.

On the project side, I think the experience of undertaking, and importantly finishing, a large project has been personally beneficial. Real experience in managing time and meeting deadlines for a unique project will no doubt prove to be a very useful experience. This project has been hard work, frustrating at times, enjoyable at other and has definitely changed my views on what it means to undertake a project of this magnitude.

I think the most important thing I have taken away from this is that it is always best to work on something that you are interested in. There is no point dedicating your time to something that you find boring. I am incredibly luckily that I was introduced to these theoretical academic problems before I might have otherwise been and that I enjoy coming up with solutions to them.

Appendices

A Safety Period Tables

Table 6: Safety Periods for the **SourceCorner** configuration for **GRID** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	10	84	16.79	33.58
11	2	10	82	8.45	16.90
11	4	10	80	4.50	8.99
11	8	10	44	4.70	9.41
15	1	14	84	24.82	49.63
15	2	14	83	12.43	24.85
15	4	14	80	6.64	13.29
15	8	14	43	7.23	14.47
21	1	20	84	36.76	73.52
21	2	20	83	18.37	36.74
21	4	20	81	9.89	19.78
21	8	20	43	11.45	22.90
25	1	24	85	44.90	89.80
25	2	24	83	22.34	44.68
25	4	24	81	12.17	24.34
25	8	24	43	14.26	28.52

Table 7: Safety Periods for the **SinkCorner** configuration for **GRID** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	10	56	15.76	31.53
11	2	10	56	7.93	15.87
11	4	10	55	4.14	8.29
11	8	10	36	4.58	9.16
15	1	14	57	23.64	47.29
15	2	14	57	11.96	23.91
15	4	14	56	6.20	12.39
15	8	14	34	7.46	14.92
21	1	20	59	35.50	71.00
21	2	20	58	17.84	35.69
21	4	20	56	9.42	18.85
21	8	20	33	12.32	24.65
25	1	24	59	43.54	87.08
25	2	24	57	21.86	43.73
25	4	24	57	11.60	23.19
25	8	24	33	15.94	31.87

Table 8: Safety Periods for the **FurtherSinkCorner** configuration for **GRID** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	14	57	23.71	47.42
11	2	14	58	11.95	23.90
11	4	14	56	6.23	12.45
11	8	14	34	7.19	14.37
15	1	22	59	39.73	79.46
15	2	22	58	19.87	39.75
15	4	22	56	10.53	21.07
15	8	22	33	12.95	25.89
21	1	34	59	63.89	127.79
21	2	34	58	32.05	64.09
21	4	34	57	17.24	34.48
21	8	34	32	22.34	44.67
25	1	42	59	80.34	160.69
25	2	42	58	40.25	80.50
25	4	42	57	21.92	43.83
25	8	42	32	28.53	57.07

Table 9: Safety Periods for the **Generic1** configuration for **GRID** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	6	97	6.17	12.35
11	2	6	98	3.26	6.53
11	4	6	84	1.93	3.87
11	8	6	48	3.27	6.54
15	1	10	89	11.24	22.48
15	2	10	89	5.73	11.47
15	4	10	82	3.56	7.12
15	8	10	46	7.31	14.62
21	1	14	91	17.16	34.32
21	2	14	91	8.61	17.23
21	4	14	81	5.64	11.28
21	8	14	46	12.38	24.77
25	1	16	92	20.41	40.83
25	2	16	92	10.32	20.63
25	4	16	82	6.93	13.86
25	8	16	46	15.63	31.26

Table 10: Safety Periods for the **Generic2** configuration for **GRID** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	12	82	20.83	41.65
11	2	12	82	10.53	21.06
11	4	12	79	5.64	11.28
11	8	12	46	6.72	13.45
15	1	20	83	36.76	73.53
15	2	20	83	18.56	37.12
15	4	20	80	10.01	20.01
15	8	20	44	12.39	24.79
21	1	32	84	61.26	122.52
21	2	32	83	30.79	61.57
21	4	32	81	16.67	33.35
21	8	32	43	21.41	42.82
25	1	40	84	77.47	154.94
25	2	40	83	38.79	77.57
25	4	40	81	21.26	42.51
25	8	40	43	27.51	55.02

Table 11: Safety Periods for the **RingTop** configuration for **RING** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	10	90	13.04	26.09
11	2	10	88	5.68	11.36
11	4	10	83	2.89	5.78
11	8	10	51	1.48	2.96
15	1	14	87	20.09	40.18
15	2	14	85	8.23	16.46
15	4	14	79	4.00	8.01
15	8	14	55	2.03	4.07
21	1	20	81	25.50	51.00
21	2	20	79	12.04	24.07
21	4	20	66	5.90	11.81
21	8	20	58	2.98	5.96
25	1	24	77	32.71	65.41
25	2	24	71	15.04	30.09
25	4	24	67	7.30	14.61
25	8	24	58	3.67	7.33

Table 12: Safety Periods for the **RingOpposite** configuration for **RING** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	20	54	23.66	47.31
11	2	20	53	12.05	24.11
11	4	20	51	6.10	12.20
11	8	20	45	3.10	6.20
15	1	28	58	35.02	70.03
15	2	28	56	17.62	35.24
15	4	28	53	8.89	17.78
15	8	28	48	4.48	8.96
21	1	40	58	54.89	109.79
21	2	40	57	27.57	55.14
21	4	40	54	13.87	27.73
21	8	40	49	6.96	13.92
25	1	48	56	70.49	140.98
25	2	48	55	35.42	70.84
25	4	48	53	17.71	35.42
25	8	48	48	8.90	17.79

Table 13: Safety Periods for the **RingMiddle** configuration for **RING** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	20	54	23.71	47.42
11	2	20	53	12.04	24.08
11	4	20	51	6.11	12.22
11	8	20	46	3.10	6.19
15	1	28	58	35.00	70.00
15	2	28	56	17.63	35.26
15	4	28	53	8.89	17.78
15	8	28	48	4.48	8.96
21	1	40	58	54.95	109.91
21	2	40	56	27.60	55.19
21	4	40	54	13.84	27.68
21	8	40	49	6.95	13.89
25	1	48	56	70.55	141.11
25	2	48	55	35.37	70.73
25	4	48	53	17.75	35.49
25	8	48	48	8.90	17.79

Table 14: Safety Periods for the **CircleEdges** configuration for **CIRCLE** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	14	65	23.50	47.00
11	2	14	65	11.94	23.88
11	4	14	63	6.28	12.55
11	8	14	35	6.45	12.90
15	1	18	69	29.59	59.18
15	2	18	69	14.88	29.76
15	4	18	67	8.04	16.09
15	8	18	36	9.42	18.84
21	1	26	69	43.99	87.99
21	2	26	67	21.96	43.92
21	4	26	67	12.48	24.96
21	8	26	36	16.02	32.04
25	1	30	69	49.99	99.97
25	2	30	68	25.08	50.16
25	4	30	67	14.88	29.76
25	8	30	36	19.83	39.67

Table 15: Safety Periods for the **CircleSourceCentre** configuration for **CIRCLE** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	7	65	9.99	19.98
11	2	7	66	5.21	10.42
11	4	7	65	2.71	5.41
11	8	7	42	2.88	5.76
15	1	9	67	12.90	25.80
15	2	9	67	6.54	13.09
15	4	9	66	3.44	6.88
15	8	9	41	4.41	8.82
21	1	12	66	17.04	34.08
21	2	12	66	8.64	17.28
21	4	12	64	4.72	9.44
21	8	12	40	7.39	14.79
25	1	15	68	22.56	45.13
25	2	15	68	11.36	22.72
25	4	15	66	6.30	12.59
25	8	15	39	10.26	20.52

Table 16: Safety Periods for the **CircleSinkCentre** configuration for **CIRCLE** networks

Size	Rate msg / sec	Source-Sink Distance (hop)	Received (%)	Average Time Taken (seconds)	Safety Period (seconds)
11	1	7	82	10.73	21.47
11	2	7	85	5.59	11.18
11	4	7	81	2.97	5.95
11	8	7	46	3.01	6.02
15	1	9	84	14.06	28.13
15	2	9	84	7.13	14.25
15	4	9	81	3.77	7.54
15	8	9	45	4.34	8.68
21	1	12	83	18.15	36.30
21	2	12	83	9.07	18.14
21	4	12	79	5.06	10.11
21	8	12	45	6.89	13.79
25	1	15	83	23.75	47.49
25	2	15	83	11.84	23.68
25	4	15	80	6.70	13.40
25	8	15	44	9.26	18.52

B Template Algorithm

```
message - Normal
variables
    % Sink-Source Distance
    ssd: int;

    % The hop count from the source
    hop: int;

    % What number message this is. Starts at 1
    count: int;

    % The maximum hop seen
    maxHop: int;
```

Figure 29: Source Location Privacy Algorithm - Normal Message

```
message - Fake
variables
    % Sink-Source Distance
    ssd: int;

    % Is from a permanent source
    Fperm: boolean;

    % The maximum hop seen
    maxHop: int;

    % The $\Delta_{source}$ of the sending node
    FDsrc: int;

    % The $\Delta_{sink}$ of the sending node
    FDsink: int;

    % The id of the sending node
    Fid: int;
```

Figure 30: Source Location Privacy Algorithm - Fake Message

```
message - Away/Choose
variables
    % Sink-Source Distance
    ssd: int;

    % The number of hops travelled from the sink
    Dsink: int;

    % The maximum hop seen
    maxHop: int;

    % The algorithm
    algorithm: Algorithm;
```

Figure 31: Source Location Privacy Algorithm - Away/Choose Message

$$\mathbf{min}_{\perp}(x) = \begin{cases} \perp & x = \{\perp\} \\ \mathbf{min}(x \setminus \{\perp\}) & otherwise \end{cases} \tag{9}$$

```
process $j$ - If type is Sink
**variables**
    % Messages seen
    $messages$: set of int init $\emptyset$;

    % Records if the first away message has been sent
    $sinkSent$: boolean init False;

    % Distance between the sink and source
    $\Delta_{sink-source}$: int init $\perp$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;

**constants**
    % The source period
    $P_{source}$: time init;

**actions**
    % Receiving Normal Message
    $SinkReceiveNormal$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$

    % Receiving Fake Message
    $SinkReceiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
```

Figure 32: Source Location Privacy Algorithm - Sink

```
    % Receiving Normal Message
    $SinkReceiveNormal$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$
        **if** $(hash \notin messages)$ **then**
            $messages := messages \cup \{hash\}$;
            $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, hop + 1\}$;
            **if** $(\neg sinkSent)$ **then**
                $sinkSent := True$;
                $\mathcal{A} := \mathbf{GetAlgorithm}()$;
                $\mathbf{BCAST}\langle Away, \mathbf{hash}(Away), \Delta_{sink-source}, 0, maxHop, \mathcal{A}\rangle$
                            **sense repeat**(3) **in** $\frac{P_{source}}{2}$;
            **fi**;
        **fi**;

    % Receiving Fake Message
    $SinkReceiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
        $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
        **if** $(hash \notin messages)$ **then**
            $messages := messages \cup \{hash\}$;
            $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm, maxHop, FDsrc, FDsink, Fid\rangle$;
        **fi**;
```

Figure 33: Source Location Privacy Algorithm - Sink

```
process $j$ - If type is Fake Source
variables
    % Messages seen
    $messages$: set of int init $\emptyset$;

    % The hop count of this node in the first wave of normal messages
    $firstHop$: int init $\bot$;

    % A flag that indicates if we think we should be a permanent fake source
    $isPerm$: boolean init $False$;

    % Distance to to the source, to the sink, between the sink and source
    $\Delta_{source}, \Delta_{sink}, \Delta_{sink-source}$: int, int, int init $\bot, \bot, \bot$;

    % period: how often fake messages are sent
    % duration: how long we will stay a Fake Source
    $period, duration$: timer, timer init $fakeSourcePeriod, fakeSourceDuration$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\bot$;

constants
    % Parameters given to the network
    $fakeSourcePeriod, fakeSourceDuration$: time, time;

actions
    % Sending Fake Messages
    $FSSendFake$:: timeout($period$) $\rightarrow$

    % Stop Sending Fake Messages after the given duration
    $FSStopSending$:: timeout($duration$) $\rightarrow$

    % Receiving Normal Message
    $FSReceiveNormal$:: rcv$\langle Normal, hash, ssd, hop, count, maxHop \rangle \rightarrow$

    % Receiving Fake Message
    $FSReceiveFake$:: rcv$\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid \rangle \rightarrow$
```

Figure 34: Source Location Privacy Algorithm - Fake Source

```
% Sending Fake Message when $\mathcal{A}$ is Generic Specialisation
$\mathit{FSSendFake}$:: $\textbf{timeout}(\mathit{period}) \rightarrow$
    $\textbf{BCAST}\langle Fake, \textbf{hash}(Fake), \Delta_{sink-source}, \mathit{duration} = \infty, firstHop, \Delta_{source}, \Delta_{sink}, j\rangle$;
    $\textbf{if}$ $(\mathit{duration} = \infty \wedge \neg \mathit{isPerm})$ $\textbf{then}$
        $\textbf{set}(\mathit{duration}, 0)$;
    $\textbf{else}$
        $\textbf{set}(\mathit{period}, \mathit{fakeSourceRate})$;
    $\textbf{fi}$;

% Sending Fake Message when $\mathcal{A}$ is Further Specialisation
$\mathit{FSSendFake}$:: $\textbf{timeout}(\mathit{period}) \rightarrow$
    $\textbf{BCAST}\langle Fake, \textbf{hash}(Fake), \Delta_{sink-source}, \mathit{duration} = \infty, firstHop, \Delta_{source}, \Delta_{sink}, j\rangle$;
    $\textbf{set}(\mathit{period}, \mathit{fakeSourceRate})$;

% Stop Sending Fake Messages after the given duration
$\mathit{FSStopSending}$:: $\textbf{timeout}(\mathit{duration}) \rightarrow$
    $\textbf{BCAST}\langle \mathit{Choose}, \textbf{hash}(\mathit{Choose}), \Delta_{sink-source}, \mathit{hop} + 1, \mathit{firstHop}, \mathcal{A}\rangle$ $\textbf{sense repeat}(3)$;
    $\textbf{BecomeNormal}()$;

% Receiving Normal Message
$\mathit{FSReceiveNormal}$:: $\textbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$
    $\Delta_{sink-source} := \textbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    $\textbf{if}$ $(\mathit{hash} \notin \mathit{messages})$ $\textbf{then}$
        $\mathit{messages} := \mathit{messages} \cup \{\mathit{hash}\}$;
        $\textbf{BCAST}\langle Normal, hash, \Delta_{sink-source}, hop + 1, count, \textbf{max}\{\mathit{firstHop}, maxHop\}\rangle$;
    $\textbf{fi}$;

% Receiving Fake Message when $\mathcal{A}$ is Generic Specialisation
$\mathit{FSReceiveFake}$:: $\textbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
    $\textbf{if}$ $(\mathit{firstHop} = \perp \vee maxHop - 1 > \mathit{firstHop})$ $\textbf{then}$
        $\mathit{isPerm} := False$;
    $\textbf{fi}$;
    $\Delta_{sink-source} := \textbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    $\textbf{if}$ $(\mathit{hash} \notin \mathit{messages})$ $\textbf{then}$
        $\mathit{messages} := \mathit{messages} \cup \{\mathit{hash}\}$;
        $\mathit{seenPerm} := \mathit{seenPerm} \vee Fperm$;
        $\textbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm,$
                    $\textbf{max}\{\mathit{firstHop}, maxHop\}, \mathit{FDsrc}, FDsink, Fid\rangle$;
        $\textbf{if}$ $(\mathit{duration} = \infty \wedge$
          $\mathit{Fperm} \wedge$
          $((\mathit{FDsrc} > \Delta_{source}) \vee$
          $(\mathit{FDsrc} = \Delta source \wedge \Delta_{sink} < FDsink)$
          $(\mathit{FDsrc} = \Delta source \wedge \Delta_{sink} = FDsink \wedge j < fromId)))$ $\textbf{then}$
            $\textbf{BecomeNormal}()$;
        $\textbf{fi}$;
    $\textbf{fi}$;

% Receiving Fake Message when $\mathcal{A}$ is Further Specialisation
$\mathit{FSReceiveFake}$:: $\textbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
    $\textbf{if}$ $(\mathit{firstHop} = \perp \vee maxHop - 1 > \mathit{firstHop})$ $\textbf{then}$
        $\mathit{isPerm} := False$;
    $\textbf{fi}$;
    $\Delta_{sink-source} := \textbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    $\textbf{if}$ $(\mathit{hash} \notin \mathit{messages})$ $\textbf{then}$
        $\mathit{messages} := \mathit{messages} \cup \{\mathit{hash}\}$;
        $\mathit{seenPerm} := \mathit{seenPerm} \vee Fperm$;
        $\textbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm,$
                    $\textbf{max}\{\mathit{firstHop}, maxHop\}, \mathit{FDsrc}, FDsink, Fid\rangle$;
    $\textbf{fi}$;
```

Figure 35: Source Location Privacy Algorithm - Fake Source

```
process $j$ - If type is Source
variables
    % The number of messages sent
    $count$: int init 0;

    % Distance from source to sink
    $\Delta_{sink-source}$: int init $\perp$;

    % How often messages are sent
    $period$: timer init $P_{source}$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;

constants
    % The source period an algorithm parameter
    $P_{source}$: time;

actions
    % Sending Normal Messages
    $SourceSendNormal$:: timeout($period$) $\rightarrow$

    % Receiving Away Message
    $SourceReceiveAway$:: rcv$\langle Away, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$

    % Receiving Fake Message
    $SourceReceiveFake$:: rcv$\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
```

Figure 36: Source Location Privacy Algorithm - Source

```
    % Sending Normal Messages
    $SourceSendNormal$:: timeout($period$) $\rightarrow$
        $count := count + 1$;
        BCAST$\langle Normal$, hash$(Normal), \Delta_{sink-source}, 0, count, firstHop\rangle$;
        set($period$, $P_{source}$);

    % Receiving Away Message
    $SourceReceiveAway$:: rcv$\langle Away, hash, ssd, Dsink, maxHop, algorithm\rangle \rightarrow$
        if $(algorithm \neq \perp)$ then
            $\mathcal{A} := algorithm$;
        fi;
        $\Delta_{sink-source} :=$ min$_{\perp}\{\Delta_{sink-source}, ssd\}$;
        if $(hash \notin messages)$ then
            $messages := messages \cup \{hash\}$;
            $\Delta_{sink-source} :=$ min$_{\perp}\{\Delta_{sink-source}, Dsink + 1\}$;
            BCAST$\langle Away, hash, \Delta_{sink-source}, Dsink + 1, maxHop, \mathcal{A}\rangle$ sense repeat(2);
        fi;

    % Receiving Fake Message
    $SourceReceiveFake$:: rcv$\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
        $\Delta_{sink-source} :=$ min$_{\perp}\{\Delta_{sink-source}, ssd\}$;
```

Figure 37: Source Location Privacy Algorithm - Source

```
process $j$ - If type is Normal
variables
    % Messages seen
    $\textit{messages}$: set of int init $\emptyset$;

    % The hop count of this node in the first wave of normal messages
    $\textit{firstHop}$: int init $\perp$;

    % The flag that indicates if this is a Permanent Fake Source
    $\textit{isPerm}$: boolean init $False$;

    % Indicates if this node has received a Fake message from a Permanent Fake Source
    $\textit{seenPerm}$: boolean init $False$;

    % Distance to the source, to the sink, between the sink and source
    $\Delta_{source}, \Delta_{sink}, \Delta_{sink-source}$: int, int, int init $\perp, \perp, \perp$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;

constants
    % Duration of a Temporary Fake Source, an algorithm parameter
    $\tau$: timer;

actions
    % Receiving Normal Message
    $\textit{NormalReceiveNormal}$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$

    % Receiving Fake Message
    $\textit{NormalReceiveFake}$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$

    % Receiving Away Message
    $\textit{NormalReceiveAway}$:: $\mathbf{rcv}\langle Away, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$

    % Receiving Choose Message
    $\textit{NormalReceiveChoose}$:: $\mathbf{rcv}\langle Choose, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$
```

Figure 38: Source Location Privacy Algorithm - Normal

```
    % Receiving Normal Message
    $\textit{NormalReceiveNormal}$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$
        if ($\textit{firstHop} = \perp \vee maxHop - 1 > \textit{firstHop}$) then
            $\textit{isPerm} := False$;
        fi;
        $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
        if ($hash \notin messages$) then
            $messages := messages \cup \{hash\}$;
            if (count = 1) then
                $\textit{firstHop}, \textit{isPerm} := hop + 1, True$;
            fi;
            $\Delta_{source} := \mathbf{min}_{\perp}\{\Delta_{source}, hop + 1\}$;
            $\mathbf{BCAST}\langle Normal, hash, \Delta_{sink-source}, hop + 1, count, \mathbf{max}\{\textit{firstHop}, maxHop\}\rangle$;
        fi;
```

Figure 39: Source Location Privacy Algorithm - Normal

```
% Receiving Fake Message
$NormalReceiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, FDsrc, FDsink, Fid\rangle \rightarrow$
    **if** $(firstHop = \bot \vee maxHop - 1 > firstHop)$ **then**
        $isPerm := False$;
    **fi**;
    $\Delta_{sink-source} := \mathbf{min}_{\bot}\{\Delta_{sink-source}, ssd\}$;
    **if** $(hash \notin messages)$ **then**
        $messages := messages \cup \{hash\}$;
        $seenPerm := seenPerm \vee Fperm$;
        $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm,$
                $\mathbf{max}\{firstHop, maxHop\}, FDsrc, FDsink, Fid\rangle$;
    **fi**;
```

Figure 40: Source Location Privacy Algorithm - Normal

```
% Receiving Away Message
$NormalReceiveAway$:: $\mathbf{rcv}\langle Away, hash, ssd, Dsink, maxHop, algorithm\rangle \rightarrow$
    **if** $(firstHop = \bot \vee maxHop - 1 > firstHop)$ **then**
        $isPerm := False$;
    **fi**;
    **if** $(algorithm \neq \bot)$ **then**
        $\mathcal{A} := algorithm$;
    **fi**;
    $\Delta_{sink-source} := \mathbf{min}_{\bot}\{\Delta_{sink-source}, ssd\}$;
    **if** $(hash \notin messages)$ **then**
        $messages := messages \cup \{hash\}$;
        $\Delta_{sink} := \mathbf{min}_{\bot}\{\Delta_{sink}, Dsink\}$;
        **if** $(Dsink = 0)$ **then**
            **BecomeFakeSource**(duration=$\tau$);
            % Create a new hash otherwise future choose messages will fail to work
            $\mathbf{BCAST}\langle Away, \mathbf{hash}(Away), \Delta_{sink-source}, Dsink + 1,$
                    $\mathbf{max}\{firstHop, maxHop\}, \mathcal{A}\rangle$ **sense repeat**(2);
        **else**
            $\mathbf{BCAST}\langle Away, hash, \Delta_{sink-source}, Dsink + 1,$
                    $\mathbf{max}\{firstHop, maxHop\}, \mathcal{A}\rangle$ **sense repeat**(2);
        **fi**;
    **fi**;
```

Figure 41: Source Location Privacy Algorithm - Normal

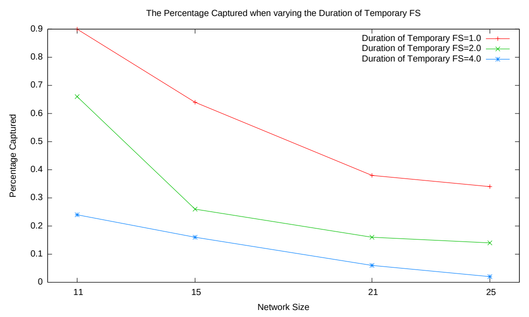
```
% Receiving Choose Message when $\mathcal{A}$ is Further Specialisation
$\mathit{NormalReceiveChoose}$:: $\mathbf{rcv}\langle Choose, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$
    if $(\mathit{firstHop} = \perp \vee maxHop - 1 > \mathit{firstHop})$ then
        $\mathit{isPerm} := False$;
    fi;
    if $(\mathit{algorithm} \neq \perp)$ then
        $\mathcal{A} := algorithm$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(\mathit{hash} \notin \mathit{messages} \wedge \neg \mathit{seenPerm} \wedge$
            $\neg(\Delta_{sink-source} \neq \perp \wedge \Delta_{source} \leq \frac{1}{2}\Delta_{sink-source} - 1))$ then
        $\mathit{messages} := \mathit{messages} \cup \{hash\}$;
        if $(\mathit{isPerm})$ then
            BecomeFakeSource(duration=$\infty$);
        else
            BecomeFakeSource(duration=$\tau$);
        fi;
    fi;

% Receiving Choose Message when $\mathcal{A}$ is Generic Specialisation
$\mathit{NormalReceiveChoose}$:: $\mathbf{rcv}\langle Choose, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$
    if $(\mathit{firstHop} = \perp \vee maxHop - 1 > \mathit{firstHop})$ then
        $\mathit{isPerm} := False$;
    fi;
    if $(\mathit{algorithm} \neq \perp)$ then
        $\mathcal{A} := algorithm$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(\mathit{hash} \notin \mathit{messages} \wedge \neg(\Delta_{sink-source} \neq \perp \wedge \Delta_{source} \leq \frac{3}{4}\Delta_{sink-source}))$ then
        $\mathit{messages} := \mathit{messages} \cup \{hash\}$;
        if $(\mathit{isPerm})$ then
            BecomeFakeSource(duration=$\infty$);
        else
            BecomeFakeSource(duration=$\tau$);
        fi;
    fi;
```

Figure 42: Source Location Privacy Algorithm - Normal

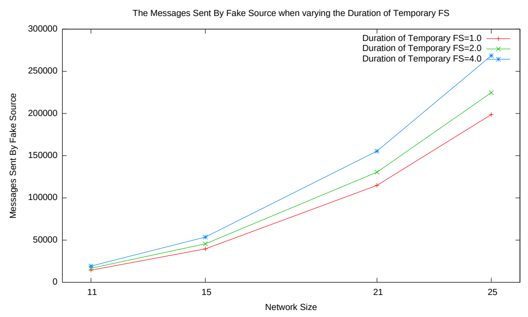
C Selected Template Graphs

C.1 Effect of Varying the Duration of Temporary Fake Sources

SourceCorner



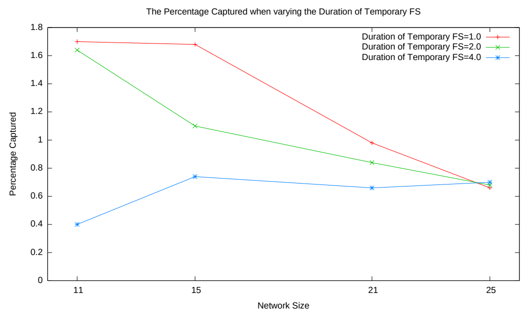
(a) Capture Rate



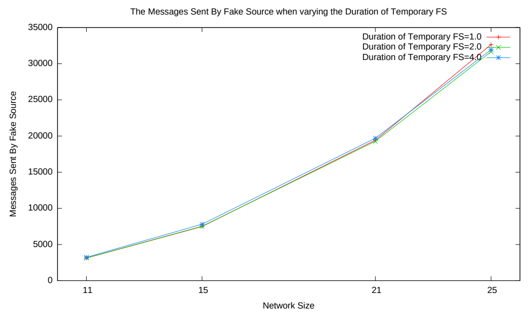
(b) Fake Messages Sent

Figure 43: Varying the Duration of Temporary Fake Sources for the SourceCorner configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.25 sec, Pr(TFS)=100%

SinkCorner



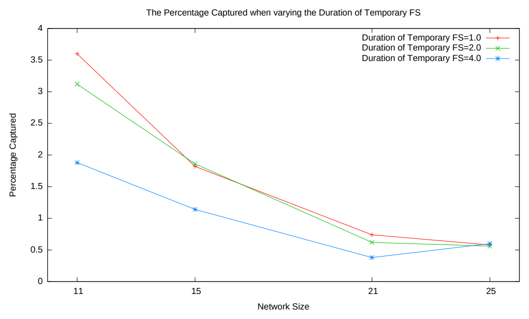
(a) Capture Rate



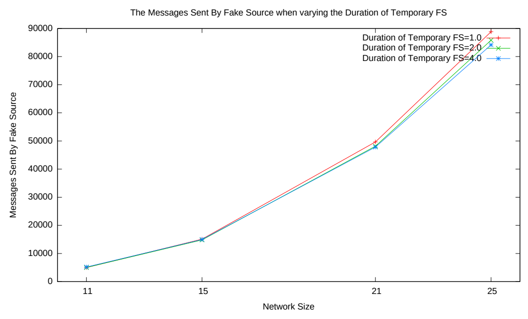
(b) Fake Messages Sent

Figure 44: Varying the Duration of Temporary Fake Sources for the SinkCorner configuration. Parameters: Source Period=0.25 sec, Fake Source Period=0.0625 sec, Pr(TFS)=100%

FurtherSinkCorner



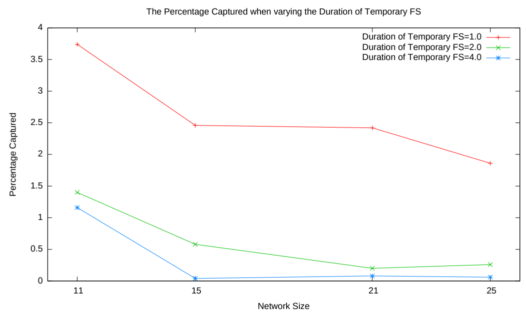
(a) Capture Rate



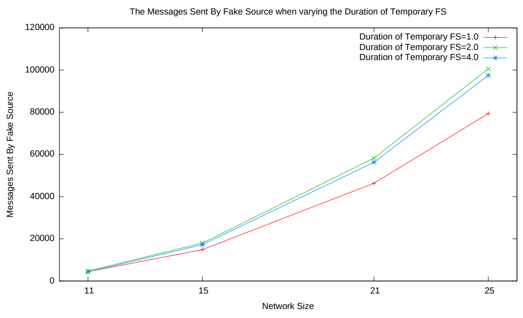
(b) Fake Messages Sent

Figure 45: Varying the Duration of Temporary Fake Sources for the FurtherSinkCorner configuration. Parameters: Source Period=0.125 sec, Fake Source Period=0.0625 sec, Pr(TFS)=100%

Generic1



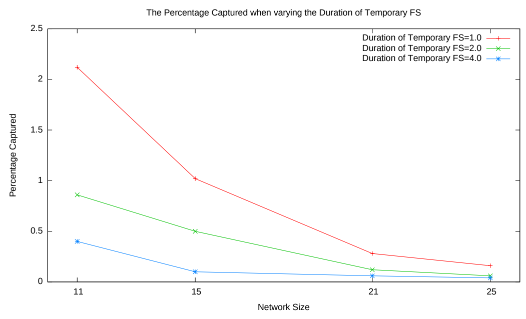
(a) Capture Rate



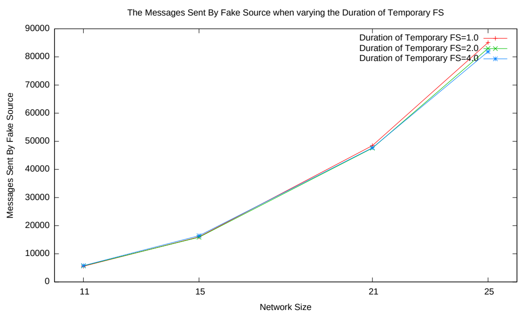
(b) Fake Messages Sent

Figure 46: Varying the Duration of Temporary Fake Sources for the Generic1 configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.5 sec, Pr(TFS)=90%

Generic2



(a) Capture Rate

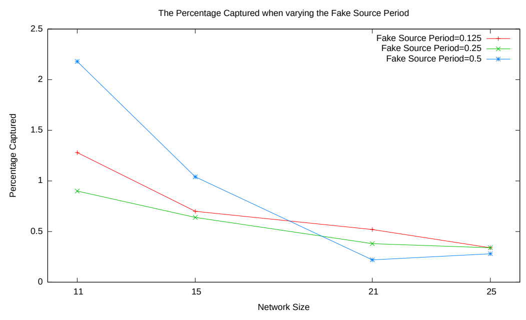


(b) Fake Messages Sent

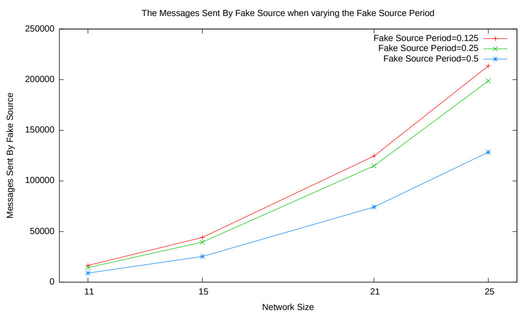
Figure 47: Varying the Duration of Temporary Fake Sources for the Generic2 configuration. Parameters: Source Period=0.25 sec, Fake Source Period=0.0625 sec, Pr(TFS)=100%

C.2 Effect of Varying the Rate of Fake Sources

SourceCorner



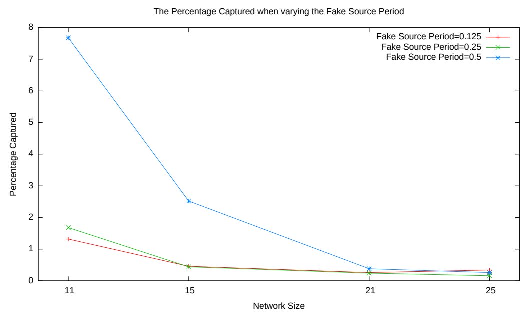
(a) Capture Rate



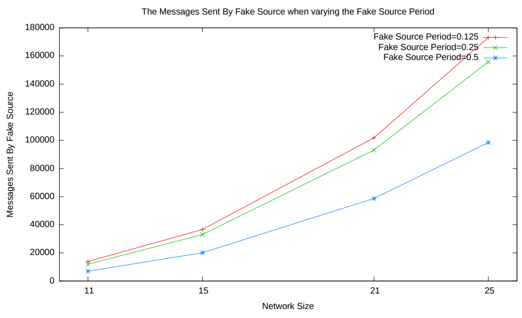
(b) Fake Messages Sent

Figure 48: Varying the Fake Source Rate for the SourceCorner configuration. Parameters: Source Period=1.0 sec, Pr(TFS)=100%, Duration=1.0 sec

SinkCorner



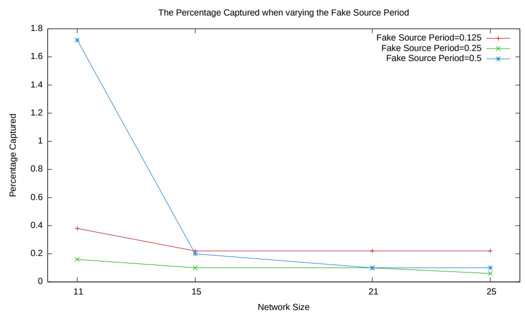
(a) Capture Rate



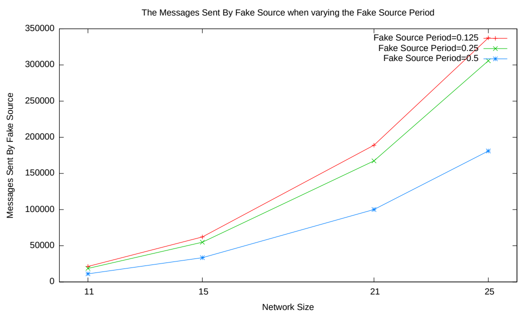
(b) Fake Messages Sent

Figure 49: Varying the Fake Source Rate for the SinkCorner configuration. Parameters: Source Period=1.0 sec, Pr(TFS)=100%, Duration=2.0 sec

FurtherSinkCorner



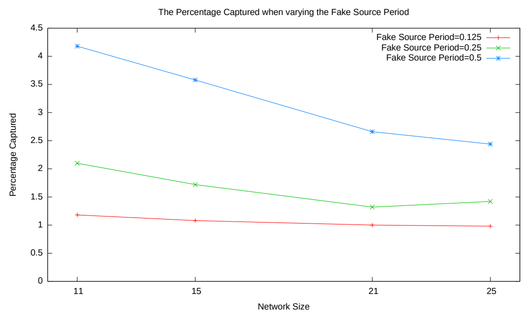
(a) Capture Rate



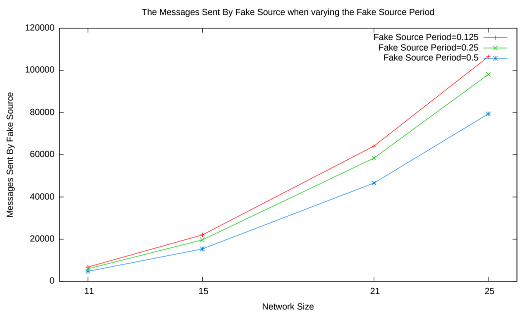
(b) Fake Messages Sent

Figure 50: Varying the Fake Source Rate for the FurtherSinkCorner configuration. Parameters: Source Period=1.0 sec, Pr(TFS)=100%, Duration=4.0 sec

Generic1



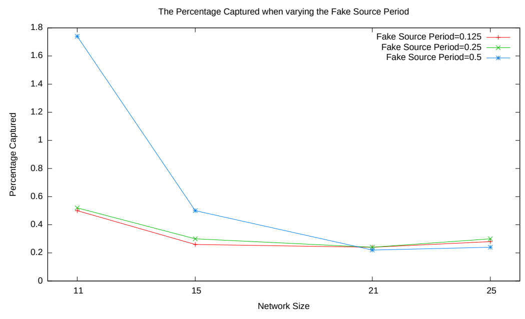
(a) Capture Rate



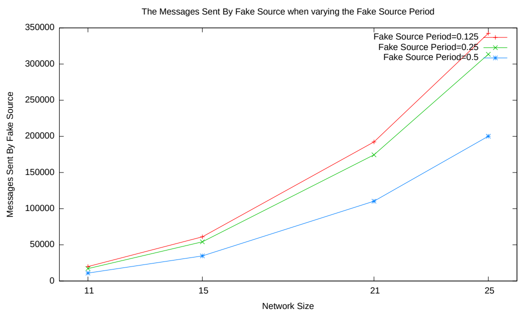
(b) Fake Messages Sent

Figure 51: Varying the Fake Source Rate for the Generic1 configuration. Parameters: Source Period=1.0 sec, Pr(TFS)=100%, Duration=1.0 sec

Generic2



(a) Capture Rate

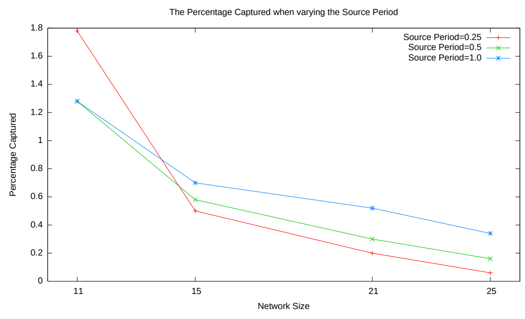


(b) Fake Messages Sent

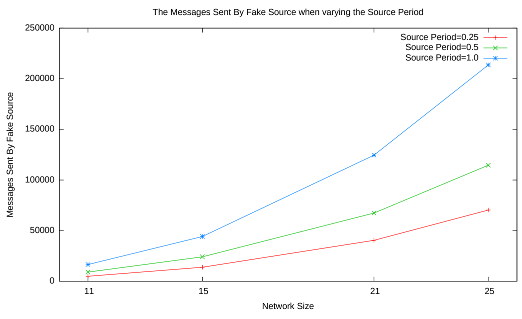
Figure 52: Varying the Fake Source Rate for the Generic2 configuration. Parameters: Source Period=1.0 sec, Pr(TFS)=100%, Duration=2.0 sec

C.3 Effect of Varying the Rate of Normal Source

SourceCorner



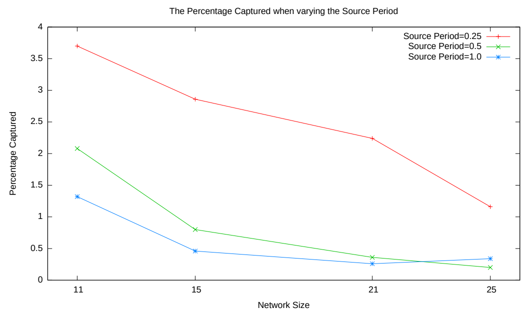
(a) Capture Rate



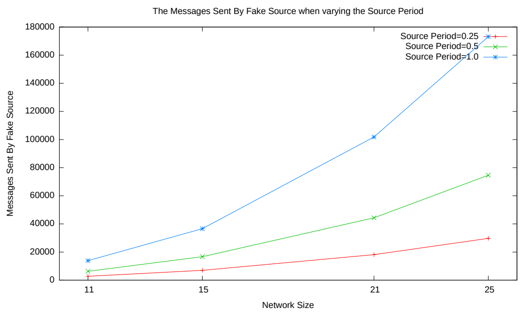
(b) Fake Messages Sent

Figure 53: Varying the Source Rate for the SourceCorner configuration. Parameters: Fake Source Period=0.125 sec, Pr(TFS)=100%, Duration=1.0 sec

SinkCorner



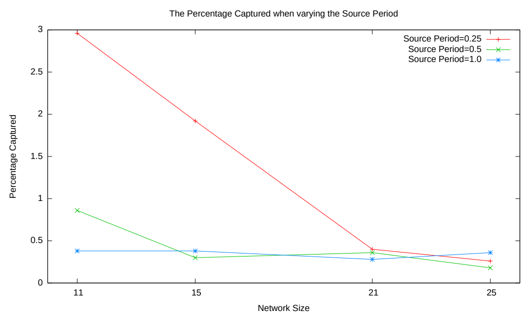
(a) Capture Rate



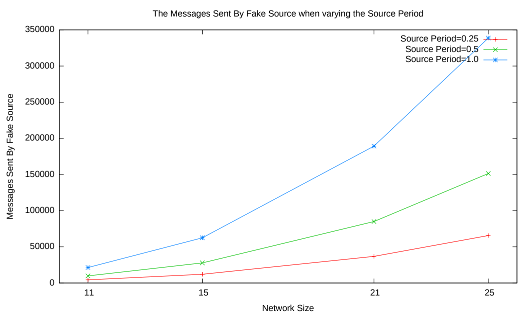
(b) Fake Messages Sent

Figure 54: Varying the Source Rate for the SinkCorner configuration. Parameters: Fake Source Period=0.125 sec, Pr(TFS)=100%, Duration=2.0 sec

FurtherSinkCorner



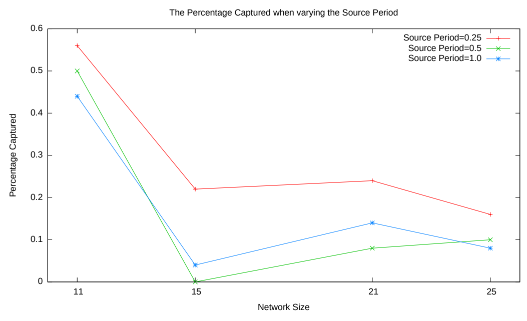
(a) Capture Rate



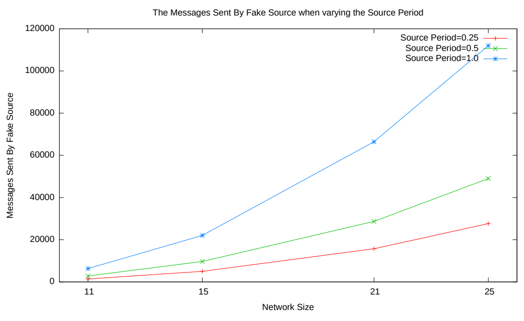
(b) Fake Messages Sent

Figure 55: Varying the Source Rate for the FurtherSinkCorner configuration. Parameters: Fake Source Period=0.125 sec, Pr(TFS)=100%, Duration=2.0 sec

Generic1



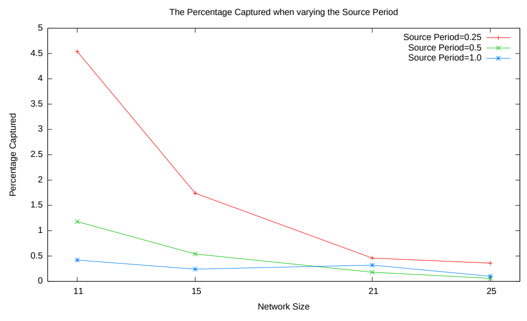
(a) Capture Rate



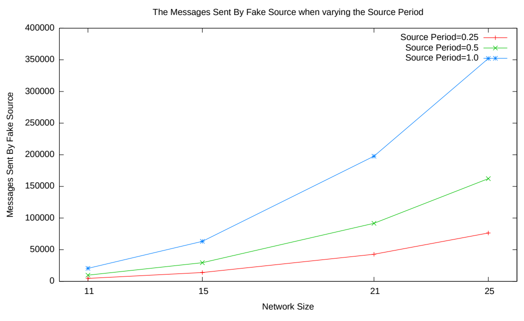
(b) Fake Messages Sent

Figure 56: Varying the Source Rate for the Generic1 configuration. Parameters: Fake Source Period=0.125 sec, Pr(TFS)=90%, Duration=4.0 sec

Generic2



(a) Capture Rate

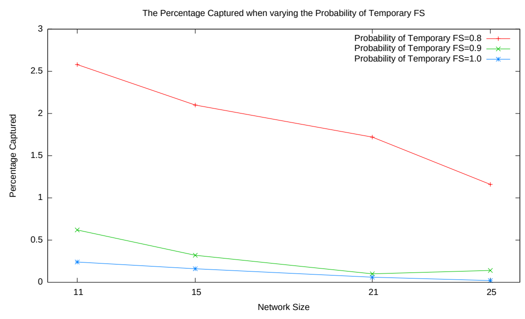


(b) Fake Messages Sent

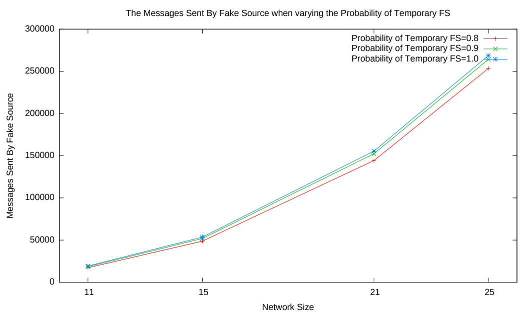
Figure 57: Varying the Source Rate for the Generic2 configuration. Parameters: Fake Source Period=0.125 sec, Pr(TFS)=100%, Duration=1.0 sec

C.4 Effect of Varying the Probability of Temporary Fake Sources

SourceCorner



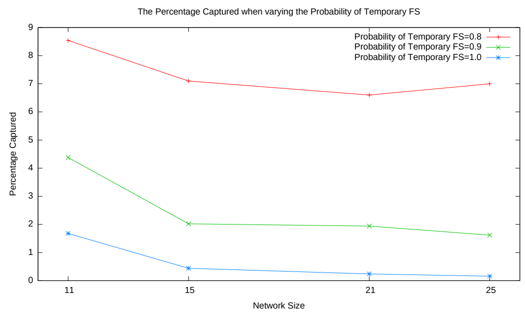
(a) Capture Rate



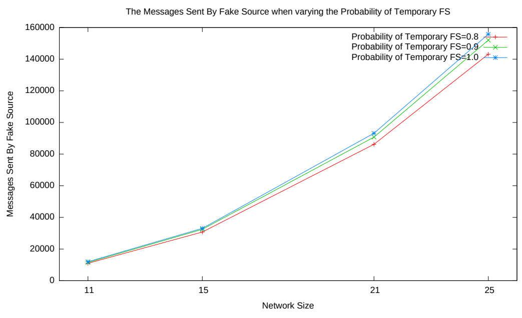
(b) Fake Messages Sent

Figure 58: Varying the Probability of Temporary Fake Sources for the SourceCorner configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.25 sec, Duration=4.0 sec

SinkCorner



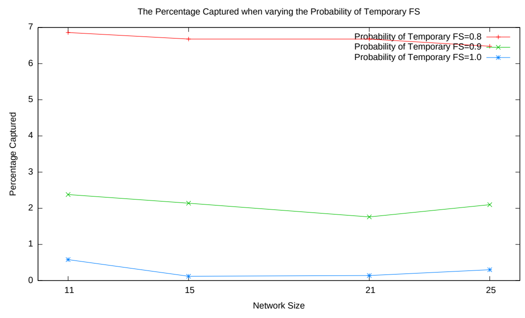
(a) Capture Rate



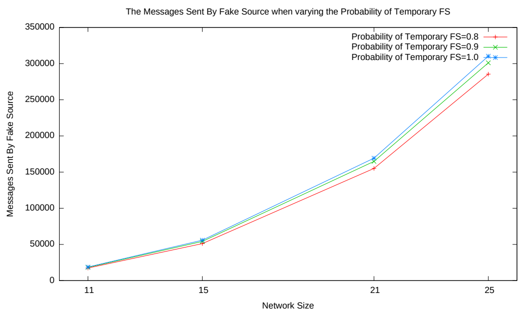
(b) Fake Messages Sent

Figure 59: Varying the Probability of Temporary Fake Sources for the SinkCorner configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.25 sec, Duration=2.0 sec

FurtherSinkCorner



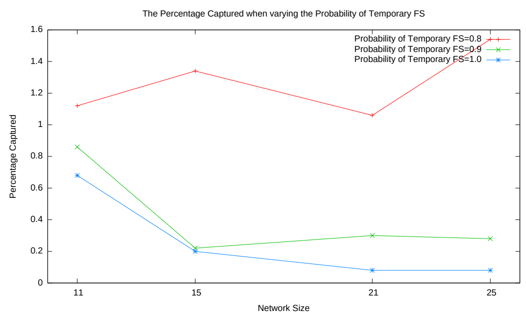
(a) Capture Rate



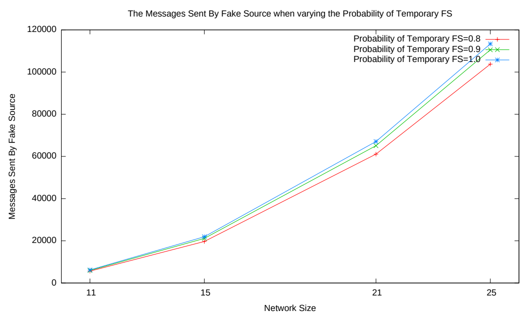
(b) Fake Messages Sent

Figure 60: Varying the Probability of Temporary Fake Sources for the FurtherSinkCorner configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.25 sec, Duration=1.0 sec

Generic1



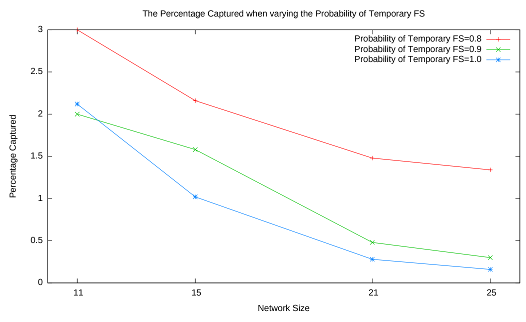
(a) Capture Rate



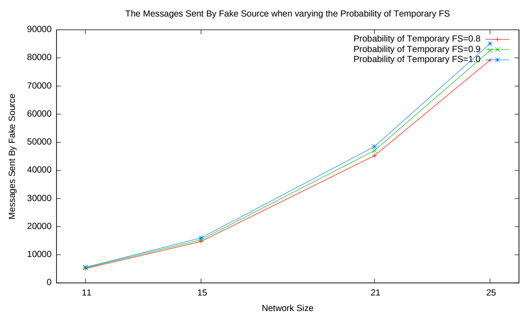
(b) Fake Messages Sent

Figure 61: Varying the Probability of Temporary Fake Sources for the Generic1 configuration. Parameters: Source Period=1.0 sec, Fake Source Period=0.25 sec, Duration=2.0 sec

Generic2



(a) Capture Rate



(b) Fake Messages Sent

Figure 62: Varying the Probability of Temporary Fake Sources for the Generic2 configuration. Parameters: Source Period=0.25 sec, Fake Source Period=0.0625 sec, Duration=1.0 sec

D Template Results Tables

Heading Meanings

P_{src}	The Source period
P_{fs}	The Fake Source period
Pr(TFS)	The probability that a temporary fake source is created
Pr(PFS)	The probability that a permanent fake source is created
Duration	The duration of a temporary fake source

D.1 SourceCorner

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	10.26	4630	7.9
0.125	0.0625	80	100	2	6.58	4818	5.3
0.125	0.0625	80	100	4	2.52	4873	3.3
0.125	0.0625	90	100	4	1.72	5086	2.6
0.125	0.0625	100	100	4	1.24	5256	2.1
0.25	0.125	80	100	1	6.36	4238	12.6
0.25	0.125	80	100	4	1.56	4297	6.3
0.25	0.125	90	100	4	1.08	4453	5.2
0.25	0.125	100	100	4	1.02	4572	4.6
0.25	0.0625	90	100	4	0.60	4973	3.9
0.25	0.0625	100	100	4	0.30	5103	3.5
0.5	0.25	80	100	1	7.60	6911	23.1
0.5	0.25	90	100	1	2.84	7566	19.5
0.5	0.25	100	100	1	1.70	7807	18.3
0.5	0.25	100	100	2	0.40	9164	9.7
0.5	0.25	80	100	4	1.64	8047	11.3
0.5	0.25	90	100	4	0.84	8452	9.4
0.5	0.25	100	100	4	0.54	8755	8.2
0.5	0.125	100	100	4	0.38	9774	5.6
1.0	0.5	80	100	1	7.82	8201	46.4
1.0	0.5	90	100	1	3.00	8854	44.9
1.0	0.5	100	100	1	2.18	9014	44.4
1.0	0.5	90	100	2	1.50	11900	32.8
1.0	0.5	100	100	2	1.30	12132	32.3
1.0	0.5	90	100	4	0.60	15083	19.8
1.0	0.25	100	100	1	0.90	14323	26.2
1.0	0.25	100	100	4	0.24	19184	10.1
1.0	0.125	100	100	2	0.58	18977	14.6

Table 17: Template: Results for networks of the configuration: SourceCorner and size 11

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	8.52	13164	6.8
0.125	0.0625	80	100	4	2.40	13407	2.9
0.125	0.0625	90	100	4	1.06	14117	1.9
0.125	0.0625	100	100	4	0.50	14607	1.5
0.25	0.125	80	100	4	1.40	11684	5.7
0.25	0.125	90	100	4	0.52	12212	4.2
0.25	0.125	100	100	4	0.16	12586	3.4
0.25	0.0625	100	100	4	0.08	13557	2.5
0.5	0.25	80	100	1	5.54	19809	19.0
0.5	0.25	90	100	1	1.76	21179	16.3
0.5	0.25	100	100	1	0.64	21402	16.4

0.5	0.25	90	100	4	0.30	23929	6.6
0.5	0.25	100	100	4	0.12	24712	5.6
0.5	0.125	100	100	4	0.06	26739	3.7
1.0	0.5	80	100	1	8.16	23926	43.7
1.0	0.5	100	100	1	1.04	25329	43.0
1.0	0.5	100	100	2	0.30	33952	30.6
1.0	0.5	90	100	4	0.12	45036	13.5
1.0	0.125	100	100	4	0.08	58154	5.0

Table 18: Template: Results for networks of the configuration: SourceCorner and size 15

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	7.96	41060	5.7
0.125	0.0625	80	100	4	2.04	41111	2.3
0.125	0.0625	90	100	4	0.48	43505	1.3
0.125	0.0625	100	100	4	0.12	44986	0.9
0.25	0.125	80	100	4	1.34	34103	4.8
0.25	0.125	90	100	4	0.28	35759	3.1
0.25	0.125	100	100	4	0.08	36764	2.4
0.25	0.0625	100	100	4	0.04	38634	1.7
0.5	0.25	80	100	1	5.40	58750	16.2
0.5	0.25	100	100	1	0.20	61543	14.7
0.5	0.25	90	100	4	0.06	70920	4.6
0.5	0.25	100	100	4	0.02	72814	3.9
1.0	0.5	80	100	1	8.08	72521	40.6
1.0	0.5	100	100	1	0.22	74182	40.0
1.0	0.5	100	100	2	0.16	99018	27.7
1.0	0.5	90	100	4	0.12	136823	9.1
1.0	0.5	100	100	4	0.00	139509	8.4

Table 19: Template: Results for networks of the configuration: SourceCorner and size 21

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	1.84	72777	2.0
0.125	0.0625	90	100	4	0.48	76789	1.1
0.125	0.0625	100	100	4	0.06	79446	0.7
0.25	0.125	100	100	2	0.00	68706	2.0
0.25	0.125	80	100	4	1.06	60211	3.9
0.25	0.0625	100	100	4	0.02	67104	1.4
0.25	0.125	90	100	4	0.18	62966	2.5
0.25	0.125	100	100	4	0.08	64416	2.0
0.5	0.25	90	100	4	0.02	123612	3.7
0.5	0.25	100	100	4	0.00	126404	3.2
0.5	0.25	80	100	1	4.74	103258	14.3
0.5	0.25	100	100	1	0.10	105858	14.0
1.0	0.5	100	100	1	0.28	128374	39.7
1.0	0.5	100	100	2	0.08	171686	27.0
1.0	0.5	90	100	4	0.00	241306	7.2

Table 20: Template: Results for networks of the configuration: SourceCorner and size 25

D.2 SinkCorner

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	12.28	2597	6.1
0.125	0.0625	90	100	1	8.84	2760	4.9
0.125	0.0625	100	100	1	5.62	2943	3.9

0.125	0.0625	80	100	2	10.24	2651	5.7
0.125	0.0625	90	100	2	5.88	2855	4.3
0.125	0.0625	100	100	2	3.50	3008	3.4
0.125	0.0625	80	100	4	7.78	2767	4.6
0.125	0.0625	90	100	4	3.60	2971	3.3
0.125	0.0625	100	100	4	1.78	3113	2.5
0.25	0.125	80	100	1	11.62	2446	11.9
0.25	0.125	80	100	2	9.26	2502	10.7
0.25	0.125	90	100	2	4.98	2659	8.9
0.25	0.125	80	100	4	6.10	2603	8.4
0.25	0.125	90	100	4	3.12	2756	6.5
0.25	0.125	100	100	4	1.92	2860	5.4
0.25	0.0625	100	100	1	1.70	3126	5.9
0.25	0.0625	100	100	2	1.64	3136	5.5
0.25	0.0625	100	100	4	0.40	3232	4.0
0.5	0.25	80	100	1	9.68	4885	18.7
0.5	0.25	90	100	1	5.12	5180	17.0
0.5	0.25	100	100	1	3.44	5324	16.4
0.5	0.25	90	100	2	5.54	5159	17.0
0.5	0.25	100	100	2	3.40	5339	16.1
0.5	0.25	80	100	4	7.90	5025	16.8
0.5	0.25	90	100	4	4.00	5307	14.9
0.5	0.25	100	100	4	2.84	5461	14.0
0.5	0.125	100	100	1	2.00	6331	10.2
0.5	0.125	100	100	2	2.08	6299	10.0
0.5	0.125	100	100	4	1.52	6434	8.2
1.0	0.5	80	100	1	12.32	6542	31.7
1.0	0.5	90	100	1	8.24	6907	30.6
1.0	0.5	100	100	1	6.62	7177	29.7
1.0	0.5	80	100	2	13.08	6329	32.2
1.0	0.5	90	100	2	9.02	6759	30.9
1.0	0.5	100	100	2	7.68	6955	29.8
1.0	0.25	90	100	1	3.90	11669	18.4
1.0	0.25	90	100	2	4.38	11478	18.5
1.0	0.25	100	100	2	1.68	11953	17.6
1.0	0.25	90	100	4	4.28	11644	17.4
1.0	0.125	100	100	1	1.14	13949	11.1
1.0	0.125	100	100	2	1.32	13848	11.2
1.0	0.125	100	100	4	1.30	13897	10.3

Table 21: Template: Results for networks of the configuration: SinkCorner and size 11

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	10.68	6918	5.4
0.125	0.0625	90	100	1	5.50	7529	3.9
0.125	0.0625	100	100	1	3.34	7843	3.1
0.125	0.0625	80	100	2	9.28	7028	5.2
0.125	0.0625	90	100	2	4.90	7585	3.8
0.125	0.0625	100	100	2	3.18	7849	3.1
0.125	0.0625	80	100	4	7.94	7318	4.5
0.125	0.0625	90	100	4	3.86	7783	3.4
0.125	0.0625	100	100	4	1.52	8170	2.5
0.25	0.125	80	100	1	9.46	6316	11.2
0.25	0.125	90	100	1	5.46	6694	9.3
0.25	0.125	100	100	1	4.16	6886	8.4
0.25	0.125	80	100	2	8.92	6396	10.7
0.25	0.125	90	100	2	4.54	6753	8.7
0.25	0.125	100	100	2	2.86	6982	7.8
0.25	0.125	80	100	4	6.42	6633	8.8
0.25	0.125	90	100	4	2.68	7094	6.7

0.25	0.125	100	100	4	1.22	7281	5.9
0.25	0.0625	100	100	2	1.10	7530	5.0
0.25	0.0625	100	100	4	0.74	7806	4.0
0.5	0.25	80	100	1	7.94	13755	16.4
0.5	0.25	100	100	1	1.30	14947	13.7
0.5	0.25	80	100	2	8.02	13709	16.5
0.5	0.25	90	100	2	3.08	14558	14.6
0.5	0.25	100	100	2	1.38	14866	14.1
0.5	0.25	80	100	4	6.42	14083	15.1
0.5	0.25	90	100	4	2.08	14821	13.3
0.5	0.25	100	100	4	0.96	15088	12.9
0.5	0.125	100	100	2	0.80	16678	9.0
0.5	0.125	100	100	4	0.68	16779	8.3
1.0	0.5	80	100	1	7.90	18713	38.4
1.0	0.5	100	100	1	2.10	20793	35.9
1.0	0.5	80	100	2	8.08	18330	39.3
1.0	0.5	90	100	2	3.56	19484	38.2
1.0	0.5	100	100	2	2.52	20062	37.5
1.0	0.25	90	100	2	2.02	32462	16.0
1.0	0.25	100	100	2	0.44	33041	15.8
1.0	0.25	100	100	4	0.38	33252	15.1
1.0	0.125	100	100	4	0.34	36699	9.8

Table 22: Template: Results for networks of the configuration: SinkCorner and size 15

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	8.16	19528	4.8
0.125	0.0625	90	100	2	3.78	20841	3.5
0.125	0.0625	100	100	2	1.96	21823	2.7
0.125	0.0625	80	100	4	7.96	19619	4.6
0.125	0.0625	100	100	4	1.64	22128	2.4
0.25	0.125	90	100	1	3.98	17694	9.1
0.25	0.125	80	100	2	7.98	16541	11.1
0.25	0.125	90	100	2	4.08	17622	9.2
0.25	0.125	100	100	2	2.24	18174	8.3
0.25	0.125	80	100	4	6.50	17057	9.8
0.25	0.125	90	100	4	2.60	18061	8.0
0.25	0.125	100	100	4	1.56	18711	6.8
0.25	0.0625	100	100	2	0.84	19282	4.7
0.25	0.0625	100	100	4	0.66	19708	4.1
0.5	0.25	90	100	1	2.30	39438	17.0
0.5	0.25	100	100	1	0.44	40453	16.1
0.5	0.25	80	100	2	6.64	37114	19.5
0.5	0.25	90	100	2	2.44	39250	17.1
0.5	0.25	100	100	2	0.52	40154	16.6
0.5	0.25	90	100	4	1.90	39612	16.7
0.5	0.25	100	100	4	0.38	40510	15.8
0.5	0.125	100	100	2	0.36	44353	10.0
0.5	0.125	100	100	4	0.34	44590	9.6
1.0	0.5	80	100	2	6.54	53398	37.1
1.0	0.5	90	100	2	1.76	56838	35.9
1.0	0.5	100	100	2	0.38	58668	35.1
1.0	0.5	80	100	4	5.66	54024	36.7
1.0	0.5	90	100	4	1.70	57112	35.8
1.0	0.5	100	100	4	0.40	58474	35.2
1.0	0.25	100	100	4	0.12	92928	19.2

Table 23: Template: Results for networks of the configuration: SinkCorner and size 21

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	7.90	33355	4.6
0.125	0.0625	90	100	2	3.30	35859	3.2
0.125	0.0625	100	100	2	1.26	37241	2.5
0.125	0.0625	80	100	4	8.64	33229	4.7
0.125	0.0625	90	100	4	3.32	35754	3.2
0.125	0.0625	100	100	4	0.80	37442	2.4
0.25	0.125	100	100	2	1.16	29708	8.2
0.25	0.125	80	100	4	6.34	27487	10.3
0.25	0.125	90	100	4	2.26	29108	8.4
0.25	0.125	100	100	4	0.80	30231	7.3
0.25	0.0625	100	100	1	0.66	32642	4.2
0.25	0.125	90	100	1	3.12	29028	9.3
0.25	0.125	80	100	2	7.84	26660	11.6
0.25	0.125	90	100	2	3.44	28311	9.5
0.25	0.0625	100	100	2	0.68	31664	4.5
0.5	0.25	80	100	1	6.16	61432	20.9
0.5	0.25	80	100	2	6.94	60579	21.5
0.5	0.25	90	100	2	2.00	64045	19.7
0.5	0.25	100	100	2	0.36	65620	18.8
0.5	0.25	80	100	4	6.28	61120	21.0
0.5	0.25	90	100	4	1.54	64694	19.1
0.5	0.25	100	100	4	0.20	66294	18.2
0.5	0.125	100	100	4	0.12	74728	9.8
1.0	0.5	100	100	1	0.14	102280	36.8
1.0	0.5	80	100	2	6.74	89495	40.0
1.0	0.5	90	100	2	1.96	95069	38.8
1.0	0.5	80	100	4	5.28	90726	39.7
1.0	0.5	90	100	4	1.36	95620	38.7
1.0	0.5	100	100	4	0.18	98269	38.0
1.0	0.25	100	100	4	0.04	155748	22.2

Table 24: Template: Results for networks of the configuration: SinkCorner and size 25

D.3 FurtherSinkCorner

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	9.00	4514	5.2
0.125	0.0625	90	100	1	5.08	4814	3.8
0.125	0.0625	80	100	2	9.28	4484	5.3
0.125	0.0625	90	100	2	5.10	4805	3.8
0.125	0.0625	100	100	2	3.12	5010	3.1
0.125	0.0625	80	100	4	8.66	4618	4.7
0.125	0.0625	90	100	4	4.10	4959	3.3
0.125	0.0625	100	100	4	1.88	5186	2.4
0.25	0.125	80	100	1	11.06	3886	11.2
0.25	0.125	90	100	1	5.80	4195	8.7
0.25	0.125	100	100	1	3.76	4312	8.0
0.25	0.125	80	100	2	10.06	3939	10.5
0.25	0.125	90	100	2	4.88	4221	8.4
0.25	0.125	100	100	2	2.96	4372	7.3
0.25	0.125	80	100	4	6.18	4161	8.3
0.25	0.125	100	100	4	1.36	4522	5.6
0.25	0.0625	100	100	2	1.30	4860	5.0
0.25	0.0625	100	100	4	0.66	4996	4.0
0.5	0.25	80	100	1	7.38	7972	14.5
0.5	0.25	90	100	1	3.12	8323	13.3
0.5	0.25	80	100	2	7.90	7825	15.0

0.5	0.25	90	100	2	3.18	8307	13.1
0.5	0.25	100	100	2	1.14	8529	12.5
0.5	0.25	80	100	4	6.98	8039	13.6
0.5	0.25	90	100	4	2.36	8504	11.7
0.5	0.25	100	100	4	0.88	8683	11.3
0.5	0.125	100	100	2	0.86	9800	8.7
0.5	0.125	100	100	4	0.60	9990	7.4
1.0	0.5	100	100	1	1.28	11344	33.9
1.0	0.5	80	100	2	7.96	9988	37.5
1.0	0.5	90	100	2	3.16	10686	35.9
1.0	0.5	100	100	2	1.36	11014	35.6
1.0	0.5	80	100	4	7.00	10221	36.7
1.0	0.5	90	100	4	3.10	10785	35.3
1.0	0.25	100	100	2	0.44	18591	13.3
1.0	0.25	100	100	4	0.16	18691	12.8

Table 25: Template: Results for networks of the configuration: FurtherSinkCorner and size 11

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	8.88	13143	4.9
0.125	0.0625	90	100	2	3.88	14147	3.4
0.125	0.0625	100	100	2	1.86	14818	2.6
0.125	0.0625	80	100	4	7.82	13399	4.4
0.125	0.0625	90	100	4	3.34	14359	3.1
0.125	0.0625	100	100	4	1.14	14959	2.5
0.25	0.125	80	100	2	8.46	10891	11.2
0.25	0.125	90	100	2	3.54	11612	9.1
0.25	0.125	100	100	2	1.92	12024	8.1
0.25	0.125	80	100	4	7.48	11127	10.2
0.25	0.125	90	100	4	2.86	11819	8.2
0.25	0.125	100	100	4	1.20	12353	7.0
0.25	0.0625	100	100	2	0.70	13239	4.6
0.25	0.0625	100	100	4	0.52	13393	4.1
0.5	0.25	80	100	2	6.46	22493	18.3
0.5	0.25	90	100	2	1.96	23624	16.8
0.5	0.25	100	100	2	0.32	24129	16.1
0.5	0.25	80	100	4	5.68	22776	17.7
0.5	0.25	90	100	4	1.68	23839	16.2
0.5	0.25	100	100	4	0.20	24523	15.0
1.0	0.5	80	100	2	6.54	30430	35.4
1.0	0.5	90	100	2	2.26	32144	34.4
1.0	0.5	80	100	4	6.16	30613	35.2
1.0	0.5	90	100	4	1.50	32360	34.1
1.0	0.5	100	100	4	0.20	33510	33.5
1.0	0.25	100	100	4	0.10	54915	17.9

Table 26: Template: Results for networks of the configuration: FurtherSinkCorner and size 15

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	90	100	2	2.42	46287	2.5
0.125	0.0625	80	100	4	6.88	43096	3.7
0.125	0.0625	90	100	4	2.46	46075	2.5
0.125	0.0625	100	100	4	0.38	47833	1.9
0.25	0.125	80	100	1	6.24	34218	9.2
0.25	0.125	90	100	2	2.20	35463	7.6
0.25	0.125	80	100	4	6.34	33366	9.3
0.25	0.125	90	100	4	2.16	35489	7.2
0.25	0.125	100	100	4	0.28	36558	6.3

0.5	0.25	80	100	2	6.72	66120	19.5
0.5	0.25	90	100	2	1.80	70222	17.5
0.5	0.25	100	100	2	0.16	72584	16.1
0.5	0.25	80	100	4	5.60	67115	18.9
0.5	0.25	90	100	4	1.52	70751	16.9
0.5	0.125	100	100	4	0.12	84486	7.7
1.0	0.5	80	100	2	6.36	90988	38.5
1.0	0.5	90	100	2	2.04	96427	37.2
1.0	0.5	100	100	2	0.06	99632	36.1
1.0	0.5	80	100	4	5.46	91837	38.1
1.0	0.5	90	100	4	1.58	96786	37.0

Table 27: Template: Results for networks of the configuration: FurtherSinkCorner and size 21

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	100	100	4	0.60	84147	1.8
0.125	0.0625	80	100	2	7.12	77587	3.6
0.125	0.0625	100	100	2	0.56	85822	1.7
0.125	0.0625	80	100	4	8.26	75554	3.9
0.125	0.0625	90	100	4	2.66	81568	2.4
0.125	0.0625	90	100	2	2.48	83027	2.3
0.25	0.125	80	100	1	6.14	60955	8.7
0.25	0.125	100	100	2	0.26	65613	6.1
0.25	0.125	80	100	4	6.24	59453	9.0
0.25	0.125	90	100	4	1.96	63080	6.9
0.25	0.125	100	100	4	0.30	65223	6.0
0.5	0.25	90	100	1	1.90	129675	14.8
0.5	0.25	80	100	2	6.92	120364	17.6
0.5	0.25	90	100	2	1.92	127960	15.3
0.5	0.25	100	100	2	0.16	132180	14.2
0.5	0.25	80	100	4	6.08	121380	17.2
0.5	0.25	100	100	4	0.06	132259	14.1
1.0	0.5	100	100	1	0.02	188520	31.6
1.0	0.5	80	100	2	6.24	164695	35.6
1.0	0.5	100	100	2	0.04	181976	33.0
1.0	0.5	80	100	4	5.46	165387	35.6
1.0	0.5	90	100	4	1.82	174487	34.3
1.0	0.5	100	100	4	0.10	181007	33.3

Table 28: Template: Results for networks of the configuration: FurtherSinkCorner and size 25

D.4 Generic1

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	4.08	2641	4.7
0.125	0.0625	90	100	4	3.20	2774	3.6
0.125	0.0625	100	100	4	2.36	2855	3.2
0.25	0.125	100	100	2	0.48	1473	9.0
0.25	0.125	80	100	4	0.74	1335	9.4
0.25	0.125	90	100	4	0.56	1375	8.7
0.25	0.125	100	100	4	0.54	1400	8.1
0.25	0.0625	90	100	4	0.20	1577	6.7
0.25	0.0625	100	100	4	0.16	1600	6.5
0.25	0.0625	80	100	4	0.42	1533	7.1
0.5	0.25	80	100	4	0.82	2394	16.3
0.5	0.25	90	100	4	0.64	2462	15.3
0.5	0.25	100	100	4	0.48	2512	14.7

0.5	0.125	100	100	4	0.42	2871	10.0
1.0	0.5	80	100	1	3.70	3990	44.5
1.0	0.5	80	100	4	1.28	4099	35.5
1.0	0.5	90	100	4	1.16	4359	32.9
1.0	0.5	100	100	4	0.78	4543	31.0
1.0	0.25	90	100	4	0.62	5665	17.2
1.0	0.25	100	100	4	0.48	5791	16.4
1.0	0.125	90	100	4	0.44	6334	12.2
1.0	0.125	100	100	4	0.40	6474	11.2

Table 29: Template: Results for networks of the configuration: Generic1 and size 11

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	8.84	11140	7.6
0.125	0.0625	80	100	4	3.52	11161	3.7
0.125	0.0625	90	100	4	1.84	11855	2.4
0.125	0.0625	100	100	4	0.82	12391	1.7
0.25	0.125	80	100	4	0.74	4832	7.2
0.25	0.125	90	100	4	0.22	5005	5.9
0.25	0.125	100	100	4	0.08	5117	5.2
0.25	0.0625	100	100	4	0.02	5571	4.1
0.5	0.25	80	100	4	0.30	8615	11.1
0.5	0.25	90	100	4	0.06	8895	9.7
0.5	0.25	100	100	4	0.02	9086	9.0
0.5	0.125	90	100	4	0.00	9708	6.6
1.0	0.5	80	100	1	2.86	13534	42.3
1.0	0.5	90	100	1	2.46	14858	37.8
1.0	0.5	80	100	4	0.44	16100	24.3
1.0	0.5	90	100	4	0.04	17195	20.4
1.0	0.125	100	100	4	0.00	22698	6.2

Table 30: Template: Results for networks of the configuration: Generic1 and size 15

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	3.46	36776	3.2
0.125	0.0625	90	100	4	1.24	39654	1.7
0.125	0.0625	100	100	4	0.34	41513	1.0
0.25	0.125	100	100	2	0.02	17461	3.3
0.25	0.125	80	100	4	0.62	15057	6.3
0.25	0.125	90	100	4	0.24	15707	4.8
0.25	0.125	100	100	4	0.06	16111	4.0
0.5	0.25	80	100	4	0.42	26064	10.0
0.5	0.25	90	100	4	0.10	27055	7.9
0.5	0.25	100	100	4	0.00	27712	6.9
1.0	0.5	80	100	1	2.72	43820	37.3
1.0	0.5	90	100	1	2.42	46346	34.7
1.0	0.5	80	100	4	0.32	53114	18.5
1.0	0.5	90	100	4	0.08	56308	14.1
1.0	0.25	90	100	4	0.06	63107	7.4
1.0	0.125	100	100	4	0.04	68198	4.3

Table 31: Template: Results for networks of the configuration: Generic1 and size 21

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	3.48	65996	3.2
0.125	0.0625	90	100	4	1.20	71340	1.6
0.125	0.0625	100	100	4	0.26	74707	0.8

0.125	0.0625	80	100	1	10.60	65352	7.8
0.25	0.0625	100	100	4	0.00	29723	2.4
0.25	0.125	80	100	4	0.90	26492	5.9
0.25	0.125	90	100	4	0.16	27637	4.3
0.25	0.125	100	100	4	0.06	28391	3.5
0.25	0.0625	80	100	4	0.88	27620	4.5
0.5	0.25	80	100	4	0.52	44829	9.3
0.5	0.25	90	100	4	0.06	46699	7.0
0.5	0.25	100	100	4	0.00	47739	6.0
1.0	0.5	80	100	1	2.62	76633	36.7
1.0	0.5	90	100	1	1.86	79378	35.4
1.0	0.5	100	100	1	2.44	79376	36.0
1.0	0.5	80	100	4	0.40	92715	16.7
1.0	0.5	90	100	4	0.06	97519	12.8
1.0	0.5	100	100	4	0.00	100197	10.9

Table 32: Template: Results for networks of the configuration: Generic1 and size 25

D.5 Generic2

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	6.08	5513	5.1
0.125	0.0625	80	100	2	4.02	5666	4.6
0.125	0.0625	100	100	2	2.70	6225	3.1
0.125	0.0625	80	100	4	2.88	5929	3.4
0.125	0.0625	90	100	4	1.58	6231	2.5
0.125	0.0625	100	100	4	0.90	6485	2.0
0.25	0.125	80	100	1	6.40	4253	12.3
0.25	0.125	90	100	1	4.64	4461	10.7
0.25	0.125	80	100	2	3.72	4470	10.4
0.25	0.125	90	100	2	2.76	4643	9.3
0.25	0.125	80	100	4	1.60	4767	7.0
0.25	0.125	90	100	4	1.14	4930	6.0
0.25	0.125	100	100	4	0.94	5094	5.1
0.25	0.0625	90	100	2	0.80	5483	5.3
0.25	0.0625	80	100	4	0.88	5405	4.8
0.25	0.0625	90	100	4	0.40	5619	3.8
0.5	0.25	90	100	1	1.90	7959	18.8
0.5	0.25	80	100	2	2.26	7730	20.0
0.5	0.25	90	100	2	2.06	7914	19.6
0.5	0.25	80	100	4	1.56	8234	16.3
0.5	0.25	90	100	4	1.34	8405	16.0
0.5	0.25	100	100	4	1.32	8581	15.3
0.5	0.125	90	100	1	1.20	9495	11.4
0.5	0.125	90	100	2	1.26	9459	12.0
0.5	0.125	100	100	2	1.00	9673	11.0
0.5	0.125	80	100	4	1.10	9568	10.8
0.5	0.125	90	100	4	0.60	9862	9.5
1.0	0.5	90	100	1	1.32	11376	38.5
1.0	0.5	100	100	1	1.22	11658	38.2
1.0	0.5	80	100	2	2.34	10411	41.8
1.0	0.5	90	100	2	1.76	10738	41.3
1.0	0.5	100	100	2	1.74	10955	41.1
1.0	0.5	80	100	4	1.88	10723	40.4
1.0	0.5	90	100	4	1.52	11073	39.8
1.0	0.5	100	100	4	1.40	11284	39.7
1.0	0.25	90	100	2	0.58	16964	21.4
1.0	0.25	100	100	2	0.52	17118	21.5
1.0	0.25	80	100	4	0.90	16928	21.0

1.0	0.25	90	100	4	0.24	17197	20.6

Table 33: Template: Results for networks of the configuration: Generic2 and size 11

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	3.42	16437	4.7
0.125	0.0625	90	100	2	1.96	17301	3.7
0.125	0.0625	80	100	4	2.56	17026	4.0
0.125	0.0625	90	100	4	1.48	17873	3.1
0.125	0.0625	100	100	4	0.66	18697	2.6
0.25	0.125	80	100	2	3.18	12636	12.3
0.25	0.125	90	100	2	1.64	13220	10.5
0.25	0.125	80	100	4	1.54	13486	9.5
0.25	0.125	90	100	4	0.82	14089	7.9
0.25	0.125	100	100	4	0.50	14447	7.2
0.25	0.0625	90	100	4	0.40	15860	4.3
0.25	0.0625	100	100	4	0.10	16422	3.6
0.5	0.25	80	100	2	1.04	23780	21.5
0.5	0.25	90	100	2	0.46	24224	20.5
0.5	0.25	100	100	2	0.38	24483	20.3
0.5	0.25	90	100	4	0.18	24842	19.1
0.5	0.25	100	100	4	0.14	25194	18.6
0.5	0.125	100	100	4	0.08	28865	11.1
1.0	0.5	80	100	4	0.66	32722	42.5
1.0	0.5	90	100	4	0.26	33420	42.0
1.0	0.5	100	100	4	0.14	34042	41.5
1.0	0.25	90	100	4	0.10	53399	22.5

Table 34: Template: Results for networks of the configuration: Generic2 and size 15

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	2.08	51457	3.8
0.125	0.0625	80	100	4	1.56	51529	3.6
0.125	0.0625	90	100	4	0.74	53722	3.0
0.125	0.0625	100	100	4	0.32	55464	2.5
0.25	0.125	80	100	2	1.30	39305	9.8
0.25	0.125	90	100	2	0.48	40481	8.7
0.25	0.125	80	100	4	1.02	39904	9.0
0.25	0.125	90	100	4	0.32	41128	7.9
0.25	0.125	100	100	4	0.10	42025	7.4
0.25	0.0625	100	100	4	0.06	47690	3.4
0.5	0.25	80	100	4	0.60	76112	18.7
0.5	0.25	90	100	4	0.08	77378	17.8
0.5	0.25	100	100	4	0.02	78193	17.3
0.5	0.125	100	100	4	0.00	87967	9.8
1.0	0.5	100	100	2	0.22	110310	41.6
1.0	0.5	80	100	4	0.62	102022	45.4
1.0	0.5	90	100	4	0.24	104125	44.4
1.0	0.25	100	100	1	0.16	179733	16.9

Table 35: Template: Results for networks of the configuration: Generic2 and size 21

P_{src} (seconds)	P_{src} (seconds)	Pr(TFS) (%)	Pr(PFS) (%)	Duration (seconds)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	1.42	90616	3.4
0.125	0.0625	90	100	4	0.52	93514	2.9
0.125	0.0625	100	100	4	0.16	96243	2.5

0.25	0.125	80	100	2	1.10	69704	9.2
0.25	0.125	80	100	4	1.16	69630	9.0
0.25	0.125	90	100	4	0.18	71663	8.0
0.25	0.125	100	100	4	0.04	73339	7.3
0.5	0.25	90	100	4	0.12	137625	16.7
0.5	0.25	100	100	4	0.04	139183	16.2
0.5	0.25	80	100	4	0.66	135355	17.6
1.0	0.5	100	100	1	0.16	221011	33.1
1.0	0.5	100	100	2	0.24	200172	38.0
1.0	0.5	80	100	4	0.76	185053	41.9
1.0	0.5	90	100	4	0.28	188776	40.9
1.0	0.125	100	100	1	0.10	352155	7.5

Table 36: Template: Results for networks of the configuration: Generic2 and size 25

E Template Results Tables Using the Wrong Algorithm

Heading Meanings

P_{src}	The Source period
P_{fs}	The Fake Source period
TFS	The probability that a temporary fake source is created
PFS	The probability that a permanent fake source is created
D	The duration of a temporary fake source

E.1 SourceCorner

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	7.9 (-2.3, -25%)	4654 (+24, +1%)	6 (-2, -35%)
0.125	0.0625	80	100	4	2.5 (-0.1, -2%)	4923 (+50, +1%)	3 (-0, -7%)
0.125	0.0625	90	100	4	1.5 (-0.2, -14%)	5138 (+52, +1%)	2 (-0, -11%)
0.125	0.0625	100	100	4	1.0 (-0.3, -23%)	5311 (+55, +1%)	2 (-0, -8%)
0.25	0.0625	90	100	4	0.4 (-0.2, -50%)	5005 (+32, +1%)	4 (-0, -6%)
0.25	0.0625	100	100	4	0.2 (-0.1, -50%)	5135 (+32, +1%)	3 (-0, -5%)
0.25	0.125	80	100	1	5.1 (-1.2, -21%)	4144 (-94, -2%)	11 (-2, -14%)
0.25	0.125	80	100	4	1.9 (+0.3, +20%)	4319 (+22, +1%)	6 (-0, -4%)
0.25	0.125	90	100	4	1.1 (-0.0, -2%)	4480 (+27, +1%)	5 (-0, -4%)
0.25	0.125	100	100	4	0.7 (-0.3, -40%)	4599 (+27, +1%)	4 (-0, -3%)
0.5	0.25	80	100	1	4.1 (-3.5, -59%)	6722 (-189, -3%)	21 (-2, -9%)
0.5	0.25	80	100	4	1.4 (-0.3, -19%)	8155 (+108, +1%)	10 (-1, -8%)
0.5	0.25	90	100	1	2.7 (-0.1, -4%)	7014 (-552, -8%)	20 (+0, +0%)
0.5	0.25	90	100	2	1.6 (+0.6, +43%)	8112 (-645, -8%)	14 (+2, +19%)
0.5	0.25	90	100	4	0.8 (+0.0, +0%)	8534 (+82, +1%)	9 (-0, -4%)
0.5	0.25	100	100	1	2.2 (+0.5, +27%)	7182 (-625, -8%)	19 (+1, +3%)
0.5	0.25	100	100	4	0.7 (+0.2, +29%)	8811 (+56, +1%)	8 (-0, -2%)
0.5	0.125	90	100	4	0.5 (-0.2, -40%)	9603 (+137, +1%)	6 (-1, -11%)
1.0	0.5	80	100	1	3.5 (-4.3, -76%)	8372 (+171, +2%)	40 (-7, -16%)
1.0	0.5	90	100	1	2.1 (-0.9, -36%)	8760 (-94, -1%)	38 (-7, -16%)
1.0	0.5	90	100	4	1.9 (+1.3, +102%)	13476 (-1607, -11%)	23 (+3, +16%)
1.0	0.5	100	100	1	1.9 (-0.3, -15%)	9106 (+92, +1%)	38 (-7, -16%)
1.0	0.5	100	100	4	1.7 (+1.1, +93%)	13902 (-1863, -13%)	22 (+4, +22%)
1.0	0.25	90	100	1	1.2 (-2.0, -89%)	14094 (+184, +1%)	23 (-4, -17%)
1.0	0.25	100	100	1	0.6 (-0.3, -37%)	14418 (+95, +1%)	22 (-4, -17%)

Table 37: Template: Results for networks of the configuration: SourceCorner and size 11

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	6.3 (-2.2, -30%)	12348 (-816, -6%)	5 (-1, -22%)
0.125	0.0625	80	100	2	3.2 (-2.3, -54%)	13319 (-133, -1%)	4 (-1, -18%)
0.125	0.0625	80	100	4	1.8 (-0.6, -29%)	13367 (-40, -0%)	3 (-0, -4%)
0.125	0.0625	90	100	1	4.6 (+1.1, +26%)	13071 (-1875, -13%)	4 (-0, -1%)
0.125	0.0625	90	100	4	0.8 (-0.3, -30%)	14016 (-101, -1%)	2 (+0, +4%)
0.125	0.0625	100	100	4	0.5 (-0.0, -4%)	14490 (-117, -1%)	2 (+0, +5%)
0.25	0.0625	100	100	4	0.1 (+0.0, +0%)	13401 (-156, -1%)	3 (+0, +5%)
0.25	0.125	80	100	1	3.8 (-1.0, -23%)	10932 (-1073, -9%)	11 (+1, +13%)
0.25	0.125	80	100	4	1.5 (+0.1, +6%)	11649 (-35, -0%)	6 (-0, -0%)
0.25	0.125	90	100	1	3.1 (+2.0, +97%)	11522 (-1710, -14%)	9 (+3, +39%)
0.25	0.125	90	100	4	0.4 (-0.1, -26%)	12152 (-60, -0%)	4 (+0, +3%)
0.25	0.125	100	100	4	0.2 (+0.0, +12%)	12510 (-76, -1%)	4 (+0, +5%)
0.5	0.25	80	100	1	3.0 (-2.5, -58%)	18413 (-1396, -7%)	20 (+1, +3%)
0.5	0.25	90	100	1	1.3 (-0.5, -33%)	19143 (-2036, -10%)	18 (+2, +10%)

0.5	0.25	90	100	4	0.3 (-0.0, -7%)	23957 (+28, +0%)	7 (+0, +3%)
0.5	0.25	100	100	1	1.0 (+0.3, +42%)	19472 (-1930, -9%)	18 (+1, +7%)
0.5	0.25	100	100	2	0.4 (+0.2, +71%)	23195 (-2473, -10%)	11 (+4, +48%)
0.5	0.25	100	100	4	0.2 (+0.1, +40%)	24708 (-4, -0%)	6 (+0, +5%)
0.5	0.125	90	100	1	0.8 (-0.8, -67%)	21972 (-1747, -8%)	13 (-0, -3%)
0.5	0.125	100	100	1	0.6 (-0.0, -4%)	22298 (-1843, -8%)	12 (-1, -4%)
0.5	0.125	100	100	4	0.1 (+0.1, +80%)	26853 (+114, +0%)	4 (+0, +3%)
1.0	0.5	80	100	1	3.1 (-5.0, -89%)	22904 (-1022, -4%)	42 (-2, -4%)
1.0	0.5	90	100	1	0.9 (-0.9, -68%)	24214 (-1232, -5%)	40 (-2, -5%)
1.0	0.5	100	100	1	0.4 (-0.7, -93%)	25018 (-311, -1%)	39 (-4, -10%)
1.0	0.25	100	100	1	0.1 (-0.5, -128%)	39349 (-189, -0%)	20 (-3, -13%)
1.0	0.125	100	100	2	0.0 (-0.3, -158%)	47796 (-2642, -5%)	13 (+1, +4%)

Table 38: Template: Results for networks of the configuration: SourceCorner and size 15

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	4.6 (-3.3, -53%)	36419 (-4641, -12%)	5 (-1, -12%)
0.125	0.0625	80	100	4	1.9 (-0.1, -5%)	40052 (-1059, -3%)	3 (+0, +15%)
0.125	0.0625	90	100	1	2.6 (+0.6, +28%)	38207 (-8402, -20%)	4 (+1, +25%)
0.125	0.0625	90	100	2	1.4 (+0.1, +4%)	41669 (-4153, -9%)	3 (+1, +40%)
0.125	0.0625	90	100	4	0.5 (+0.0, +8%)	42232 (-1273, -3%)	2 (+1, +33%)
0.125	0.0625	100	100	1	2.1 (+1.4, +106%)	39325 (-9427, -21%)	3 (+1, +29%)
0.125	0.0625	100	100	4	0.2 (+0.1, +67%)	43498 (-1488, -3%)	1 (+0, +41%)
0.25	0.0625	90	100	1	1.0 (-0.1, -12%)	34354 (-6901, -18%)	6 (+2, +39%)
0.25	0.0625	100	100	4	0.1 (+0.0, +67%)	37496 (-1138, -3%)	2 (+1, +32%)
0.25	0.125	80	100	1	2.5 (-1.4, -42%)	31354 (-4621, -14%)	10 (+3, +34%)
0.25	0.125	90	100	1	1.4 (+0.9, +100%)	32670 (-6557, -18%)	9 (+5, +71%)
0.25	0.125	90	100	4	0.2 (-0.1, -43%)	35343 (-416, -1%)	4 (+0, +14%)
0.25	0.125	100	100	1	1.2 (+1.0, +141%)	33354 (-7048, -19%)	8 (+4, +74%)
0.25	0.125	100	100	4	0.1 (+0.0, +22%)	36251 (-513, -1%)	3 (+1, +21%)
0.5	0.25	80	100	1	1.8 (-3.6, -98%)	54205 (-4545, -8%)	18 (+2, +13%)
0.5	0.25	90	100	1	0.5 (-0.6, -77%)	55135 (-6695, -11%)	18 (+4, +24%)
0.5	0.25	90	100	4	0.1 (+0.0, +50%)	70749 (-171, -0%)	5 (+1, +16%)
0.5	0.25	100	100	1	0.3 (+0.1, +40%)	55964 (-5579, -9%)	17 (+3, +16%)
0.5	0.25	100	100	2	0.2 (+0.1, +120%)	68818 (-6107, -8%)	9 (+4, +64%)
0.5	0.25	100	100	4	0.0 (+0.0, +0%)	72926 (+112, +0%)	4 (+1, +14%)
0.5	0.125	100	100	1	0.2 (-0.1, -22%)	62747 (-4632, -7%)	11 (-1, -5%)
0.5	0.125	100	100	4	0.0 (-0.1, -200%)	77255 (+376, +0%)	3 (+0, +13%)
1.0	0.5	80	100	1	2.4 (-5.7, -109%)	67996 (-4525, -6%)	42 (+2, +4%)
1.0	0.5	90	100	1	0.3 (-1.1, -128%)	70347 (-4519, -6%)	41 (+1, +3%)
1.0	0.5	100	100	1	0.2 (-0.0, -20%)	71355 (-2827, -4%)	41 (+1, +1%)
1.0	0.5	100	100	2	0.2 (+0.0, +0%)	88795 (-10223, -11%)	33 (+5, +18%)
1.0	0.25	100	100	2	0.0 (-0.1, -120%)	124016 (-6424, -5%)	16 (+2, +13%)

Table 39: Template: Results for networks of the configuration: SourceCorner and size 21

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	4.1 (-2.5, -46%)	63256 (-10300, -15%)	5 (-0, -2%)
0.125	0.0625	90	100	1	2.0 (+0.6, +36%)	66203 (-15957, -22%)	4 (+1, +33%)
0.125	0.0625	100	100	1	1.3 (+0.8, +98%)	68203 (-16280, -21%)	3 (+1, +25%)
0.125	0.0625	90	100	2	0.8 (+0.1, +8%)	73184 (-8304, -11%)	3 (+1, +54%)
0.125	0.0625	100	100	2	0.4 (+0.2, +92%)	74974 (-10223, -13%)	2 (+1, +75%)
0.125	0.0625	90	100	4	0.6 (+0.1, +15%)	73748 (-3041, -4%)	2 (+1, +44%)
0.125	0.0625	100	100	4	0.1 (+0.1, +67%)	76186 (-3260, -4%)	1 (+1, +58%)
0.25	0.125	80	100	1	2.2 (-1.6, -52%)	54558 (-8906, -15%)	10 (+3, +42%)
0.25	0.125	90	100	1	1.1 (+0.8, +118%)	56080 (-12754, -20%)	9 (+5, +86%)
0.25	0.125	100	100	1	0.5 (+0.5, +160%)	57582 (-12774, -20%)	8 (+5, +83%)
0.25	0.125	90	100	4	0.2 (+0.0, +11%)	61638 (-1328, -2%)	3 (+1, +28%)

0.25	0.125	100	100	4	0.0 (-0.1, -200%)	63351 (-1065, -2%)	3 (+1, +28%)
0.5	0.25	80	100	1	1.9 (-2.8, -84%)	93255 (-10003, -10%)	18 (+4, +24%)
0.5	0.25	90	100	1	0.4 (-0.0, -10%)	95149 (-11725, -12%)	17 (+4, +29%)
0.5	0.25	100	100	1	0.0 (-0.1, -86%)	96433 (-9425, -9%)	17 (+3, +18%)
0.5	0.25	100	100	4	0.0 (+0.0, +200%)	126048 (-356, -0%)	4 (+1, +25%)
0.5	0.125	100	100	4	0.0 (+0.0, +0%)	132848 (+382, +0%)	3 (+0, +21%)
1.0	0.5	80	100	1	3.3 (-5.0, -86%)	118817 (-9637, -8%)	41 (+1, +4%)
1.0	0.5	90	100	1	0.4 (-1.2, -117%)	122628 (-7589, -6%)	40 (+0, +0%)
1.0	0.5	100	100	1	0.0 (-0.2, -150%)	124476 (-3898, -3%)	39 (-1, -2%)
1.0	0.5	90	100	4	0.0 (+0.0, +0%)	231165 (-10141, -4%)	11 (+4, +39%)

Table 40: Template: Results for networks of the configuration: SourceCorner and size 25

E.2 SinkCorner

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	33.1 (+20.8, +92%)	1824 (-773, -35%)	12 (+6, +65%)
0.125	0.0625	80	100	2	24.8 (+14.6, +83%)	2146 (-505, -21%)	9 (+4, +50%)
0.125	0.0625	80	100	4	10.7 (+2.9, +31%)	2605 (-162, -6%)	6 (+1, +21%)
0.125	0.0625	90	100	1	30.2 (+21.3, +109%)	1918 (-842, -36%)	11 (+6, +77%)
0.125	0.0625	90	100	2	20.5 (+14.6, +111%)	2313 (-542, -21%)	8 (+4, +63%)
0.125	0.0625	90	100	4	6.9 (+3.3, +63%)	2799 (-172, -6%)	4 (+1, +31%)
0.125	0.0625	100	100	1	26.6 (+20.9, +130%)	2044 (-899, -36%)	10 (+6, +89%)
0.125	0.0625	100	100	2	18.6 (+15.1, +137%)	2403 (-605, -22%)	7 (+4, +76%)
0.125	0.0625	100	100	4	5.1 (+3.3, +96%)	2912 (-201, -7%)	4 (+1, +41%)
0.25	0.0625	90	100	2	17.2 (+14.2, +142%)	2532 (-491, -18%)	12 (+5, +59%)
0.25	0.0625	90	100	4	2.1 (+0.1, +5%)	3030 (-75, -2%)	5 (+0, +8%)
0.25	0.0625	100	100	4	0.7 (+0.3, +55%)	3137 (-95, -3%)	5 (+1, +14%)
0.25	0.125	80	100	1	30.2 (+18.6, +89%)	1864 (-582, -27%)	19 (+7, +45%)
0.25	0.125	80	100	4	6.9 (+0.8, +12%)	2559 (-44, -2%)	9 (+1, +7%)
0.25	0.125	90	100	1	26.1 (+18.7, +111%)	1981 (-628, -27%)	17 (+7, +54%)
0.25	0.125	90	100	2	19.5 (+14.5, +119%)	2244 (-415, -17%)	14 (+5, +46%)
0.25	0.125	90	100	4	3.7 (+0.6, +16%)	2684 (-72, -3%)	7 (+1, +11%)
0.25	0.125	100	100	1	23.5 (+18.4, +129%)	2053 (-660, -28%)	16 (+8, +60%)
0.25	0.125	100	100	2	17.4 (+13.7, +130%)	2333 (-436, -17%)	13 (+5, +51%)
0.25	0.125	100	100	4	2.3 (+0.4, +17%)	2791 (-69, -2%)	6 (+1, +13%)
0.5	0.25	80	100	1	25.0 (+15.4, +88%)	3838 (-1047, -24%)	26 (+7, +31%)
0.5	0.25	90	100	1	21.0 (+15.9, +122%)	4040 (-1140, -25%)	24 (+7, +35%)
0.5	0.25	100	100	1	17.3 (+13.9, +134%)	4198 (-1126, -24%)	24 (+7, +36%)
0.5	0.25	100	100	4	15.1 (+12.3, +137%)	4822 (-639, -12%)	18 (+4, +28%)
0.5	0.125	100	100	4	13.3 (+11.7, +159%)	5696 (-738, -12%)	13 (+4, +43%)
1.0	0.5	80	100	1	26.6 (+14.2, +73%)	4534 (-2008, -36%)	36 (+5, +14%)
1.0	0.5	80	100	2	31.1 (+18.0, +82%)	4469 (-1860, -34%)	37 (+4, +13%)
1.0	0.5	90	100	1	23.3 (+15.0, +95%)	4751 (-2156, -37%)	36 (+5, +16%)
1.0	0.5	100	100	1	20.3 (+13.7, +102%)	4940 (-2237, -37%)	35 (+6, +18%)
1.0	0.25	90	100	2	20.2 (+15.8, +129%)	9282 (-2196, -21%)	25 (+7, +31%)
1.0	0.25	100	100	1	16.5 (+14.7, +162%)	9433 (-2570, -24%)	26 (+9, +39%)

Table 41: Template: Results for networks of the configuration: SinkCorner and size 11

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	27.5 (+16.8, +88%)	5150 (-1768, -29%)	11 (+5, +64%)
0.125	0.0625	80	100	2	21.2 (+11.9, +78%)	5797 (-1231, -19%)	9 (+3, +48%)
0.125	0.0625	80	100	4	13.2 (+5.3, +50%)	6820 (-498, -7%)	6 (+2, +29%)
0.125	0.0625	90	100	1	22.8 (+17.3, +122%)	5611 (-1918, -29%)	9 (+5, +80%)
0.125	0.0625	90	100	2	17.0 (+12.1, +110%)	6234 (-1351, -20%)	7 (+4, +63%)
0.125	0.0625	90	100	4	9.0 (+5.1, +80%)	7262 (-521, -7%)	5 (+1, +35%)
0.125	0.0625	100	100	1	21.3 (+18.0, +146%)	5683 (-2160, -32%)	9 (+6, +95%)

0.125	0.0625	100	100	2	13.9 (+10.7, +125%)	6531 (-1318, -18%)	6 (+3, +72%)
0.125	0.0625	100	100	4	7.1 (+5.6, +129%)	7662 (-508, -6%)	4 (+2, +47%)
0.25	0.0625	90	100	1	10.6 (+6.8, +95%)	6080 (-1148, -17%)	11 (+4, +49%)
0.25	0.0625	100	100	1	8.2 (+6.5, +132%)	6259 (-1226, -18%)	10 (+5, +61%)
0.25	0.0625	100	100	2	4.2 (+3.1, +116%)	6961 (-569, -8%)	7 (+2, +36%)
0.25	0.0625	100	100	4	0.7 (-0.0, -6%)	7777 (-29, -0%)	4 (+0, +7%)
0.25	0.125	80	100	1	18.5 (+9.0, +64%)	5373 (-943, -16%)	16 (+4, +33%)
0.25	0.125	90	100	1	13.4 (+8.0, +84%)	5655 (-1039, -17%)	14 (+5, +42%)
0.25	0.125	90	100	2	9.2 (+4.7, +68%)	6251 (-502, -8%)	11 (+3, +25%)
0.25	0.125	90	100	4	3.5 (+0.8, +25%)	6994 (-100, -1%)	7 (+0, +6%)
0.25	0.125	100	100	1	11.9 (+7.7, +96%)	5888 (-998, -16%)	13 (+4, +42%)
0.25	0.125	100	100	2	6.9 (+4.0, +83%)	6479 (-503, -7%)	10 (+2, +26%)
0.25	0.125	100	100	4	1.7 (+0.5, +34%)	7335 (+54, +1%)	6 (-0, -2%)
0.5	0.25	80	100	1	16.1 (+8.1, +68%)	11732 (-2023, -16%)	22 (+6, +29%)
0.5	0.25	90	100	1	11.8 (+8.6, +114%)	12312 (-2254, -17%)	21 (+6, +36%)
0.5	0.25	90	100	4	7.7 (+5.6, +115%)	13898 (-923, -6%)	16 (+3, +20%)
0.5	0.25	100	100	1	9.4 (+8.1, +151%)	12670 (-2277, -16%)	20 (+7, +40%)
0.5	0.25	100	100	4	6.3 (+5.3, +147%)	14389 (-699, -5%)	15 (+2, +17%)
0.5	0.125	100	100	4	4.6 (+3.9, +149%)	16028 (-751, -5%)	10 (+2, +22%)
1.0	0.5	80	100	1	17.0 (+9.1, +73%)	14192 (-4521, -27%)	45 (+7, +16%)
1.0	0.5	90	100	1	12.9 (+9.0, +108%)	14988 (-5065, -29%)	44 (+7, +18%)
1.0	0.5	100	100	1	10.9 (+8.8, +135%)	15419 (-5374, -30%)	44 (+8, +20%)
1.0	0.25	90	100	1	10.2 (+7.9, +128%)	28135 (-4576, -15%)	23 (+7, +37%)
1.0	0.25	100	100	1	8.2 (+7.8, +182%)	28773 (-4756, -15%)	22 (+7, +40%)

Table 42: Template: Results for networks of the configuration: SinkCorner and size 15

P_{src} (sec)	P_{fs} (sec)	TFS (%)	PFS (%)	D (s)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	18.3 (+9.1, +66%)	15793 (-3983, -22%)	8 (+3, +51%)
0.125	0.0625	80	100	2	15.6 (+7.5, +63%)	16799 (-2729, -15%)	7 (+2, +40%)
0.125	0.0625	90	100	1	13.1 (+8.8, +102%)	16866 (-4333, -23%)	7 (+4, +72%)
0.125	0.0625	90	100	2	11.3 (+7.5, +99%)	18140 (-2701, -14%)	6 (+3, +53%)
0.125	0.0625	90	100	4	5.8 (+1.9, +39%)	20671 (-370, -2%)	4 (+1, +19%)
0.125	0.0625	100	100	1	11.3 (+9.0, +132%)	17475 (-4611, -23%)	6 (+4, +84%)
0.125	0.0625	100	100	2	8.6 (+6.7, +126%)	18937 (-2886, -14%)	5 (+3, +64%)
0.125	0.0625	100	100	4	3.2 (+1.6, +65%)	21931 (-197, -1%)	3 (+1, +23%)
0.25	0.0625	80	100	1	11.8 (+4.4, +46%)	15202 (-2267, -14%)	11 (+3, +33%)
0.25	0.0625	90	100	1	6.7 (+3.2, +64%)	16216 (-2392, -14%)	9 (+3, +43%)
0.25	0.0625	100	100	1	4.4 (+3.5, +128%)	16894 (-2538, -14%)	8 (+3, +54%)
0.25	0.0625	100	100	2	2.5 (+1.6, +99%)	18367 (-915, -5%)	6 (+2, +31%)
0.25	0.0625	100	100	4	0.4 (-0.3, -59%)	20448 (+740, +4%)	4 (-0, -6%)
0.25	0.125	80	100	1	13.8 (+5.2, +46%)	14440 (-2218, -14%)	15 (+4, +30%)
0.25	0.125	90	100	1	9.2 (+5.3, +80%)	15383 (-2311, -14%)	13 (+4, +37%)
0.25	0.125	90	100	2	5.6 (+1.5, +31%)	16697 (-925, -5%)	11 (+2, +18%)
0.25	0.125	100	100	1	7.0 (+4.8, +103%)	16081 (-2337, -14%)	12 (+4, +41%)
0.25	0.125	100	100	2	4.2 (+2.0, +61%)	17489 (-685, -4%)	10 (+2, +18%)
0.25	0.125	100	100	4	0.7 (-0.9, -81%)	19487 (+776, +4%)	6 (-1, -11%)
0.5	0.25	80	100	1	11.9 (+4.7, +50%)	33081 (-4332, -12%)	24 (+5, +24%)
0.5	0.25	90	100	1	7.2 (+4.9, +103%)	34720 (-4718, -13%)	23 (+6, +29%)
0.5	0.25	100	100	1	5.3 (+4.9, +170%)	35443 (-5010, -13%)	22 (+6, +32%)
0.5	0.25	100	100	2	5.2 (+4.6, +163%)	36946 (-3208, -8%)	21 (+4, +21%)
0.5	0.25	100	100	4	3.9 (+3.6, +165%)	39809 (-701, -2%)	17 (+1, +8%)
0.5	0.125	100	100	4	3.2 (+2.8, +161%)	43672 (-918, -2%)	11 (+1, +12%)
1.0	0.5	80	100	1	12.2 (+5.6, +60%)	43158 (-11779, -24%)	42 (+6, +15%)
1.0	0.5	90	100	1	6.8 (+4.8, +108%)	46080 (-12011, -23%)	41 (+6, +16%)
1.0	0.5	100	100	1	4.9 (+4.4, +162%)	47329 (-12782, -24%)	41 (+7, +18%)
1.0	0.25	100	100	1	4.3 (+4.1, +187%)	84156 (-9757, -11%)	25 (+7, +31%)

Table 43: Template: Results for networks of the configuration: SinkCorner and size 21

P_{src}	P_{fs}	TFS	PFS	D	Captured	Fake	Received
(sec)	(sec)	(%)	(%)	(s)	(%)	Messages	(%)
0.125	0.0625	90	100	1	9.2 (+5.4, +83%)	30491 (-5824, -17%)	6 (+3, +57%)
0.125	0.0625	100	100	1	7.7 (+6.5, +146%)	31576 (-6665, -19%)	5 (+3, +79%)
0.125	0.0625	80	100	2	13.1 (+5.2, +50%)	29774 (-3581, -11%)	6 (+2, +34%)
0.125	0.0625	100	100	2	6.1 (+4.8, +132%)	34040 (-3201, -9%)	4 (+2, +54%)
0.125	0.0625	90	100	4	4.1 (+0.8, +21%)	36718 (+964, +3%)	3 (+0, +9%)
0.125	0.0625	100	100	4	1.8 (+1.0, +79%)	38907 (+1465, +4%)	3 (+0, +10%)
0.125	0.0625	80	100	1	14.2 (+5.6, +49%)	28383 (-5886, -19%)	7 (+3, +43%)
0.25	0.125	80	100	1	11.3 (+3.1, +32%)	24337 (-2906, -11%)	14 (+3, +25%)
0.25	0.125	90	100	1	6.0 (+2.8, +63%)	26432 (-2596, -9%)	12 (+3, +27%)
0.25	0.125	100	100	1	4.0 (+2.5, +94%)	27413 (-2796, -10%)	11 (+3, +34%)
0.25	0.125	80	100	2	9.8 (+1.9, +22%)	25851 (-809, -3%)	13 (+1, +10%)
0.25	0.125	100	100	2	2.1 (+1.0, +59%)	29850 (+142, +0%)	9 (+1, +8%)
0.25	0.125	90	100	4	1.8 (-0.5, -23%)	31352 (+2244, +7%)	7 (-1, -17%)
0.25	0.125	100	100	4	0.4 (-0.4, -58%)	33151 (+2920, +9%)	6 (-2, -26%)
0.25	0.0625	80	100	1	10.8 (+4.3, +49%)	25662 (-3775, -14%)	11 (+3, +38%)
0.25	0.0625	100	100	1	3.1 (+2.4, +130%)	29047 (-3595, -12%)	7 (+3, +53%)
0.25	0.0625	100	100	2	1.5 (+0.8, +75%)	31458 (-206, -1%)	5 (+1, +19%)
0.5	0.25	80	100	1	9.8 (+3.6, +45%)	55276 (-6156, -11%)	26 (+5, +21%)
0.5	0.25	90	100	1	4.8 (+2.9, +87%)	58468 (-6571, -11%)	24 (+5, +24%)
0.5	0.25	100	100	1	2.7 (+2.3, +145%)	60186 (-6604, -10%)	24 (+6, +28%)
0.5	0.25	100	100	4	1.7 (+1.5, +158%)	68617 (+2323, +3%)	17 (-1, -5%)
0.5	0.125	100	100	4	1.7 (+1.5, +173%)	75613 (+885, +1%)	10 (+0, +4%)
1.0	0.5	80	100	1	9.1 (+2.7, +36%)	75208 (-16965, -20%)	44 (+5, +12%)
1.0	0.5	90	100	1	4.4 (+2.9, +101%)	79729 (-18372, -21%)	43 (+6, +14%)
1.0	0.5	100	100	1	2.9 (+2.8, +182%)	81826 (-20454, -22%)	43 (+6, +16%)
1.0	0.5	100	100	2	2.7 (+2.4, +165%)	86208 (-12282, -13%)	42 (+4, +9%)
1.0	0.25	100	100	1	2.5 (+2.3, +171%)	144281 (-13526, -9%)	28 (+7, +27%)
1.0	0.25	100	100	2	2.3 (+2.1, +174%)	147085 (-8750, -6%)	27 (+5, +19%)

Table 44: Template: Results for networks of the configuration: SinkCorner and size 25

F Adaptive Algorithm

```
message - Normal
**variables**
    % Sink-Source Distance
    ssd: int;

    % The hop count from the source
    hop: int;

    % What number message this is. Starts at 1
    count: int;

    % The maximum hop seen
    maxHop: int;
```

Figure 63: Source Location Privacy Algorithm - Normal Message

```
message - Fake
variables
    % Sink-Source Distance
    ssd: int;

    % Is from a permanent source
    Fperm: boolean;

    % The maximum hop seen
    maxHop: int;

    % The number of hops this message can travel
    Dtravel: int;

    % The $\Delta_{source}$ of the sending node
    FDsrc: int;

    % The $\Delta_{sink}$ of the sending node
    FDsink: int;

    % The id of the sending node
    Fid: int;
```

Figure 64: Source Location Privacy Algorithm - Fake Message

```
message - Away/Choose
variables
    % Sink-Source Distance
    ssd: int;

    % The distance travelled from the sink
    Dsink: int;

    % The maximum hop seen
    maxHop: int;

    % The algorithm
    algorithm: Algorithm;
```

Figure 65: Source Location Privacy Algorithm - Away/Choose Message

$$\mathbf{min}_{\perp}(x) = \begin{cases} \perp & x = \{\perp\} \\ \mathbf{min}(x \setminus \{\perp\}) & otherwise \end{cases} \tag{10}$$

```
process $j$ - If type is Sink
variables
    % Messages seen
    $messages$: set of int init $\emptyset$;

    % Records if the sink has sent the first away message
    $sinkSent$: boolean init False;

    % Distance between the sink and source
    $\Delta_{sink-source}$: int init $\perp$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;

constants
    % Source Period an algorithm parameter
    $P_{source}$: time;

actions
    % Receiving Normal Message
    $receiveNormal$:: $\mathbf{rcv}\langle Normal, hash, ssd, count, maxHop\rangle \rightarrow$
        if $(hash \notin messages)$ then
            $messages := messages \cup \{hash\}$;
            $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, hop + 1\}$;
            if $(\neg sinkSent)$ then
                $sinkSent := True$;
                $\mathcal{A} := \mathbf{GetAlgorithm}()$;
                $\mathbf{BCAST}\langle Away, \mathbf{hash}(Away), \Delta_{sink-source}, 0, maxHop, \mathcal{A}\rangle$ sense
                            repeat$(3)$ in $\frac{P_{source}}{2}$;
            fi;
        fi;

    % Receiving Fake Message
    $receiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, hop, Dtravel, FDsrc, FDsink, Fid\rangle \rightarrow$
        $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
        if $(hash \notin messages)$ then
            $messages := messages \cup \{hash\}$;
            if $(Dtravel \neq 0)$ then
                $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm, maxHop,$
                            $Dtravel - 1, FDsrc, FDsink, Fid\rangle$;
            fi;
        fi;
```

Figure 66: Source Location Privacy Algorithm - Sink

```
process $j$ - If type is Fake Source
**variables**
    % Messages seen
    $messages$: set of int init $\emptyset$;

    % The hop count of this node in the first wave of normal messages
    $firstHop$: int init $\perp$;

    % A flag that indicates if this node believes it should be a Permanent Fake Source
    $isPerm$: boolean init $False$;

    % Distance to to the source, distance to the sink, between the sink and source
    $\Delta_{source}, \Delta_{sink}, \Delta_{sink-source}$: int, int, int init $\perp, \perp, \perp$;

    % What percentage to adjust the fake range to
    $modifier$: real init $start$;

    % when type is **Temporary Fake Source** and $\mathcal{A}$ is Generic Specialisation
    $toSend$: int init $max_{\perp}\{\Delta_{source} - \Delta_{sink-source}, 1\}$;

    % when type is **Temporary Fake Source** and $\mathcal{A}$ is Further Specialisation
    $toSend$: int init $max_{\perp}\{\Delta_{source}, 1\}$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;

**constants**
    % The Dsink of the Away/Choose message that led to this node becoming a fake source
    $Dsink$: int;

    % The maximum time a node takes to send a message to another node
    $T_{send}$: time;

    % The source period
    $P_{source}$: time;

    % These control how the fake rate starts and is changed
    $start, change$: real, real init 1.00, 0.01;

    % when $\mathcal{A}$ is Generic Specialisation
    $end$: real init 0.40;

    % when $\mathcal{A}$ is Further Specialisation
    $end$: real init 0.75;
```

Figure 67: Source Location Privacy Algorithm - Fake Source

```
actions
    % Sending Fake Messagese when $\mathcal{A}$ is Further Specialisation
    sendFake:: timeout(period) $\rightarrow$
        BCAST$\langle Fake, \textbf{hash}(Fake), \Delta_{sink-source}, True,$
                $firstHop, \lceil modifier \times \Delta_{source} \rceil, \Delta_{source}, \Delta_{sink}, j \rangle$;
        if (modifier $>$ end)
            modifier := modifier $-$ change;
        fi;
        set(period, $0.85 \times P_{source}$);

    % Sending Fake Messagese when $\mathcal{A}$ is Generic Specialisation
    sendFake:: timeout(period) $\rightarrow$
        BCAST$\langle Fake, \textbf{hash}(Fake), \Delta_{sink-source}, True,$
                $firstHop, \lceil modifier \times \Delta_{source} \rceil, \Delta_{source}, \Delta_{sink}, j \rangle$;
        if (modifier $>$ end)
            modifier := modifier $-$ change;
        fi;
        if ($\neg$isPerm)
            BCAST$\langle Choose, \textbf{hash}(Choose), \Delta_{sink-source},$
                $Dsink + 1, firstHop, \mathcal{A} \rangle$ sense repeat(3);
            BecomeNormal();
        else
            set(period, $0.55 \times P_{source}$);
        fi;
```

Figure 68: Source Location Privacy Algorithm - **Permanent** Fake Source

```
    % Sending Fake Messages
    sendFake:: timeout(period) $\rightarrow$
        if (toSend $> 0$) then
            BCAST$\langle Fake, \textbf{hash}(Fake), \Delta_{sink-source}, False,$
                $firstHop, \lceil modifier \times \Delta_{source} \rceil, \Delta_{source}, \Delta_{sink}, j \rangle$;
            if (modifier $>$ end)
                modifier := modifier $-$ change;
            fi;
        fi;
        if (toSend $> 1$) then
            toSend := toSend $- 1$;
            set(period, $3 \times T_{send}$);
        else
            BCAST$\langle Choose, \textbf{hash}(Choose), \Delta_{sink-source},$
                $Dsink + 1, firstHop, \mathcal{A} \rangle$ sense repeat(3);
            BecomeNormal();
        fi;
```

Figure 69: Source Location Privacy Algorithm - **Temporary** Fake Source

```
% Receiving Normal Message
$receiveNormal$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages)$ then
        $messages := messages \cup \{hash\}$;
        $\mathbf{BCAST}\langle Normal, hash, \Delta_{sink-source}, hop+1, count, \mathbf{max}\{firstHop, maxHop\}\rangle$;
    fi;
```

Figure 70: Source Location Privacy Algorithm - Fake Source

```
% Receiving Fake Message when $\mathcal{A}$ is Generic Specialisation
$receiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, Dtravel, FDsrc, FDsink, Fid\rangle \rightarrow$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages)$ then
        $messages := messages \cup \{hash\}$;
        $seenPerm := seenPerm \vee Fperm$;
        if $(Dtravel \neq 0)$ then
            $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm, \mathbf{max}\{firstHop, maxHop\}$
                    $Dtravel - 1, FDsrc, FDsink, Fid\rangle$;
        fi;
        if $(duration = \infty \wedge$
          $Fperm \wedge$
          $((FDsrc > \Delta_{source}) \vee$
          $(FDsrc = \Delta source \wedge \Delta_{sink} < FDsink)$
          $(FDsrc = \Delta source \wedge \Delta_{sink} = FDsink \wedge j < fromId)))$ then
            BecomeNormal();
        fi;
    fi;

% Receiving Fake Message when $\mathcal{A}$ is Further Specialisation
$receiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, Dtravel, FDsrc, FDsink, Fid\rangle \rightarrow$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages)$ then
        $messages := messages \cup \{hash\}$;
        $seenPerm := seenPerm \vee Fperm$;
        if $(Dtravel \neq 0)$ then
            $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm, \mathbf{max}\{firstHop, maxHop\}$
                    $Dtravel - 1, FDsrc, FDsink, Fid\rangle$;
        fi;
    fi;
```

Figure 71: Source Location Privacy Algorithm - Fake Source

```
process $j$ - If type is Source
variables
    % The number of messages sent
    $count$: int init 0;

    % Distance from source to sink
    $\Delta_{sink-source}$: int init $\bot$;

    % How often messages are sent
    $period$: timer init $P_{source}$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\bot$;

constants
    % The source period
    $P_{source}$: time;

actions
    % Sending Normal Messages
    $sendNormal$:: timeout($period$) $\rightarrow$
        $count := count + 1$;
        BCAST$\langle Normal, \mathbf{hash}(Normal), \Delta_{sink-source}, 0, count, firstHop\rangle$;
        set($period$, $P_{source}$);

    % Receiving Away Message
    $receiveAway$:: rcv$\langle Away, hash, ssd, Dsink, maxHop, algorithm\rangle \rightarrow$
        if ($algorithm \neq \bot$) then
            $\mathcal{A} := algorithm$;
        fi;
        $\Delta_{sink-source} := \mathbf{min}_{\bot}\{\Delta_{sink-source}, ssd\}$;
        if ($hash \notin messages$) then
            $messages := messages \cup \{hash\}$;
            $\Delta_{sink-source} := \mathbf{min}_{\bot}\{\Delta_{sink-source}, Dsink + 1\}$;
            BCAST$\langle Away, hash, \Delta_{sink-source}, Dsink + 1, maxHop, \mathcal{A}\rangle$ sense repeat(2);
        fi;

    % Receiving Fake Message
    $receiveFake$:: rcv$\langle Fake, hash, ssd, Fperm, maxHop, Dtravel, FDsrc, FDsink, Fid\rangle \rightarrow$
        $\Delta_{sink-source} := \mathbf{min}_{\bot}\{\Delta_{sink-source}, ssd\}$;
```

Figure 72: Source Location Privacy Algorithm - Source

```
process $j$ - If type is Normal
variables
    % Messages seen
    $messages$: set of int init $\emptyset$;

    % The hop count of this node in the first wave of normal messages
    $firstHop$: int init $\perp$;

    % The flag that indicates if this is a Permanent Fake Source
    $isPerm$: boolean init $False$;

    % Indicates if this node has received a Fake message from a Permanent Fake Source
    $seenPerm$: boolean init $False$;

    % Distance to the source, to the sink, between the sink and source
    $\Delta_{source}, \Delta_{sink}, \Delta_{sink-source}$: int, int, int init $\perp, \perp, \perp$;

    % The algorithm running on this node
    $\mathcal{A}$: Algorithm init $\perp$;
```

Figure 73: Source Location Privacy Algorithm - Normal

```
actions
    % Receiving Normal Message
    $receiveNormal$:: $\mathbf{rcv}\langle Normal, hash, ssd, hop, count, maxHop\rangle \rightarrow$
        if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
            $isPerm := False$;
        fi;
        $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
        if $(hash \notin messages)$ then
            $messages := messages \cup \{hash\}$;
            if (count = 1) then
                $firstHop, isPerm := hop + 1, True$;
            fi;
            $\Delta_{source} := \mathbf{min}_{\perp}\{\Delta_{source}, hop + 1\}$;
            if $(\Delta_{sink-source} = \perp \vee \Delta_{source} = \perp \vee \Delta_{sink-source} \times 1.125 \geq \Delta_{source})$ then
                $\mathbf{BCAST}\langle Normal, hash, \Delta_{sink-source}, hop + 1, count,$
                        $\mathbf{max}\{firstHop, maxHop\}\rangle$;
            fi;
        fi;
```

Figure 74: Source Location Privacy Algorithm - Normal

```
% Receiving Fake Message
$receiveFake$:: $\mathbf{rcv}\langle Fake, hash, ssd, Fperm, maxHop, Dtravel, FDsrc, FDsink, Fid\rangle \to$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages)$ then
        $messages := messages \cup \{hash\}$;
        $seenPerm := seenPerm \vee Fperm$;
        if $(Dtravel \neq 0)$ then
            $\mathbf{BCAST}\langle Fake, hash, \Delta_{sink-source}, Fperm, \mathbf{max}\{firstHop, maxHop\},$
                        $Dtravel - 1, FDsrc, FDsink, Fid\rangle$;
        fi;
    fi;
```

Figure 75: Source Location Privacy Algorithm - Normal

```
% Receiving Away Message
$receiveAway$:: $\mathbf{rcv}\langle Away, hash, ssd, Dsink, maxHop, algorithm\rangle \to$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    if $(algorithm \neq \perp)$ then
        $\mathcal{A} := algorithm$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages)$ then
        $messages := messages \cup \{hash\}$;
        $\Delta_{sink} := \mathbf{min}_{\perp}\{\Delta_{sink}, Dsink\}$;
        if $(Dsink = 0)$ then
            BecomeTempFakeSource();
            % Need to create a new hash
            $\mathbf{BCAST}\langle Away, \mathbf{hash}(Away), \Delta_{sink-source}, Dsink + 1,$
                        $\mathbf{max}\{firstHop, maxHop\}, \mathcal{A}\rangle$ sense repeat(2);
        else
            $\mathbf{BCAST}\langle Away, hash, \Delta_{sink-source}, Dsink + 1,$
                        $\mathbf{max}\{firstHop, maxHop\}, \mathcal{A}\rangle$ sense repeat(2);
        fi;
    fi;
```

Figure 76: Source Location Privacy Algorithm - Normal

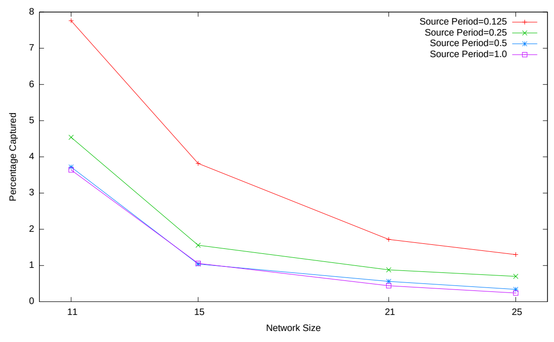
```
% Receiving Choose Message when $\mathcal{A}$ is Further Specialisation
$receiveChoose$:: $\mathbf{rcv}\langle Choose, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    if $(algorithm \neq \perp)$ then
        $\mathcal{A} := algorithm$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages \wedge \neg seenPerm \wedge$
            $\neg(\Delta_{sink-source} \neq \perp \wedge \Delta_{source} \leq \frac{1}{2}\Delta_{sink-source} - 1))$ then
        $messages := messages \cup \{hash\}$;
        if $(isPerm)$ then
            BecomeFakeSource();
        else
            BecomeTempFakeSource();
        fi;
    fi;

% Receiving Choose Message when $\mathcal{A}$ is Generic Specialisation
$receiveChoose$:: $\mathbf{rcv}\langle Choose, hash, ssd, hop, maxHop, algorithm\rangle \rightarrow$
    if $(firstHop = \perp \vee maxHop - 1 > firstHop)$ then
        $isPerm := False$;
    fi;
    if $(algorithm \neq \perp)$ then
        $\mathcal{A} := algorithm$;
    fi;
    $\Delta_{sink-source} := \mathbf{min}_{\perp}\{\Delta_{sink-source}, ssd\}$;
    if $(hash \notin messages \wedge \neg(\Delta_{sink-source} \neq \perp \wedge \Delta_{source} \leq \frac{4}{5}\Delta_{sink-source}))$ then
        $messages := messages \cup \{hash\}$;
        if $(isPerm)$ then
            BecomePermFakeSource();
        else
            BecomeTempFakeSource();
        fi;
    fi;
```

Figure 77: Source Location Privacy Algorithm - Normal

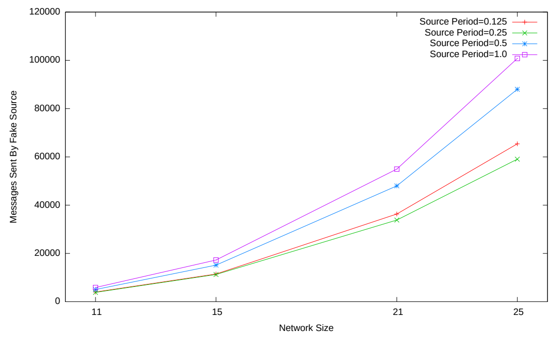
G Adaptive Results Tables

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	7.76	4004	11
	0.25	4.54	3856	17
	0.5	3.72	5031	35
	1.0	3.64	5882	54
15	0.125	3.82	11459	11
	0.25	1.56	11274	14
	0.5	1.04	15154	33
	1.0	1.06	17274	55
21	0.125	1.72	36342	10
	0.25	0.88	33815	12
	0.5	0.56	47987	33
	1.0	0.44	54943	58
25	0.125	1.30	65400	9
	0.25	0.70	59057	12
	0.5	0.34	87992	30
	1.0	0.24	100766	58

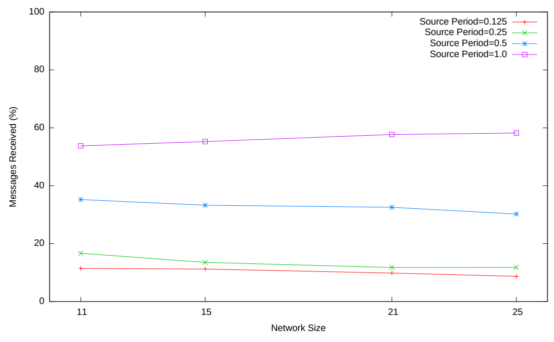
Table 45: Adaptive results for the SourceCorner configuration



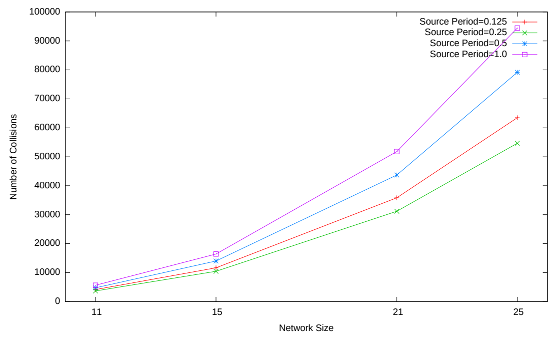
(a) Capture Rates



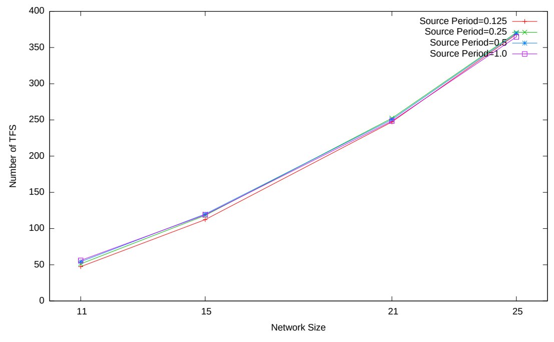
(b) Fake Messages Sent



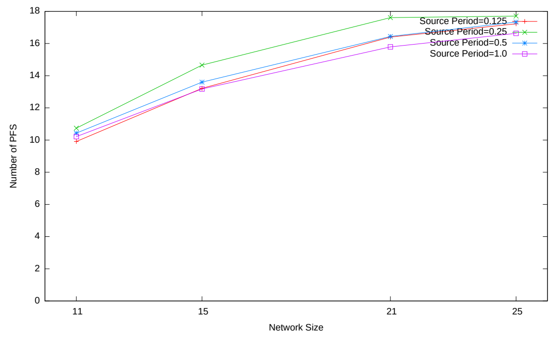
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

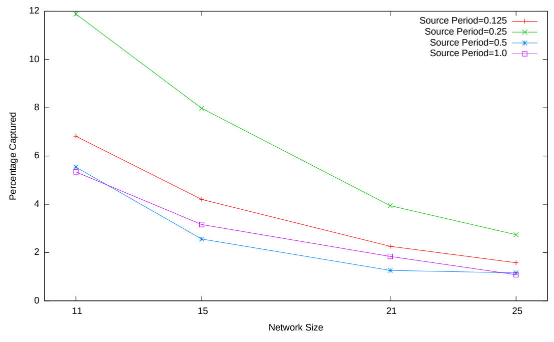


(f) Number of Permanent Fake Sources created

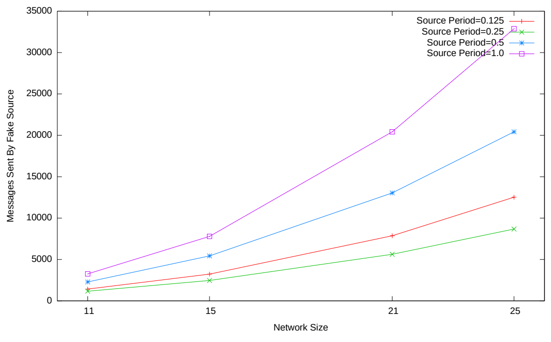
Figure 78: Results for the SourceCorner configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	6.82	1423	5
	0.25	11.88	1164	15
	0.5	5.54	2282	29
	1.0	5.34	3257	42
15	0.125	4.20	3229	4
	0.25	7.98	2460	12
	0.5	2.56	5426	24
	1.0	3.16	7791	38
21	0.125	2.26	7858	3
	0.25	3.94	5629	10
	0.5	1.26	13042	21
	1.0	1.84	20429	38
25	0.125	1.58	12520	3
	0.25	2.74	8679	9
	0.5	1.16	20428	20
	1.0	1.08	32881	36

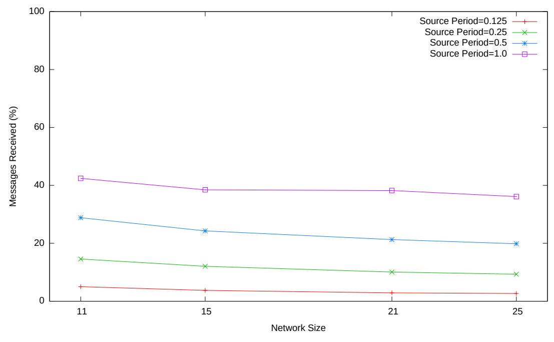
Table 46: Adaptive results for the SinkCorner configuration



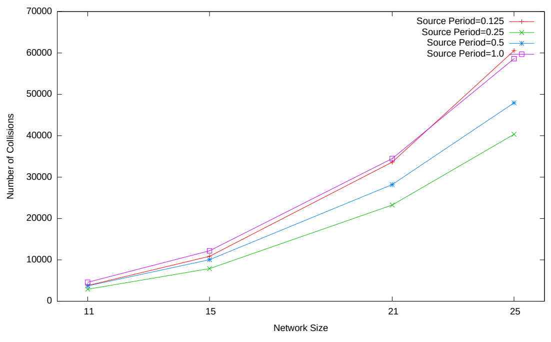
(a) Capture Rates



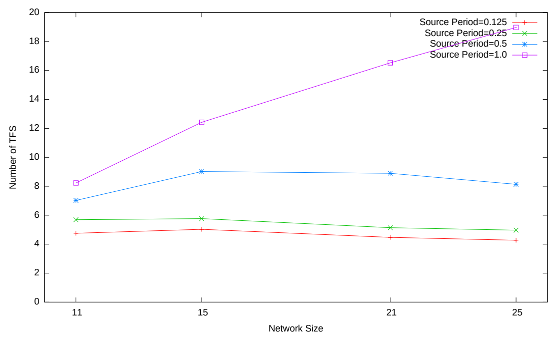
(b) Fake Messages Sent



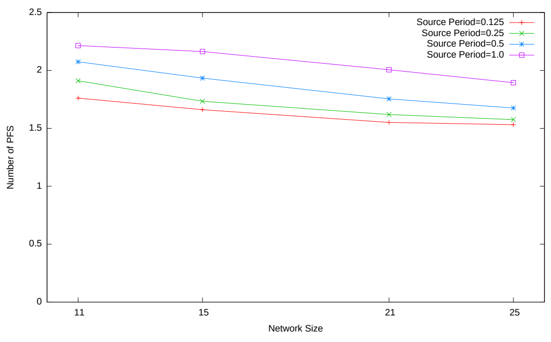
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

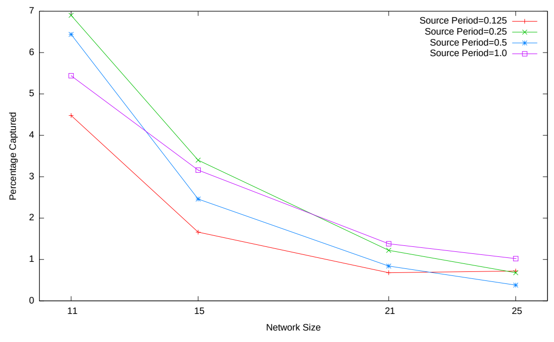


(f) Number of Permanent Fake Sources created

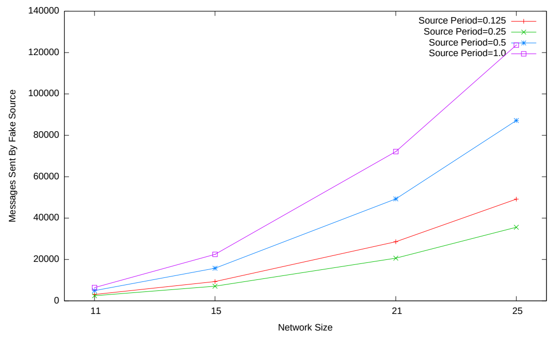
Figure 79: Results for the SinkCorner configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	4.48	3064	4
	0.25	6.90	2468	12
	0.5	6.44	4958	22
	1.0	5.44	6396	37
15	0.125	1.66	9328	3
	0.25	3.40	7091	9
	0.5	2.46	15746	19
	1.0	3.16	22481	35
21	0.125	0.68	28572	2
	0.25	1.22	20642	8
	0.5	0.84	49293	17
	1.0	1.38	72154	34
25	0.125	0.72	49154	2
	0.25	0.68	35555	7
	0.5	0.38	87150	16
	1.0	1.02	123655	35

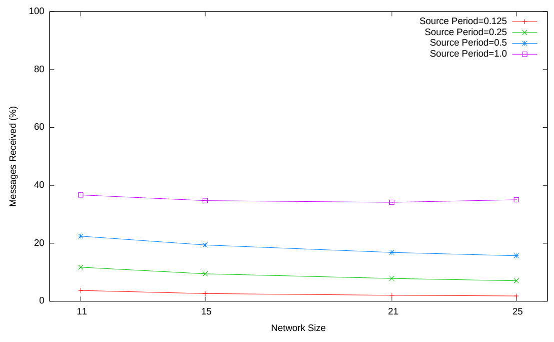
Table 47: Adaptive results for the FurtherSinkCorner configuration



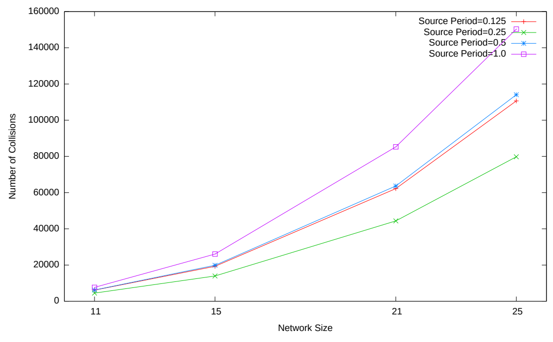
(a) Capture Rates



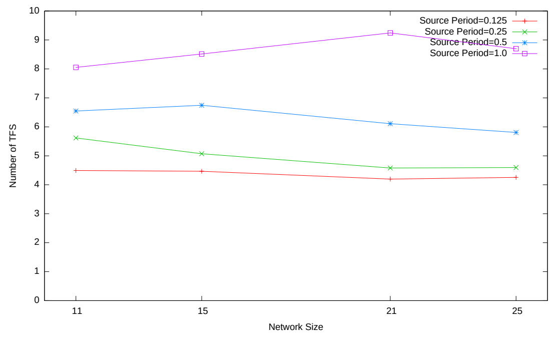
(b) Fake Messages Sent



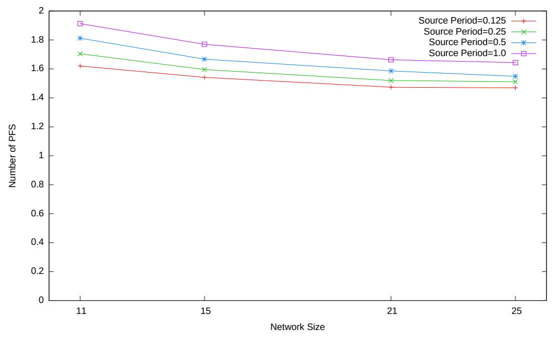
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

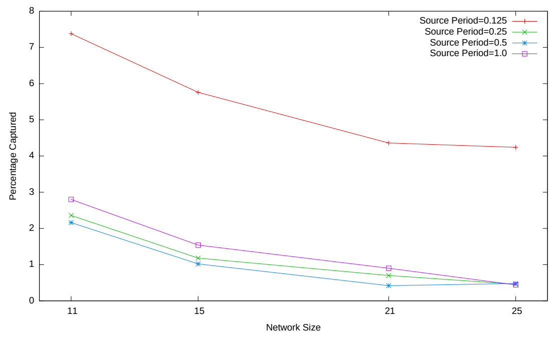


(f) Number of Permanent Fake Sources created

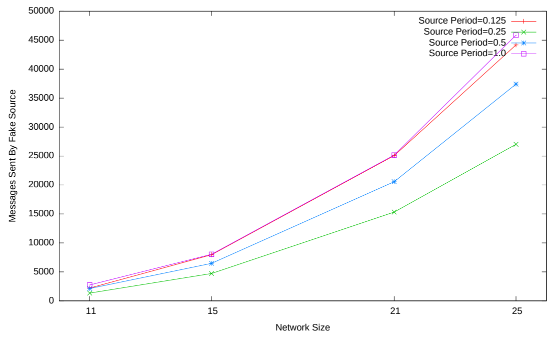
Figure 80: Results for the FurtherSinkCorner configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	7.38	2210	8
	0.25	2.36	1340	12
	0.5	2.16	2129	21
	1.0	2.80	2739	50
15	0.125	5.76	7958	12
	0.25	1.18	4719	9
	0.5	1.02	6476	23
	1.0	1.54	8040	55
21	0.125	4.36	25068	16
	0.25	0.70	15324	6
	0.5	0.42	20569	17
	1.0	0.90	25173	48
25	0.125	4.24	44185	18
	0.25	0.46	27028	6
	0.5	0.48	37413	14
	1.0	0.44	45871	47

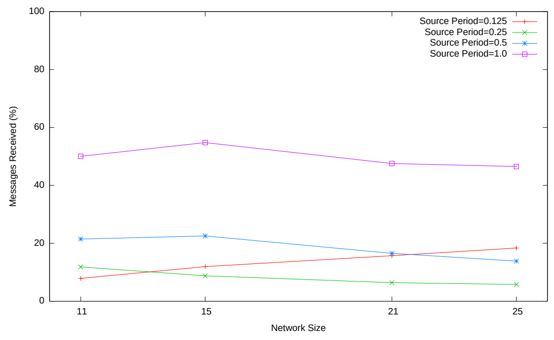
Table 48: Adaptive results for the Generic1 configuration



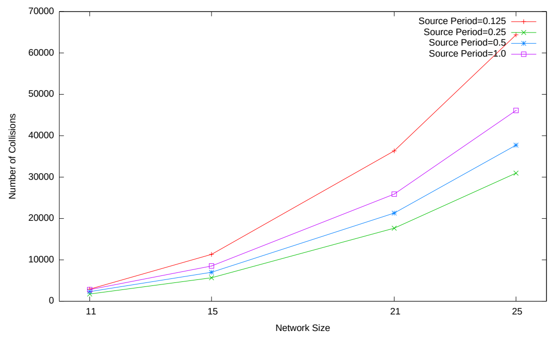
(a) Capture Rates



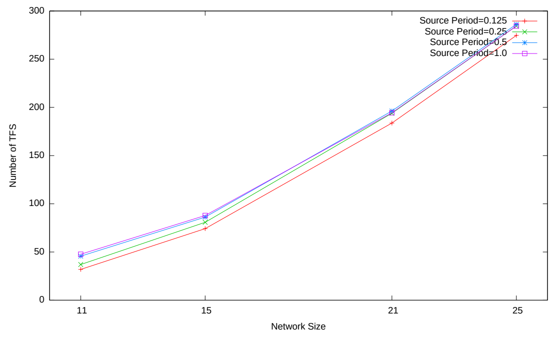
(b) Fake Messages Sent



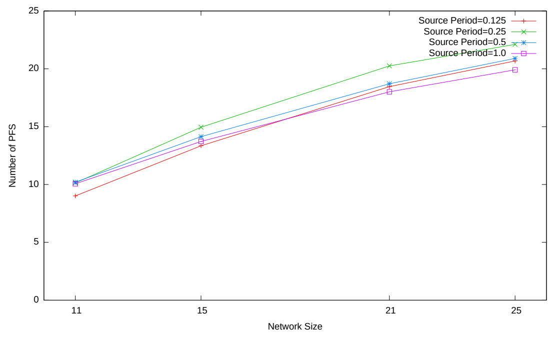
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

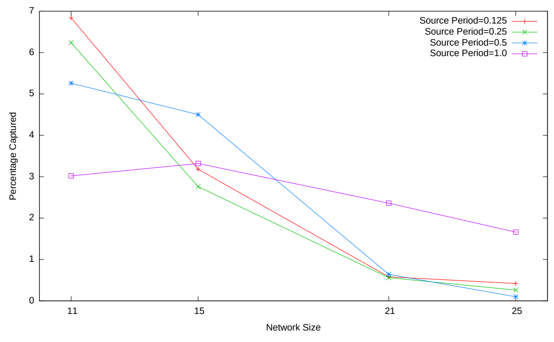


(f) Number of Permanent Fake Sources created

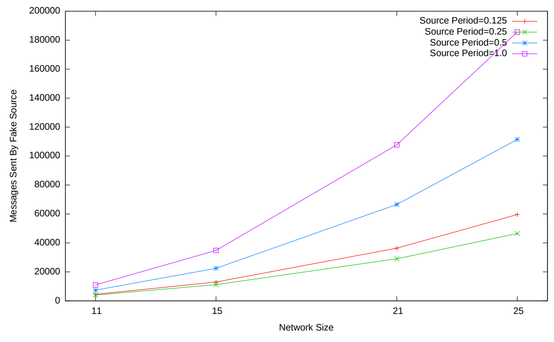
Figure 81: Results for the Generic1 configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	6.84	4519	4
	0.25	6.24	3971	9
	0.5	5.26	7357	18
	1.0	3.02	11006	35
15	0.125	3.18	13016	4
	0.25	2.76	11203	8
	0.5	4.50	22486	16
	1.0	3.32	34830	34
21	0.125	0.58	36372	3
	0.25	0.56	29069	8
	0.5	0.64	66549	15
	1.0	2.36	107710	34
25	0.125	0.42	59569	3
	0.25	0.26	46469	8
	0.5	0.10	111453	15
	1.0	1.66	185607	34

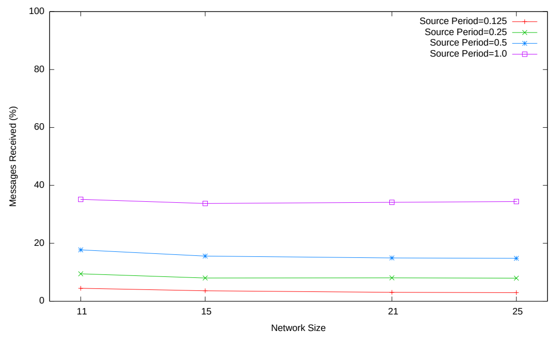
Table 49: Adaptive results for the Generic2 configuration



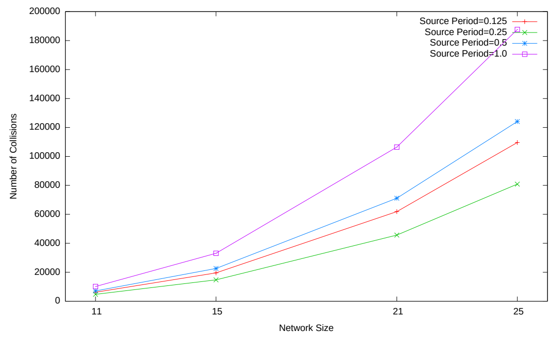
(a) Capture Rates



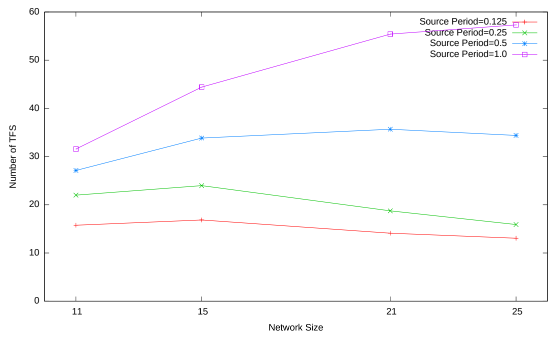
(b) Fake Messages Sent



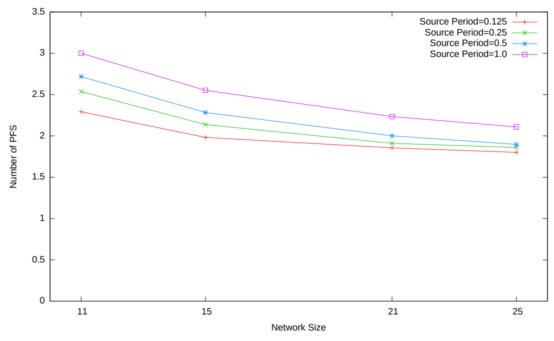
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

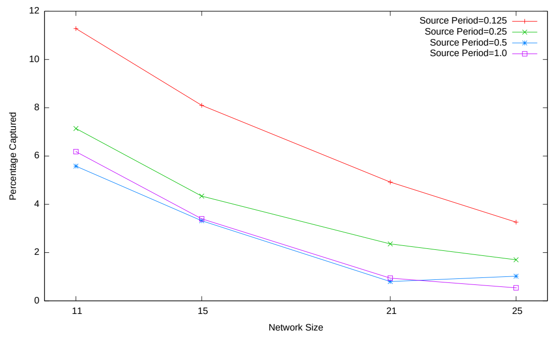


(f) Number of Permanent Fake Sources created

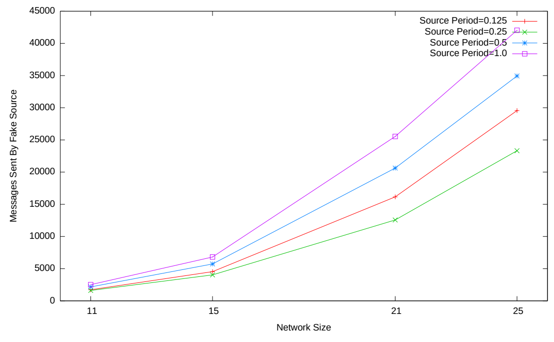
Figure 82: Results for the Generic2 configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	11.28	1724	11
	0.25	7.14	1616	18
	0.5	5.58	2160	32
	1.0	6.18	2509	49
15	0.125	8.10	4535	11
	0.25	4.34	4037	16
	0.5	3.32	5734	33
	1.0	3.40	6816	53
21	0.125	4.92	16146	9
	0.25	2.36	12570	12
	0.5	0.80	20625	21
	1.0	0.94	25534	47
25	0.125	3.26	29564	10
	0.25	1.70	23325	11
	0.5	1.02	34942	24
	1.0	0.54	42036	51

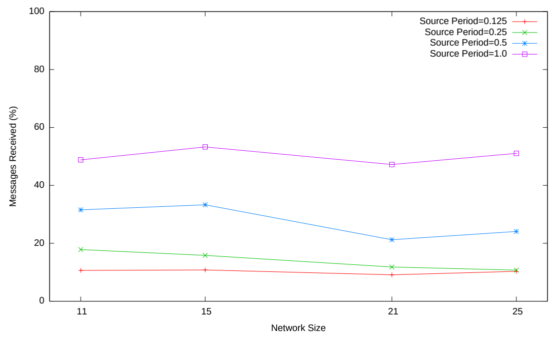
Table 50: Adaptive results for the CircleSinkCentre configuration



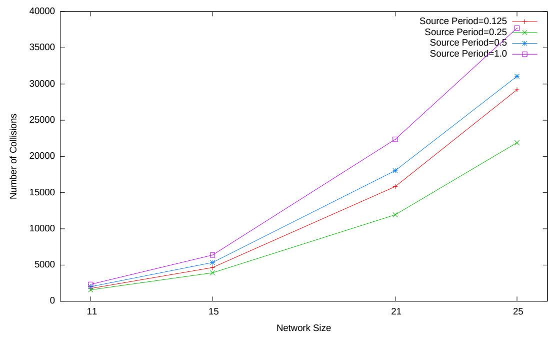
(a) Capture Rates



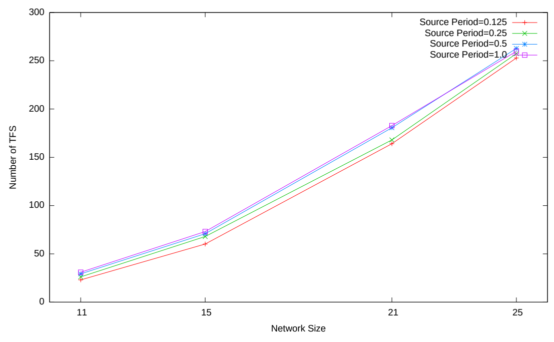
(b) Fake Messages Sent



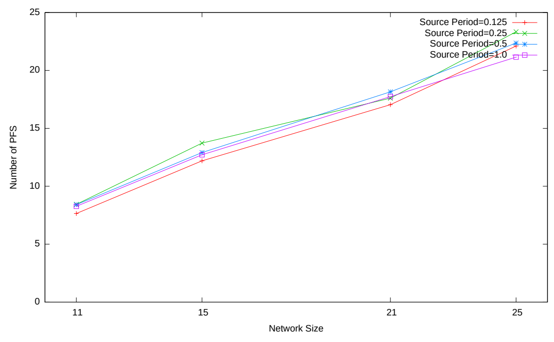
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

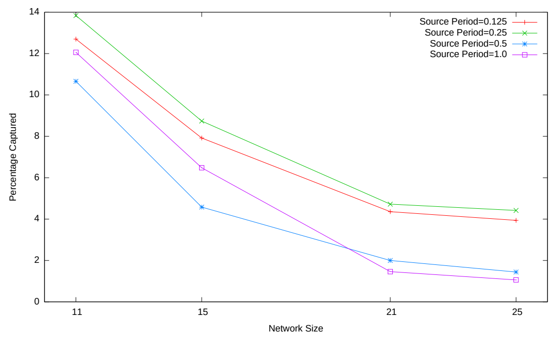


(f) Number of Permanent Fake Sources created

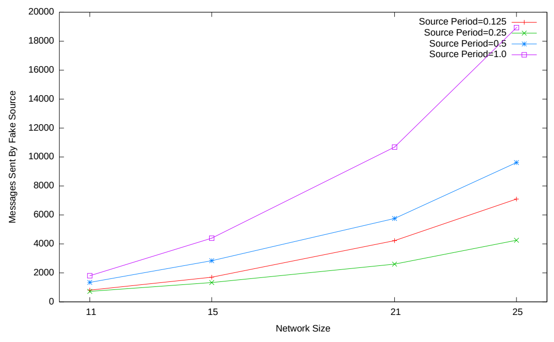
Figure 83: Results for the CircleSinkCentre configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	12.70	805	6
	0.25	13.84	724	16
	0.5	10.66	1343	31
	1.0	12.06	1803	49
15	0.125	7.92	1702	5
	0.25	8.74	1330	14
	0.5	4.58	2841	29
	1.0	6.48	4395	47
21	0.125	4.36	4224	5
	0.25	4.72	2602	12
	0.5	2.00	5758	25
	1.0	1.46	10689	43
25	0.125	3.94	7100	4
	0.25	4.42	4250	12
	0.5	1.44	9618	25
	1.0	1.06	18933	43

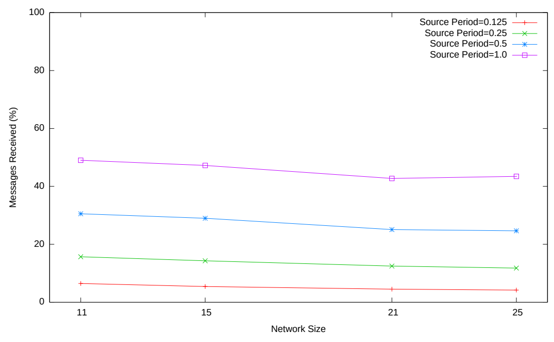
Table 51: Adaptive results for the CircleSourceCentre configuration



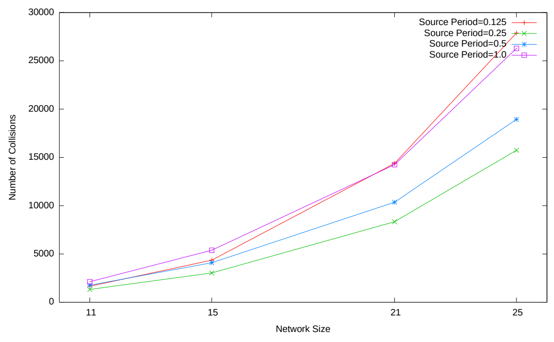
(a) Capture Rates



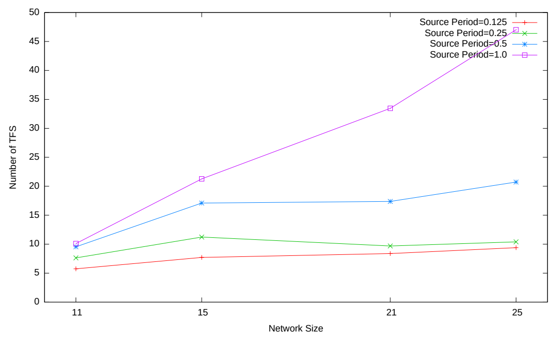
(b) Fake Messages Sent



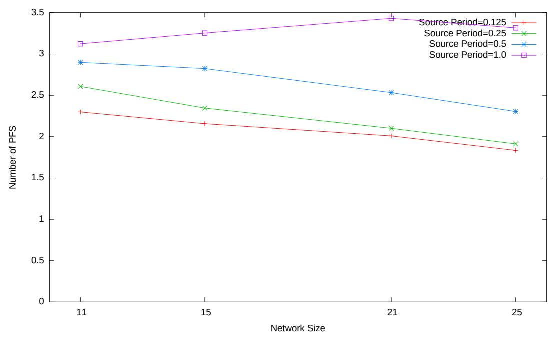
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

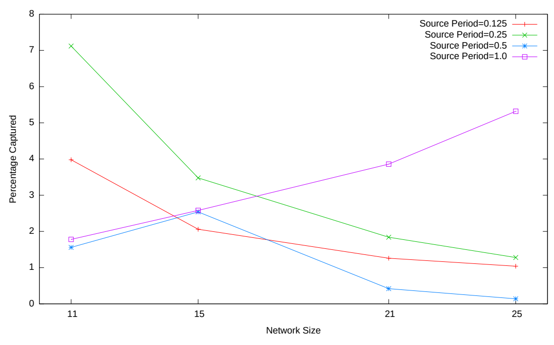


(f) Number of Permanent Fake Sources created

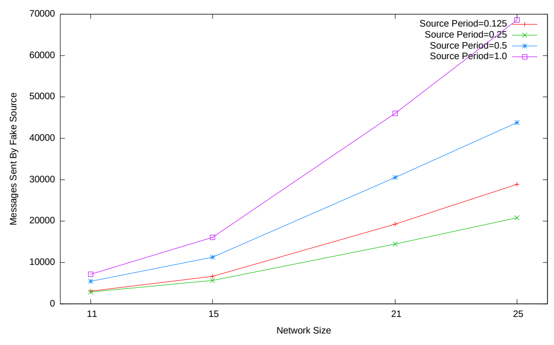
Figure 84: Results for the CircleSourceCentre configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	3.98	3065	4
	0.25	7.12	2860	10
	0.5	1.56	5456	18
	1.0	1.78	7145	32
15	0.125	2.06	6652	3
	0.25	3.48	5653	10
	0.5	2.54	11264	19
	1.0	2.58	16063	36
21	0.125	1.26	19261	3
	0.25	1.84	14447	10
	0.5	0.42	30548	19
	1.0	3.86	46053	39
25	0.125	1.04	28889	3
	0.25	1.28	20790	10
	0.5	0.14	43809	20
	1.0	5.32	68612	41

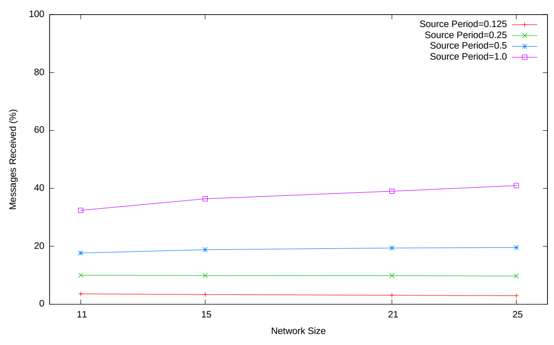
Table 52: Adaptive results for the CircleEdges configuration



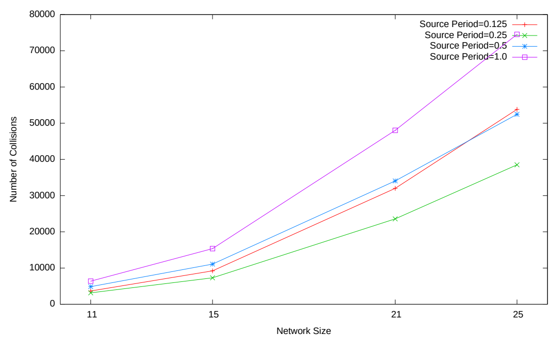
(a) Capture Rates



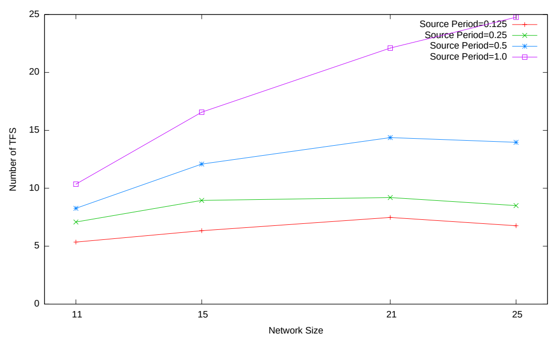
(b) Fake Messages Sent



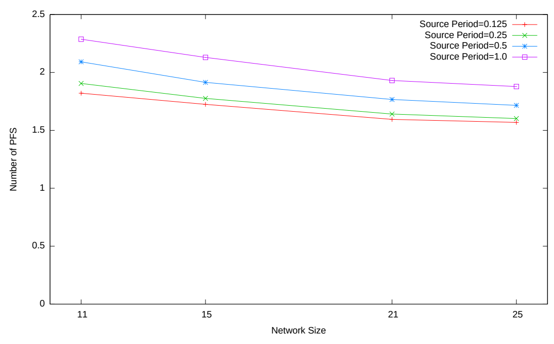
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

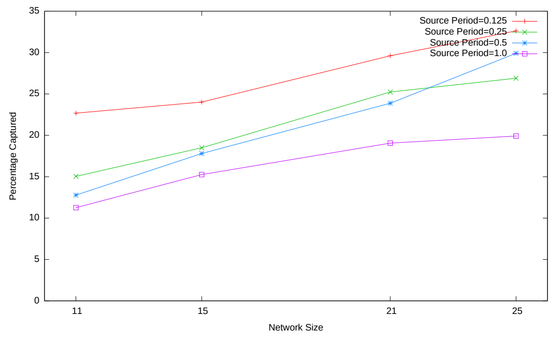


(f) Number of Permanent Fake Sources created

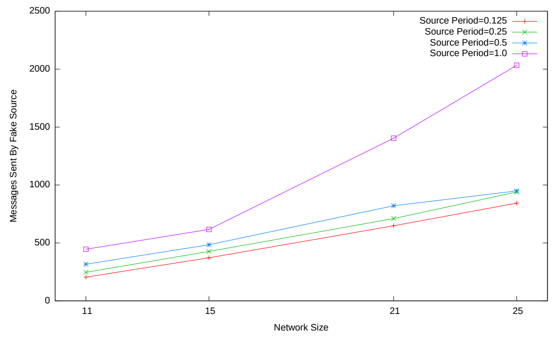
Figure 85: Results for the CircleEdges configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	22.68	204	15
	0.25	15.04	246	19
	0.5	12.78	316	26
	1.0	11.26	444	51
15	0.125	24.02	371	15
	0.25	18.50	426	19
	0.5	17.80	483	27
	1.0	15.26	616	42
21	0.125	29.62	647	18
	0.25	25.24	710	21
	0.5	23.88	820	26
	1.0	19.06	1403	34
25	0.125	32.62	842	20
	0.25	26.90	939	21
	0.5	29.92	948	27
	1.0	19.92	2033	31

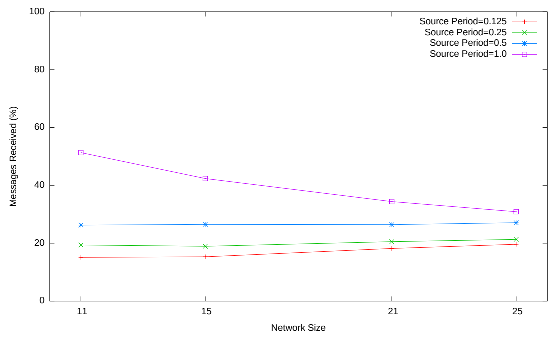
Table 53: Adaptive results for the RingTop configuration



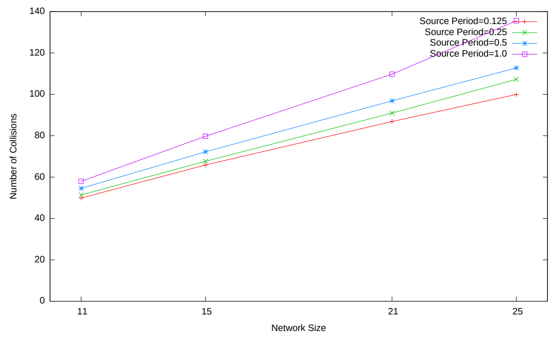
(a) Capture Rates



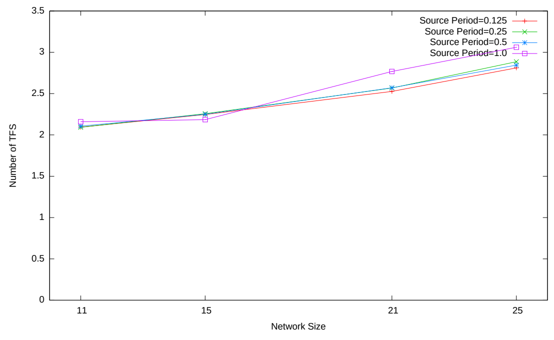
(b) Fake Messages Sent



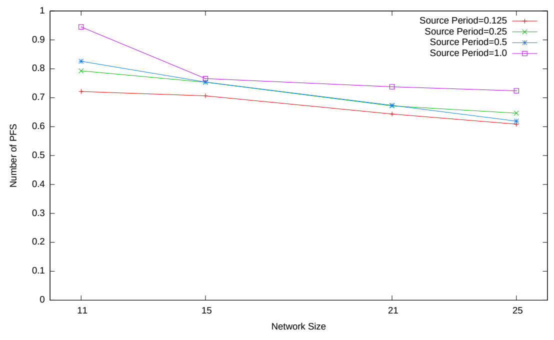
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

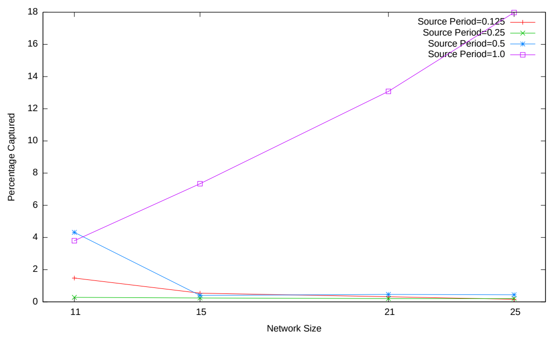


(f) Number of Permanent Fake Sources created

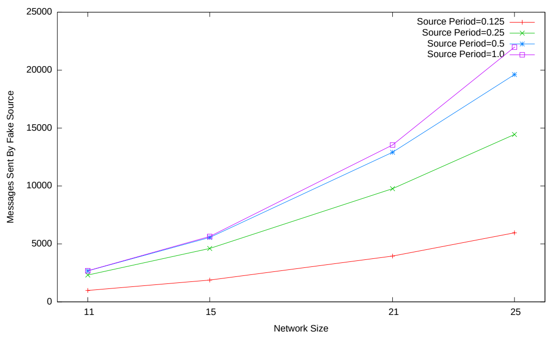
Figure 86: Results for the RingTop configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	1.48	983	2
	0.25	0.28	2314	2
	0.5	4.32	2676	3
	1.0	3.80	2679	13
15	0.125	0.54	1876	2
	0.25	0.24	4603	2
	0.5	0.40	5559	2
	1.0	7.34	5642	9
21	0.125	0.32	3954	1
	0.25	0.20	9767	1
	0.5	0.46	12911	1
	1.0	13.08	13541	6
25	0.125	0.16	5956	1
	0.25	0.20	14448	1
	0.5	0.44	19614	1
	1.0	17.98	21982	5

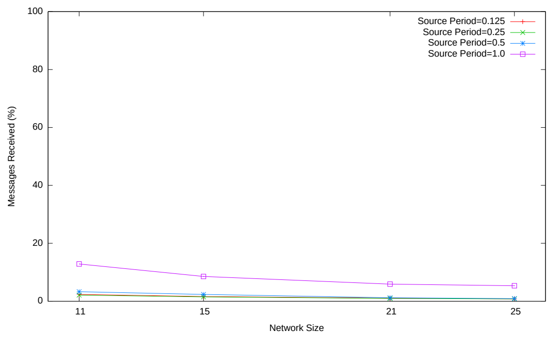
Table 54: Adaptive results for the RingMiddle configuration



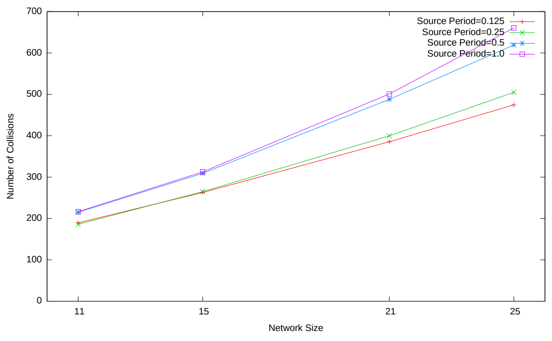
(a) Capture Rates



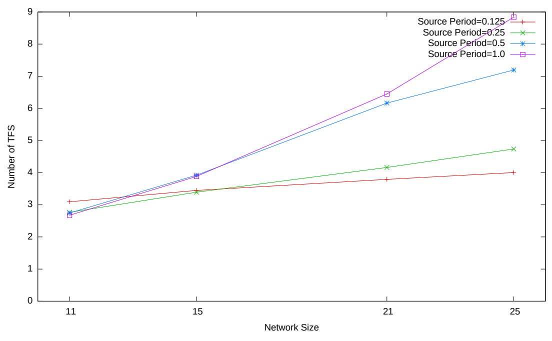
(b) Fake Messages Sent



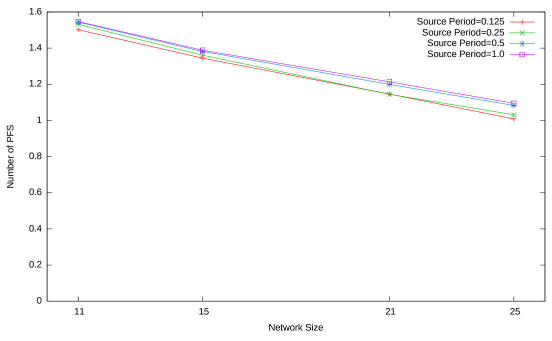
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created

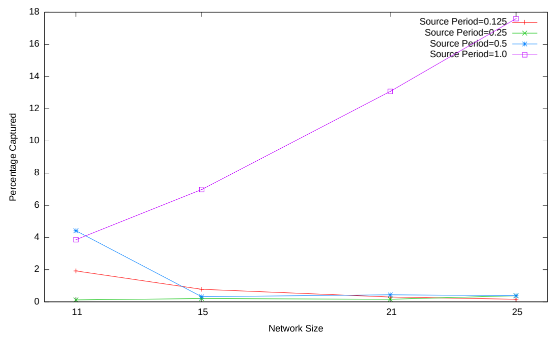


(f) Number of Permanent Fake Sources created

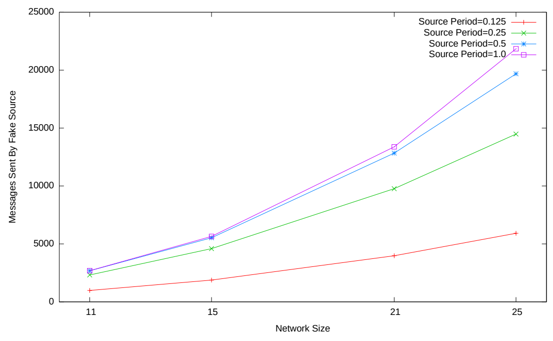
Figure 87: Results for the RingMiddle configuration

Size	Source Period (seconds)	Captured (%)	Fake Messages	Received (%)
11	0.125	1.92	981	2
	0.25	0.12	2322	2
	0.5	4.42	2682	3
	1.0	3.86	2686	13
15	0.125	0.78	1875	2
	0.25	0.20	4593	1
	0.5	0.32	5530	2
	1.0	6.98	5649	8
21	0.125	0.30	3978	1
	0.25	0.16	9765	1
	0.5	0.44	12839	1
	1.0	13.08	13379	6
25	0.125	0.16	5916	1
	0.25	0.38	14479	1
	0.5	0.38	19681	1
	1.0	17.60	21844	5

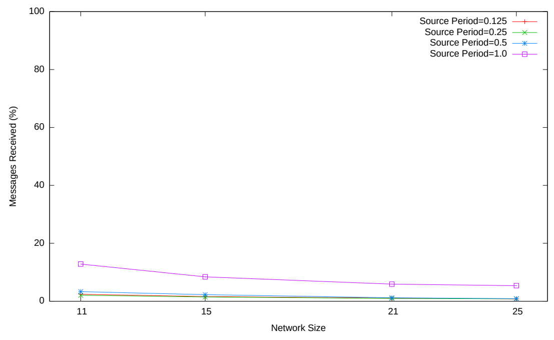
Table 55: Adaptive results for the RingOpposite configuration



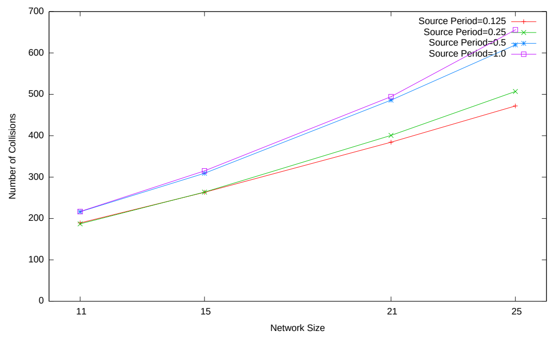
(a) Capture Rates



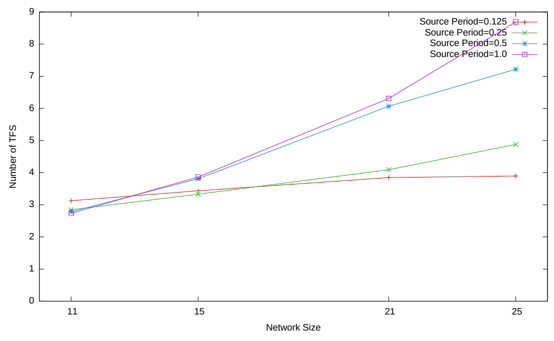
(b) Fake Messages Sent



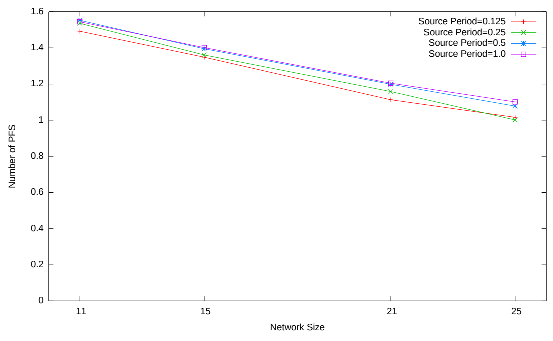
(c) Percentage of Messages Received



(d) Number of Collisions



(e) Number of Temporary Fake Sources created



(f) Number of Permanent Fake Sources created

Figure 88: Results for the RingOpposite configuration

H Adaptive Comparison Tables

Heading Meanings

P_{src}	The Source period
P_{fs}	The Fake Source period
TFS	The probability that a temporary fake source is created
PFS	The probability that a permanent fake source is created
D	The duration of a temporary fake source

H.1 SourceCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-2.50 (-28%)	-626 (-15%)	+4 (+36%)
0.125	0.0625	80	100	2	+1.18 (+16%)	-814 (-18%)	+6 (+74%)
0.125	0.0625	80	100	4	+5.24 (+102%)	-869 (-20%)	+8 (+110%)
0.125	0.0625	90	100	4	+6.04 (+127%)	-1082 (-24%)	+9 (+126%)
0.125	0.0625	100	100	4	+6.52 (+145%)	-1252 (-27%)	+9 (+138%)
0.25	0.125	80	100	1	-1.82 (-33%)	-382 (-9%)	+4 (+28%)
0.25	0.125	80	100	4	+2.98 (+98%)	-441 (-11%)	+10 (+90%)
0.25	0.125	90	100	4	+3.46 (+123%)	-597 (-14%)	+11 (+105%)
0.25	0.125	100	100	4	+3.52 (+127%)	-716 (-17%)	+12 (+114%)
0.25	0.0625	90	100	4	+3.94 (+153%)	-1117 (-25%)	+13 (+123%)
0.25	0.0625	100	100	4	+4.24 (+175%)	-1247 (-28%)	+13 (+131%)
0.5	0.25	80	100	1	-3.88 (-69%)	-1880 (-31%)	+12 (+42%)
0.5	0.25	90	100	1	+0.88 (+27%)	-2535 (-40%)	+16 (+57%)
0.5	0.25	100	100	1	+2.02 (+75%)	-2776 (-43%)	+17 (+63%)
0.5	0.25	100	100	2	+3.32 (+161%)	-4133 (-58%)	+25 (+113%)
0.5	0.25	80	100	4	+2.08 (+78%)	-3016 (-46%)	+24 (+103%)
0.5	0.25	90	100	4	+2.88 (+126%)	-3421 (-51%)	+26 (+116%)
0.5	0.25	100	100	4	+3.18 (+149%)	-3724 (-54%)	+27 (+124%)
0.5	0.125	100	100	4	+3.34 (+163%)	-4743 (-64%)	+30 (+145%)
1.0	0.5	80	100	1	-4.18 (-73%)	-2319 (-33%)	+7 (+15%)
1.0	0.5	90	100	1	+0.64 (+19%)	-2972 (-40%)	+9 (+18%)
1.0	0.5	100	100	1	+1.46 (+50%)	-3132 (-42%)	+9 (+19%)
1.0	0.5	90	100	2	+2.14 (+83%)	-6018 (-68%)	+21 (+48%)
1.0	0.5	100	100	2	+2.34 (+95%)	-6250 (-69%)	+21 (+50%)
1.0	0.5	90	100	4	+3.04 (+143%)	-9201 (-88%)	+34 (+93%)
1.0	0.25	100	100	1	+2.74 (+121%)	-8441 (-84%)	+28 (+69%)
1.0	0.25	100	100	4	+3.40 (+175%)	-13302 (-106%)	+44 (+137%)
1.0	0.125	100	100	2	+3.06 (+145%)	-13095 (-105%)	+39 (+114%)

Table 56: Adaptive: Results for networks of size 11 and configuration SourceCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-4.70 (-76%)	-1705 (-14%)	+4 (+50%)
0.125	0.0625	80	100	4	+1.42 (+46%)	-1948 (-16%)	+8 (+118%)
0.125	0.0625	90	100	4	+2.76 (+113%)	-2658 (-21%)	+9 (+141%)
0.125	0.0625	100	100	4	+3.32 (+154%)	-3148 (-24%)	+10 (+154%)
0.25	0.125	80	100	4	+0.16 (+11%)	-410 (-4%)	+8 (+81%)
0.25	0.125	90	100	4	+1.04 (+100%)	-938 (-8%)	+9 (+105%)
0.25	0.125	100	100	4	+1.40 (+163%)	-1312 (-11%)	+10 (+120%)
0.25	0.0625	100	100	4	+1.48 (+180%)	-2283 (-18%)	+11 (+137%)
0.5	0.25	80	100	1	-4.50 (-137%)	-4655 (-27%)	+14 (+54%)
0.5	0.25	90	100	1	-0.72 (-51%)	-6025 (-33%)	+17 (+69%)
0.5	0.25	100	100	1	+0.40 (+48%)	-6248 (-34%)	+17 (+68%)

0.5	0.25	90	100	4	+0.74 (+110%)	-8775 (-45%)	+27 (+134%)
0.5	0.25	100	100	4	+0.92 (+159%)	-9558 (-48%)	+28 (+142%)
0.5	0.125	100	100	4	+0.98 (+178%)	-11585 (-55%)	+30 (+160%)
1.0	0.5	80	100	1	-7.10 (-154%)	-6652 (-32%)	+12 (+23%)
1.0	0.5	100	100	1	+0.02 (+2%)	-8055 (-38%)	+12 (+25%)
1.0	0.5	100	100	2	+0.76 (+112%)	-16678 (-65%)	+25 (+57%)
1.0	0.5	90	100	4	+0.94 (+159%)	-27762 (-89%)	+42 (+121%)
1.0	0.125	100	100	4	+0.98 (+172%)	-40880 (-108%)	+50 (+167%)

Table 57: Adaptive: Results for networks of size 15 and configuration SourceCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-6.24 (-129%)	-4718 (-12%)	+4 (+53%)
0.125	0.0625	80	100	4	-0.32 (-17%)	-4769 (-12%)	+8 (+123%)
0.125	0.0625	90	100	4	+1.24 (+113%)	-7163 (-18%)	+9 (+154%)
0.125	0.0625	100	100	4	+1.60 (+174%)	-8644 (-21%)	+9 (+165%)
0.25	0.125	80	100	4	-0.46 (-41%)	-288 (-1%)	+7 (+84%)
0.25	0.125	90	100	4	+0.60 (+103%)	-1944 (-6%)	+9 (+116%)
0.25	0.125	100	100	4	+0.80 (+167%)	-2949 (-8%)	+9 (+131%)
0.25	0.0625	100	100	4	+0.84 (+183%)	-4819 (-13%)	+10 (+150%)
0.5	0.25	80	100	1	-4.84 (-162%)	-10763 (-20%)	+16 (+67%)
0.5	0.25	100	100	1	+0.36 (+95%)	-13556 (-25%)	+18 (+75%)
0.5	0.25	90	100	4	+0.50 (+161%)	-22933 (-39%)	+28 (+151%)
0.5	0.25	100	100	4	+0.54 (+186%)	-24827 (-41%)	+29 (+157%)
1.0	0.5	80	100	1	-7.64 (-179%)	-17578 (-28%)	+17 (+35%)
1.0	0.5	100	100	1	+0.22 (+67%)	-19239 (-30%)	+18 (+36%)
1.0	0.5	100	100	2	+0.28 (+93%)	-44075 (-57%)	+30 (+70%)
1.0	0.5	90	100	4	+0.32 (+114%)	-81880 (-85%)	+49 (+145%)
1.0	0.5	100	100	4	+0.44 (+200%)	-84566 (-87%)	+49 (+149%)

Table 58: Adaptive: Results for networks of size 21 and configuration SourceCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	-0.54 (-34%)	-7377 (-11%)	+7 (+125%)
0.125	0.0625	90	100	4	+0.82 (+92%)	-11389 (-16%)	+8 (+156%)
0.125	0.0625	100	100	4	+1.24 (+182%)	-14046 (-19%)	+8 (+170%)
0.25	0.125	100	100	2	+0.70 (+200%)	-9649 (-15%)	+10 (+142%)
0.25	0.125	80	100	4	-0.36 (-41%)	-1154 (-2%)	+8 (+100%)
0.25	0.0625	100	100	4	+0.68 (+189%)	-8047 (-13%)	+10 (+159%)
0.25	0.125	90	100	4	+0.52 (+118%)	-3909 (-6%)	+9 (+131%)
0.25	0.125	100	100	4	+0.62 (+159%)	-5359 (-9%)	+10 (+143%)
0.5	0.25	90	100	4	+0.32 (+178%)	-35620 (-34%)	+26 (+156%)
0.5	0.25	100	100	4	+0.34 (+200%)	-38412 (-36%)	+27 (+162%)
0.5	0.25	80	100	1	-4.40 (-173%)	-15266 (-16%)	+16 (+72%)
0.5	0.25	100	100	1	+0.24 (+109%)	-17866 (-18%)	+16 (+73%)
1.0	0.5	100	100	1	-0.04 (-15%)	-27608 (-24%)	+19 (+38%)
1.0	0.5	100	100	2	+0.16 (+100%)	-70920 (-52%)	+31 (+73%)
1.0	0.5	90	100	4	+0.24 (+200%)	-140540 (-82%)	+51 (+156%)

Table 59: Adaptive: Results for networks of size 25 and configuration SourceCorner

H.2 SinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-5.46 (-57%)	-1174 (-58%)	-1 (-19%)
0.125	0.0625	90	100	1	-2.02 (-26%)	-1337 (-64%)	+0 (+2%)
0.125	0.0625	100	100	1	+1.20 (+19%)	-1520 (-70%)	+1 (+25%)

0.125	0.0625	80	100	2	-3.42 (-40%)	-1228 (-60%)	-1 (-12%)
0.125	0.0625	90	100	2	+0.94 (+15%)	-1432 (-67%)	+1 (+16%)
0.125	0.0625	100	100	2	+3.32 (+64%)	-1585 (-72%)	+2 (+40%)
0.125	0.0625	80	100	4	-0.96 (-13%)	-1344 (-64%)	+0 (+9%)
0.125	0.0625	90	100	4	+3.22 (+62%)	-1548 (-70%)	+2 (+42%)
0.125	0.0625	100	100	4	+5.04 (+117%)	-1690 (-75%)	+3 (+68%)
0.25	0.125	80	100	1	+0.26 (+2%)	-1282 (-71%)	+3 (+21%)
0.25	0.125	80	100	2	+2.62 (+25%)	-1338 (-73%)	+4 (+30%)
0.25	0.125	90	100	2	+6.90 (+82%)	-1495 (-78%)	+6 (+49%)
0.25	0.125	80	100	4	+5.78 (+64%)	-1439 (-76%)	+6 (+54%)
0.25	0.125	90	100	4	+8.76 (+117%)	-1592 (-81%)	+8 (+76%)
0.25	0.125	100	100	4	+9.96 (+144%)	-1696 (-84%)	+9 (+92%)
0.25	0.0625	100	100	1	+10.18 (+150%)	-1962 (-91%)	+9 (+84%)
0.25	0.0625	100	100	2	+10.24 (+151%)	-1972 (-92%)	+9 (+91%)
0.25	0.0625	100	100	4	+11.48 (+187%)	-2068 (-94%)	+11 (+113%)
0.5	0.25	80	100	1	-4.14 (-54%)	-2603 (-73%)	+10 (+42%)
0.5	0.25	90	100	1	+0.42 (+8%)	-2898 (-78%)	+12 (+52%)
0.5	0.25	100	100	1	+2.10 (+47%)	-3042 (-80%)	+12 (+55%)
0.5	0.25	90	100	2	+0.00 (+0%)	-2877 (-77%)	+12 (+52%)
0.5	0.25	100	100	2	+2.14 (+48%)	-3057 (-80%)	+13 (+57%)
0.5	0.25	80	100	4	-2.36 (-35%)	-2743 (-75%)	+12 (+53%)
0.5	0.25	90	100	4	+1.54 (+32%)	-3025 (-80%)	+14 (+64%)
0.5	0.25	100	100	4	+2.70 (+64%)	-3179 (-82%)	+15 (+69%)
0.5	0.125	100	100	1	+3.54 (+94%)	-4049 (-94%)	+19 (+95%)
0.5	0.125	100	100	2	+3.46 (+91%)	-4017 (-94%)	+19 (+97%)
0.5	0.125	100	100	4	+4.02 (+114%)	-4152 (-95%)	+21 (+111%)
1.0	0.5	80	100	1	-6.98 (-79%)	-3285 (-67%)	+11 (+29%)
1.0	0.5	90	100	1	-2.90 (-43%)	-3650 (-72%)	+12 (+33%)
1.0	0.5	100	100	1	-1.28 (-21%)	-3920 (-75%)	+13 (+35%)
1.0	0.5	80	100	2	-7.74 (-84%)	-3072 (-64%)	+10 (+27%)
1.0	0.5	90	100	2	-3.68 (-51%)	-3502 (-70%)	+12 (+31%)
1.0	0.5	100	100	2	-2.34 (-36%)	-3698 (-72%)	+13 (+35%)
1.0	0.25	90	100	1	+1.44 (+31%)	-8412 (-113%)	+24 (+79%)
1.0	0.25	90	100	2	+0.96 (+20%)	-8221 (-112%)	+24 (+78%)
1.0	0.25	100	100	2	+3.66 (+104%)	-8696 (-114%)	+25 (+83%)
1.0	0.25	90	100	4	+1.06 (+22%)	-8387 (-113%)	+25 (+84%)
1.0	0.125	100	100	1	+4.20 (+130%)	-10692 (-124%)	+31 (+117%)
1.0	0.125	100	100	2	+4.02 (+121%)	-10591 (-124%)	+31 (+117%)
1.0	0.125	100	100	4	+4.04 (+122%)	-10640 (-124%)	+32 (+122%)

Table 60: Adaptive: Results for networks of size 11 and configuration SinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-6.48 (-87%)	-3689 (-73%)	-2 (-37%)
0.125	0.0625	90	100	1	-1.30 (-27%)	-4300 (-80%)	-0 (-5%)
0.125	0.0625	100	100	1	+0.86 (+23%)	-4614 (-83%)	+1 (+17%)
0.125	0.0625	80	100	2	-5.08 (-75%)	-3799 (-74%)	-2 (-34%)
0.125	0.0625	90	100	2	-0.70 (-15%)	-4356 (-81%)	-0 (-2%)
0.125	0.0625	100	100	2	+1.02 (+28%)	-4620 (-83%)	+1 (+20%)
0.125	0.0625	80	100	4	-3.74 (-62%)	-4089 (-78%)	-1 (-19%)
0.125	0.0625	90	100	4	+0.34 (+8%)	-4554 (-83%)	+0 (+11%)
0.125	0.0625	100	100	4	+2.68 (+94%)	-4941 (-87%)	+1 (+40%)
0.25	0.125	80	100	1	-1.48 (-17%)	-3856 (-88%)	+1 (+8%)
0.25	0.125	90	100	1	+2.52 (+37%)	-4234 (-93%)	+3 (+26%)
0.25	0.125	100	100	1	+3.82 (+63%)	-4426 (-95%)	+4 (+36%)
0.25	0.125	80	100	2	-0.94 (-11%)	-3936 (-89%)	+1 (+12%)
0.25	0.125	90	100	2	+3.44 (+55%)	-4293 (-93%)	+3 (+32%)
0.25	0.125	100	100	2	+5.12 (+94%)	-4522 (-96%)	+4 (+43%)
0.25	0.125	80	100	4	+1.56 (+22%)	-4173 (-92%)	+3 (+31%)
0.25	0.125	90	100	4	+5.30 (+99%)	-4634 (-97%)	+5 (+57%)

0.25	0.125	100	100	4	+6.76 (+147%)	-4821 (-99%)	+6 (+69%)
0.25	0.0625	100	100	2	+6.88 (+152%)	-5070 (-102%)	+7 (+82%)
0.25	0.0625	100	100	4	+7.24 (+166%)	-5346 (-104%)	+8 (+101%)
0.5	0.25	80	100	1	-5.38 (-102%)	-8329 (-87%)	+8 (+39%)
0.5	0.25	100	100	1	+1.26 (+65%)	-9521 (-93%)	+11 (+56%)
0.5	0.25	80	100	2	-5.46 (-103%)	-8283 (-87%)	+8 (+38%)
0.5	0.25	90	100	2	-0.52 (-18%)	-9132 (-91%)	+10 (+50%)
0.5	0.25	100	100	2	+1.18 (+60%)	-9440 (-93%)	+10 (+53%)
0.5	0.25	80	100	4	-3.86 (-86%)	-8657 (-89%)	+9 (+47%)
0.5	0.25	90	100	4	+0.48 (+21%)	-9395 (-93%)	+11 (+58%)
0.5	0.25	100	100	4	+1.60 (+91%)	-9662 (-94%)	+11 (+61%)
0.5	0.125	100	100	2	+1.76 (+105%)	-11252 (-102%)	+15 (+92%)
0.5	0.125	100	100	4	+1.88 (+116%)	-11353 (-102%)	+16 (+98%)
1.0	0.5	80	100	1	-4.74 (-86%)	-10922 (-82%)	+0 (+0%)
1.0	0.5	100	100	1	+1.06 (+40%)	-13002 (-91%)	+3 (+7%)
1.0	0.5	80	100	2	-4.92 (-88%)	-10539 (-81%)	-1 (-2%)
1.0	0.5	90	100	2	-0.40 (-12%)	-11693 (-86%)	+0 (+1%)
1.0	0.5	100	100	2	+0.64 (+23%)	-12271 (-88%)	+1 (+2%)
1.0	0.25	90	100	2	+1.14 (+44%)	-24671 (-123%)	+22 (+82%)
1.0	0.25	100	100	2	+2.72 (+151%)	-25250 (-124%)	+23 (+83%)
1.0	0.25	100	100	4	+2.78 (+157%)	-25461 (-124%)	+23 (+87%)
1.0	0.125	100	100	4	+2.82 (+161%)	-28908 (-130%)	+29 (+119%)

Table 61: Adaptive: Results for networks of size 15 and configuration SinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	-5.90 (-113%)	-11670 (-85%)	-2 (-50%)
0.125	0.0625	90	100	2	-1.52 (-50%)	-12983 (-90%)	-1 (-18%)
0.125	0.0625	100	100	2	+0.30 (+14%)	-13965 (-94%)	+0 (+7%)
0.125	0.0625	80	100	4	-5.70 (-112%)	-11761 (-86%)	-2 (-46%)
0.125	0.0625	100	100	4	+0.62 (+32%)	-14270 (-95%)	+0 (+17%)
0.25	0.125	90	100	1	-0.04 (-1%)	-12065 (-103%)	+1 (+10%)
0.25	0.125	80	100	2	-4.04 (-68%)	-10912 (-98%)	-1 (-10%)
0.25	0.125	90	100	2	-0.14 (-3%)	-11993 (-103%)	+1 (+9%)
0.25	0.125	100	100	2	+1.70 (+55%)	-12545 (-105%)	+2 (+19%)
0.25	0.125	80	100	4	-2.56 (-49%)	-11428 (-101%)	+0 (+2%)
0.25	0.125	90	100	4	+1.34 (+41%)	-12432 (-105%)	+2 (+23%)
0.25	0.125	100	100	4	+2.38 (+87%)	-13082 (-107%)	+3 (+38%)
0.25	0.0625	100	100	2	+3.10 (+130%)	-13653 (-110%)	+5 (+72%)
0.25	0.0625	100	100	4	+3.28 (+143%)	-14079 (-111%)	+6 (+85%)
0.5	0.25	90	100	1	-1.04 (-58%)	-26396 (-101%)	+4 (+23%)
0.5	0.25	100	100	1	+0.82 (+96%)	-27411 (-102%)	+5 (+28%)
0.5	0.25	80	100	2	-5.38 (-136%)	-24072 (-96%)	+2 (+9%)
0.5	0.25	90	100	2	-1.18 (-64%)	-26208 (-100%)	+4 (+22%)
0.5	0.25	100	100	2	+0.74 (+83%)	-27112 (-102%)	+5 (+25%)
0.5	0.25	90	100	4	-0.64 (-41%)	-26570 (-101%)	+5 (+24%)
0.5	0.25	100	100	4	+0.88 (+107%)	-27468 (-103%)	+5 (+29%)
0.5	0.125	100	100	2	+0.90 (+111%)	-31311 (-109%)	+11 (+72%)
0.5	0.125	100	100	4	+0.92 (+115%)	-31548 (-109%)	+12 (+76%)
1.0	0.5	80	100	2	-4.70 (-112%)	-32969 (-89%)	+1 (+3%)
1.0	0.5	90	100	2	+0.08 (+4%)	-36409 (-94%)	+2 (+6%)
1.0	0.5	100	100	2	+1.46 (+132%)	-38239 (-97%)	+3 (+9%)
1.0	0.5	80	100	4	-3.82 (-102%)	-33595 (-90%)	+2 (+4%)
1.0	0.5	90	100	4	+0.14 (+8%)	-36683 (-95%)	+2 (+6%)
1.0	0.5	100	100	4	+1.44 (+129%)	-38045 (-96%)	+3 (+8%)
1.0	0.25	100	100	4	+1.72 (+176%)	-72499 (-128%)	+19 (+66%)

Table 62: Adaptive: Results for networks of size 21 and configuration SinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	-6.32 (-133%)	-20835 (-91%)	-2 (-54%)
0.125	0.0625	90	100	2	-1.72 (-70%)	-23339 (-96%)	-1 (-18%)
0.125	0.0625	100	100	2	+0.32 (+23%)	-24721 (-99%)	+0 (+6%)
0.125	0.0625	80	100	4	-7.06 (-138%)	-20709 (-91%)	-2 (-55%)
0.125	0.0625	90	100	4	-1.74 (-71%)	-23234 (-96%)	-1 (-18%)
0.125	0.0625	100	100	4	+0.78 (+66%)	-24922 (-100%)	+0 (+9%)
0.25	0.125	100	100	2	+1.58 (+81%)	-21029 (-110%)	+1 (+13%)
0.25	0.125	80	100	4	-3.60 (-79%)	-18808 (-104%)	-1 (-10%)
0.25	0.125	90	100	4	+0.48 (+19%)	-20429 (-108%)	+1 (+11%)
0.25	0.125	100	100	4	+1.94 (+110%)	-21552 (-111%)	+2 (+25%)
0.25	0.0625	100	100	1	+2.08 (+122%)	-23963 (-116%)	+5 (+76%)
0.25	0.125	90	100	1	-0.38 (-13%)	-20349 (-108%)	+0 (+0%)
0.25	0.125	80	100	2	-5.10 (-96%)	-17981 (-102%)	-2 (-22%)
0.25	0.125	90	100	2	-0.70 (-23%)	-19632 (-106%)	-0 (-2%)
0.25	0.0625	100	100	2	+2.06 (+120%)	-22985 (-114%)	+5 (+69%)
0.5	0.25	80	100	1	-5.00 (-137%)	-41004 (-100%)	-1 (-5%)
0.5	0.25	80	100	2	-5.78 (-143%)	-40151 (-99%)	-2 (-8%)
0.5	0.25	90	100	2	-0.84 (-53%)	-43617 (-103%)	+0 (+1%)
0.5	0.25	100	100	2	+0.80 (+105%)	-45192 (-105%)	+1 (+5%)
0.5	0.25	80	100	4	-5.12 (-138%)	-40692 (-100%)	-1 (-5%)
0.5	0.25	90	100	4	-0.38 (-28%)	-44266 (-104%)	+1 (+4%)
0.5	0.25	100	100	4	+0.96 (+141%)	-45866 (-106%)	+2 (+9%)
0.5	0.125	100	100	4	+1.04 (+163%)	-54300 (-114%)	+10 (+68%)
1.0	0.5	100	100	1	+0.94 (+154%)	-69399 (-103%)	-1 (-2%)
1.0	0.5	80	100	2	-5.66 (-145%)	-56614 (-93%)	-4 (-10%)
1.0	0.5	90	100	2	-0.88 (-58%)	-62188 (-97%)	-3 (-7%)
1.0	0.5	80	100	4	-4.20 (-132%)	-57845 (-94%)	-4 (-9%)
1.0	0.5	90	100	4	-0.28 (-23%)	-62739 (-98%)	-3 (-7%)
1.0	0.5	100	100	4	+0.90 (+143%)	-65388 (-100%)	-2 (-5%)
1.0	0.25	100	100	4	+1.04 (+186%)	-122867 (-130%)	+14 (+48%)

Table 63: Adaptive: Results for networks of size 25 and configuration SinkCorner

H.3 FurtherSinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-4.52 (-67%)	-1450 (-38%)	-1 (-33%)
0.125	0.0625	90	100	1	-0.60 (-13%)	-1750 (-44%)	-0 (-3%)
0.125	0.0625	80	100	2	-4.80 (-70%)	-1420 (-38%)	-2 (-34%)
0.125	0.0625	90	100	2	-0.62 (-13%)	-1741 (-44%)	-0 (-3%)
0.125	0.0625	100	100	2	+1.36 (+36%)	-1946 (-48%)	+1 (+17%)
0.125	0.0625	80	100	4	-4.18 (-64%)	-1554 (-40%)	-1 (-23%)
0.125	0.0625	90	100	4	+0.38 (+9%)	-1895 (-47%)	+0 (+12%)
0.125	0.0625	100	100	4	+2.60 (+82%)	-2122 (-51%)	+1 (+42%)
0.25	0.125	80	100	1	-4.16 (-46%)	-1418 (-45%)	+1 (+5%)
0.25	0.125	90	100	1	+1.10 (+17%)	-1727 (-52%)	+3 (+29%)
0.25	0.125	100	100	1	+3.14 (+59%)	-1844 (-54%)	+4 (+38%)
0.25	0.125	80	100	2	-3.16 (-37%)	-1471 (-46%)	+1 (+11%)
0.25	0.125	90	100	2	+2.02 (+34%)	-1753 (-52%)	+3 (+33%)
0.25	0.125	100	100	2	+3.94 (+80%)	-1904 (-56%)	+4 (+47%)
0.25	0.125	80	100	4	+0.72 (+11%)	-1693 (-51%)	+3 (+34%)
0.25	0.125	100	100	4	+5.54 (+134%)	-2054 (-59%)	+6 (+71%)
0.25	0.0625	100	100	2	+5.60 (+137%)	-2392 (-65%)	+7 (+81%)
0.25	0.0625	100	100	4	+6.24 (+165%)	-2528 (-68%)	+8 (+99%)
0.5	0.25	80	100	1	-0.94 (-14%)	-3014 (-47%)	+8 (+43%)
0.5	0.25	90	100	1	+3.32 (+69%)	-3365 (-51%)	+9 (+51%)
0.5	0.25	80	100	2	-1.46 (-20%)	-2867 (-45%)	+7 (+40%)

0.5	0.25	90	100	2	+3.26 (+68%)	-3349 (-50%)	+9 (+53%)
0.5	0.25	100	100	2	+5.30 (+140%)	-3571 (-53%)	+10 (+57%)
0.5	0.25	80	100	4	-0.54 (-8%)	-3081 (-47%)	+9 (+49%)
0.5	0.25	90	100	4	+4.08 (+93%)	-3546 (-53%)	+11 (+63%)
0.5	0.25	100	100	4	+5.56 (+152%)	-3725 (-55%)	+11 (+66%)
0.5	0.125	100	100	2	+5.58 (+153%)	-4842 (-66%)	+14 (+88%)
0.5	0.125	100	100	4	+5.84 (+166%)	-5032 (-67%)	+15 (+101%)
1.0	0.5	100	100	1	+4.16 (+124%)	-4948 (-56%)	+3 (+8%)
1.0	0.5	80	100	2	-2.52 (-38%)	-3592 (-44%)	-1 (-2%)
1.0	0.5	90	100	2	+2.28 (+53%)	-4290 (-50%)	+1 (+2%)
1.0	0.5	100	100	2	+4.08 (+120%)	-4618 (-53%)	+1 (+3%)
1.0	0.5	80	100	4	-1.56 (-25%)	-3825 (-46%)	-0 (-0%)
1.0	0.5	90	100	4	+2.34 (+55%)	-4389 (-51%)	+1 (+4%)
1.0	0.25	100	100	2	+5.00 (+170%)	-12195 (-98%)	+23 (+94%)
1.0	0.25	100	100	4	+5.28 (+189%)	-12295 (-98%)	+24 (+96%)

Table 64: Adaptive: Results for networks of size 11 and configuration FurtherSinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	-7.22 (-137%)	-3815 (-34%)	-2 (-59%)
0.125	0.0625	90	100	2	-2.22 (-80%)	-4819 (-41%)	-1 (-24%)
0.125	0.0625	100	100	2	-0.20 (-11%)	-5490 (-45%)	+0 (+1%)
0.125	0.0625	80	100	4	-6.16 (-130%)	-4071 (-36%)	-2 (-49%)
0.125	0.0625	90	100	4	-1.68 (-67%)	-5031 (-42%)	-0 (-17%)
0.125	0.0625	100	100	4	+0.52 (+37%)	-5631 (-46%)	+0 (+8%)
0.25	0.125	80	100	2	-5.06 (-85%)	-3800 (-42%)	-2 (-17%)
0.25	0.125	90	100	2	-0.14 (-4%)	-4521 (-48%)	+0 (+4%)
0.25	0.125	100	100	2	+1.48 (+56%)	-4933 (-52%)	+1 (+16%)
0.25	0.125	80	100	4	-4.08 (-75%)	-4036 (-44%)	-1 (-7%)
0.25	0.125	90	100	4	+0.54 (+17%)	-4728 (-50%)	+1 (+14%)
0.25	0.125	100	100	4	+2.20 (+96%)	-5262 (-54%)	+2 (+29%)
0.25	0.0625	100	100	2	+2.70 (+132%)	-6148 (-60%)	+5 (+69%)
0.25	0.0625	100	100	4	+2.88 (+147%)	-6302 (-62%)	+5 (+79%)
0.5	0.25	80	100	2	-4.00 (-90%)	-6747 (-35%)	+1 (+6%)
0.5	0.25	90	100	2	+0.50 (+23%)	-7878 (-40%)	+3 (+15%)
0.5	0.25	100	100	2	+2.14 (+154%)	-8383 (-42%)	+3 (+19%)
0.5	0.25	80	100	4	-3.22 (-79%)	-7030 (-36%)	+2 (+9%)
0.5	0.25	90	100	4	+0.78 (+38%)	-8093 (-41%)	+3 (+18%)
0.5	0.25	100	100	4	+2.26 (+170%)	-8777 (-44%)	+4 (+26%)
1.0	0.5	80	100	2	-3.38 (-70%)	-7949 (-30%)	-1 (-2%)
1.0	0.5	90	100	2	+0.90 (+33%)	-9663 (-35%)	+0 (+1%)
1.0	0.5	80	100	4	-3.00 (-64%)	-8132 (-31%)	-1 (-1%)
1.0	0.5	90	100	4	+1.66 (+71%)	-9879 (-36%)	+1 (+2%)
1.0	0.5	100	100	4	+2.96 (+176%)	-11029 (-39%)	+1 (+4%)
1.0	0.25	100	100	4	+3.06 (+188%)	-32434 (-84%)	+17 (+64%)

Table 65: Adaptive: Results for networks of size 15 and configuration FurtherSinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	90	100	2	-1.74 (-112%)	-17715 (-47%)	-0 (-20%)
0.125	0.0625	80	100	4	-6.20 (-164%)	-14524 (-41%)	-2 (-58%)
0.125	0.0625	90	100	4	-1.78 (-113%)	-17503 (-47%)	-0 (-20%)
0.125	0.0625	100	100	4	+0.30 (+57%)	-19261 (-50%)	+0 (+8%)
0.25	0.125	80	100	1	-5.02 (-135%)	-13576 (-49%)	-1 (-16%)
0.25	0.125	90	100	2	-0.98 (-57%)	-14821 (-53%)	+0 (+3%)
0.25	0.125	80	100	4	-5.12 (-135%)	-12724 (-47%)	-1 (-17%)
0.25	0.125	90	100	4	-0.94 (-56%)	-14847 (-53%)	+1 (+9%)
0.25	0.125	100	100	4	+0.94 (+125%)	-15916 (-56%)	+2 (+22%)

0.5	0.25	80	100	2	-5.88 (-156%)	-16827 (-29%)	-3 (-15%)
0.5	0.25	90	100	2	-0.96 (-73%)	-20929 (-35%)	-1 (-4%)
0.5	0.25	100	100	2	+0.68 (+136%)	-23291 (-38%)	+1 (+4%)
0.5	0.25	80	100	4	-4.76 (-148%)	-17822 (-31%)	-2 (-11%)
0.5	0.25	90	100	4	-0.68 (-58%)	-21458 (-36%)	-0 (-1%)
0.5	0.125	100	100	4	+0.72 (+150%)	-35193 (-53%)	+9 (+75%)
1.0	0.5	80	100	2	-4.98 (-129%)	-18834 (-23%)	-4 (-12%)
1.0	0.5	90	100	2	-0.66 (-39%)	-24273 (-29%)	-3 (-8%)
1.0	0.5	100	100	2	+1.32 (+183%)	-27478 (-32%)	-2 (-6%)
1.0	0.5	80	100	4	-4.08 (-119%)	-19683 (-24%)	-4 (-11%)
1.0	0.5	90	100	4	-0.20 (-14%)	-24632 (-29%)	-3 (-8%)

Table 66: Adaptive: Results for networks of size 21 and configuration FurtherSinkCorner

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	100	100	4	+0.12 (+18%)	-34993 (-53%)	+0 (+2%)
0.125	0.0625	80	100	2	-6.40 (-163%)	-28433 (-45%)	-2 (-66%)
0.125	0.0625	100	100	2	+0.16 (+25%)	-36668 (-54%)	+0 (+6%)
0.125	0.0625	80	100	4	-7.54 (-168%)	-26400 (-42%)	-2 (-73%)
0.125	0.0625	90	100	4	-1.94 (-115%)	-32414 (-50%)	-1 (-27%)
0.125	0.0625	90	100	2	-1.76 (-110%)	-33873 (-51%)	-0 (-23%)
0.25	0.125	80	100	1	-5.46 (-160%)	-25400 (-53%)	-2 (-21%)
0.25	0.125	100	100	2	+0.42 (+89%)	-30058 (-59%)	+1 (+14%)
0.25	0.125	80	100	4	-5.56 (-161%)	-23898 (-50%)	-2 (-24%)
0.25	0.125	90	100	4	-1.28 (-97%)	-27525 (-56%)	+0 (+2%)
0.25	0.125	100	100	4	+0.38 (+78%)	-29668 (-59%)	+1 (+16%)
0.5	0.25	90	100	1	-1.52 (-133%)	-42525 (-39%)	+1 (+6%)
0.5	0.25	80	100	2	-6.54 (-179%)	-33214 (-32%)	-2 (-11%)
0.5	0.25	90	100	2	-1.54 (-134%)	-40810 (-38%)	+0 (+3%)
0.5	0.25	100	100	2	+0.22 (+81%)	-45030 (-41%)	+2 (+10%)
0.5	0.25	80	100	4	-5.70 (-176%)	-34230 (-33%)	-2 (-9%)
0.5	0.25	100	100	4	+0.32 (+145%)	-45109 (-41%)	+2 (+11%)
1.0	0.5	100	100	1	+1.00 (+192%)	-64865 (-42%)	+3 (+10%)
1.0	0.5	80	100	2	-5.22 (-144%)	-41040 (-28%)	-1 (-2%)
1.0	0.5	100	100	2	+0.98 (+185%)	-58321 (-38%)	+2 (+6%)
1.0	0.5	80	100	4	-4.44 (-137%)	-41732 (-29%)	-1 (-1%)
1.0	0.5	90	100	4	-0.80 (-56%)	-50832 (-34%)	+1 (+2%)
1.0	0.5	100	100	4	+0.92 (+164%)	-57352 (-38%)	+2 (+5%)

Table 67: Adaptive: Results for networks of size 25 and configuration FurtherSinkCorner

H.4 Generic1

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	+3.30 (+58%)	-431 (-18%)	+3 (+51%)
0.125	0.0625	90	100	4	+4.18 (+79%)	-564 (-23%)	+4 (+74%)
0.125	0.0625	100	100	4	+5.02 (+103%)	-645 (-25%)	+5 (+85%)
0.25	0.125	100	100	2	+1.88 (+132%)	-133 (-9%)	+3 (+28%)
0.25	0.125	80	100	4	+1.62 (+105%)	+5 (+0%)	+2 (+23%)
0.25	0.125	90	100	4	+1.80 (+123%)	-35 (-3%)	+3 (+30%)
0.25	0.125	100	100	4	+1.82 (+126%)	-60 (-4%)	+4 (+38%)
0.25	0.0625	90	100	4	+2.16 (+169%)	-237 (-16%)	+5 (+56%)
0.25	0.0625	100	100	4	+2.20 (+175%)	-260 (-18%)	+5 (+59%)
0.25	0.0625	80	100	4	+1.94 (+140%)	-193 (-13%)	+5 (+50%)
0.5	0.25	80	100	4	+1.34 (+90%)	-265 (-12%)	+5 (+27%)
0.5	0.25	90	100	4	+1.52 (+109%)	-333 (-15%)	+6 (+33%)
0.5	0.25	100	100	4	+1.68 (+127%)	-383 (-17%)	+7 (+37%)

0.5	0.125	100	100	4	+1.74 (+135%)	-742 (-30%)	+12 (+73%)
1.0	0.5	80	100	1	-0.90 (-28%)	-1251 (-37%)	+6 (+12%)
1.0	0.5	80	100	4	+1.52 (+75%)	-1360 (-40%)	+14 (+34%)
1.0	0.5	90	100	4	+1.64 (+83%)	-1620 (-46%)	+17 (+41%)
1.0	0.5	100	100	4	+2.02 (+113%)	-1804 (-50%)	+19 (+47%)
1.0	0.25	90	100	4	+2.18 (+127%)	-2926 (-70%)	+33 (+97%)
1.0	0.25	100	100	4	+2.32 (+141%)	-3052 (-72%)	+34 (+101%)
1.0	0.125	90	100	4	+2.36 (+146%)	-3595 (-79%)	+38 (+122%)
1.0	0.125	100	100	4	+2.40 (+150%)	-3735 (-81%)	+39 (+127%)

Table 68: Adaptive: Results for networks of size 11 and configuration Generic1

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	-3.08 (-42%)	-3182 (-33%)	+4 (+44%)
0.125	0.0625	80	100	4	+2.24 (+48%)	-3203 (-34%)	+8 (+105%)
0.125	0.0625	90	100	4	+3.92 (+103%)	-3897 (-39%)	+10 (+133%)
0.125	0.0625	100	100	4	+4.94 (+150%)	-4433 (-44%)	+10 (+151%)
0.25	0.125	80	100	4	+0.44 (+46%)	-113 (-2%)	+2 (+20%)
0.25	0.125	90	100	4	+0.96 (+137%)	-286 (-6%)	+3 (+39%)
0.25	0.125	100	100	4	+1.10 (+175%)	-398 (-8%)	+4 (+50%)
0.25	0.0625	100	100	4	+1.16 (+193%)	-852 (-17%)	+5 (+74%)
0.5	0.25	80	100	4	+0.72 (+109%)	-2139 (-28%)	+11 (+68%)
0.5	0.25	90	100	4	+0.96 (+178%)	-2419 (-31%)	+13 (+79%)
0.5	0.25	100	100	4	+1.00 (+192%)	-2610 (-34%)	+14 (+86%)
0.5	0.125	90	100	4	+1.02 (+200%)	-3232 (-40%)	+16 (+109%)
1.0	0.5	80	100	1	-1.32 (-60%)	-5494 (-51%)	+12 (+26%)
1.0	0.5	90	100	1	-0.92 (-46%)	-6818 (-60%)	+17 (+37%)
1.0	0.5	80	100	4	+1.10 (+111%)	-8060 (-67%)	+30 (+77%)
1.0	0.5	90	100	4	+1.50 (+190%)	-9155 (-73%)	+34 (+91%)
1.0	0.125	100	100	4	+1.54 (+200%)	-14658 (-95%)	+49 (+159%)

Table 69: Adaptive: Results for networks of size 15 and configuration Generic1

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	+0.90 (+23%)	-11708 (-38%)	+13 (+132%)
0.125	0.0625	90	100	4	+3.12 (+111%)	-14586 (-45%)	+14 (+161%)
0.125	0.0625	100	100	4	+4.02 (+171%)	-16445 (-49%)	+15 (+175%)
0.25	0.125	100	100	2	+0.68 (+189%)	-2137 (-13%)	+3 (+63%)
0.25	0.125	80	100	4	+0.08 (+12%)	+267 (+2%)	+0 (+2%)
0.25	0.125	90	100	4	+0.46 (+98%)	-383 (-2%)	+2 (+29%)
0.25	0.125	100	100	4	+0.64 (+168%)	-787 (-5%)	+2 (+46%)
0.5	0.25	80	100	4	+0.00 (+0%)	-5495 (-24%)	+7 (+49%)
0.5	0.25	90	100	4	+0.32 (+123%)	-6486 (-27%)	+9 (+71%)
0.5	0.25	100	100	4	+0.42 (+200%)	-7143 (-30%)	+10 (+83%)
1.0	0.5	80	100	1	-1.82 (-101%)	-18647 (-54%)	+10 (+24%)
1.0	0.5	90	100	1	-1.52 (-92%)	-21173 (-59%)	+13 (+31%)
1.0	0.5	80	100	4	+0.58 (+95%)	-27941 (-71%)	+29 (+88%)
1.0	0.5	90	100	4	+0.82 (+167%)	-31135 (-76%)	+33 (+108%)
1.0	0.25	90	100	4	+0.84 (+175%)	-37934 (-86%)	+40 (+146%)
1.0	0.125	100	100	4	+0.86 (+183%)	-43025 (-92%)	+43 (+167%)

Table 70: Adaptive: Results for networks of size 21 and configuration Generic1

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	+0.76 (+20%)	-21811 (-40%)	+15 (+141%)
0.125	0.0625	90	100	4	+3.04 (+112%)	-27155 (-47%)	+17 (+169%)
0.125	0.0625	100	100	4	+3.98 (+177%)	-30522 (-51%)	+18 (+183%)

0.125	0.0625	80	100	1	-6.36 (-86%)	-21167 (-39%)	+11 (+81%)
0.25	0.0625	100	100	4	+0.46 (+200%)	-2695 (-9%)	+3 (+85%)
0.25	0.125	80	100	4	-0.44 (-65%)	+536 (+2%)	-0 (-2%)
0.25	0.125	90	100	4	+0.30 (+97%)	-609 (-2%)	+2 (+30%)
0.25	0.125	100	100	4	+0.40 (+154%)	-1363 (-5%)	+2 (+50%)
0.25	0.0625	80	100	4	-0.42 (-63%)	-592 (-2%)	+1 (+25%)
0.5	0.25	80	100	4	-0.04 (-8%)	-7416 (-18%)	+5 (+39%)
0.5	0.25	90	100	4	+0.42 (+156%)	-9286 (-22%)	+7 (+65%)
0.5	0.25	100	100	4	+0.48 (+200%)	-10326 (-24%)	+8 (+79%)
1.0	0.5	80	100	1	-2.18 (-142%)	-30762 (-50%)	+10 (+24%)
1.0	0.5	90	100	1	-1.42 (-123%)	-33507 (-54%)	+11 (+27%)
1.0	0.5	100	100	1	-2.00 (-139%)	-33505 (-54%)	+11 (+25%)
1.0	0.5	80	100	4	+0.04 (+10%)	-46844 (-68%)	+30 (+95%)
1.0	0.5	90	100	4	+0.38 (+152%)	-51648 (-72%)	+34 (+114%)
1.0	0.5	100	100	4	+0.44 (+200%)	-54326 (-74%)	+36 (+124%)

Table 71: Adaptive: Results for networks of size 25 and configuration Generic1

H.5 Generic2

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	1	+0.76 (+12%)	-994 (-20%)	-1 (-13%)
0.125	0.0625	80	100	2	+2.82 (+52%)	-1147 (-23%)	-0 (-3%)
0.125	0.0625	100	100	2	+4.14 (+87%)	-1706 (-32%)	+1 (+37%)
0.125	0.0625	80	100	4	+3.96 (+81%)	-1410 (-27%)	+1 (+26%)
0.125	0.0625	90	100	4	+5.26 (+125%)	-1712 (-32%)	+2 (+56%)
0.125	0.0625	100	100	4	+5.94 (+153%)	-1966 (-36%)	+2 (+76%)
0.25	0.125	80	100	1	-0.16 (-3%)	-282 (-7%)	-3 (-26%)
0.25	0.125	90	100	1	+1.60 (+29%)	-490 (-12%)	-1 (-12%)
0.25	0.125	80	100	2	+2.52 (+51%)	-499 (-12%)	-1 (-9%)
0.25	0.125	90	100	2	+3.48 (+77%)	-672 (-16%)	+0 (+2%)
0.25	0.125	80	100	4	+4.64 (+118%)	-796 (-18%)	+2 (+30%)
0.25	0.125	90	100	4	+5.10 (+138%)	-959 (-22%)	+4 (+45%)
0.25	0.125	100	100	4	+5.30 (+148%)	-1123 (-25%)	+4 (+60%)
0.25	0.0625	90	100	2	+5.44 (+155%)	-1512 (-32%)	+4 (+56%)
0.25	0.0625	80	100	4	+5.36 (+151%)	-1434 (-31%)	+5 (+66%)
0.25	0.0625	90	100	4	+5.84 (+176%)	-1648 (-34%)	+6 (+85%)
0.5	0.25	90	100	1	+3.36 (+94%)	-602 (-8%)	-1 (-6%)
0.5	0.25	80	100	2	+3.00 (+80%)	-373 (-5%)	-2 (-12%)
0.5	0.25	90	100	2	+3.20 (+87%)	-557 (-7%)	-2 (-10%)
0.5	0.25	80	100	4	+3.70 (+109%)	-877 (-11%)	+1 (+9%)
0.5	0.25	90	100	4	+3.92 (+119%)	-1048 (-13%)	+2 (+11%)
0.5	0.25	100	100	4	+3.94 (+120%)	-1224 (-15%)	+2 (+15%)
0.5	0.125	90	100	1	+4.06 (+126%)	-2138 (-25%)	+6 (+43%)
0.5	0.125	90	100	2	+4.00 (+123%)	-2102 (-25%)	+6 (+39%)
0.5	0.125	100	100	2	+4.26 (+136%)	-2316 (-27%)	+7 (+47%)
0.5	0.125	80	100	4	+4.16 (+131%)	-2211 (-26%)	+7 (+48%)
0.5	0.125	90	100	4	+4.66 (+159%)	-2505 (-29%)	+8 (+60%)
1.0	0.5	90	100	1	+1.70 (+78%)	-370 (-3%)	-3 (-9%)
1.0	0.5	100	100	1	+1.80 (+85%)	-652 (-6%)	-3 (-8%)
1.0	0.5	80	100	2	+0.68 (+25%)	+595 (+6%)	-7 (-17%)
1.0	0.5	90	100	2	+1.26 (+53%)	+268 (+2%)	-6 (-16%)
1.0	0.5	100	100	2	+1.28 (+54%)	+51 (+0%)	-6 (-16%)
1.0	0.5	80	100	4	+1.14 (+47%)	+283 (+3%)	-5 (-14%)
1.0	0.5	90	100	4	+1.50 (+66%)	-67 (-1%)	-5 (-12%)
1.0	0.5	100	100	4	+1.62 (+73%)	-278 (-2%)	-5 (-12%)
1.0	0.25	90	100	2	+2.44 (+136%)	-5958 (-43%)	+14 (+49%)
1.0	0.25	100	100	2	+2.50 (+141%)	-6112 (-43%)	+14 (+48%)
1.0	0.25	80	100	4	+2.12 (+108%)	-5922 (-42%)	+14 (+50%)

1.0	0.25	90	100	4	+2.78 (+171%)	-6191 (-44%)	+15 (+52%)

Table 72: Adaptive: Results for networks of size 11 and configuration Generic2

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	-0.24 (-7%)	-3421 (-23%)	-1 (-27%)
0.125	0.0625	90	100	2	+1.22 (+47%)	-4285 (-28%)	-0 (-3%)
0.125	0.0625	80	100	4	+0.62 (+22%)	-4010 (-27%)	-0 (-10%)
0.125	0.0625	90	100	4	+1.70 (+73%)	-4857 (-31%)	+0 (+14%)
0.125	0.0625	100	100	4	+2.52 (+131%)	-5681 (-36%)	+1 (+34%)
0.25	0.125	80	100	2	-0.42 (-14%)	-1433 (-12%)	-4 (-42%)
0.25	0.125	90	100	2	+1.12 (+51%)	-2017 (-17%)	-2 (-27%)
0.25	0.125	80	100	4	+1.22 (+57%)	-2283 (-18%)	-1 (-16%)
0.25	0.125	90	100	4	+1.94 (+108%)	-2886 (-23%)	+0 (+1%)
0.25	0.125	100	100	4	+2.26 (+139%)	-3244 (-25%)	+1 (+11%)
0.25	0.0625	90	100	4	+2.36 (+149%)	-4657 (-34%)	+4 (+60%)
0.25	0.0625	100	100	4	+2.66 (+186%)	-5219 (-38%)	+4 (+77%)
0.5	0.25	80	100	2	+3.46 (+125%)	-1294 (-6%)	-6 (-32%)
0.5	0.25	90	100	2	+4.04 (+163%)	-1738 (-7%)	-5 (-27%)
0.5	0.25	100	100	2	+4.12 (+169%)	-1997 (-9%)	-5 (-26%)
0.5	0.25	90	100	4	+4.32 (+185%)	-2356 (-10%)	-4 (-20%)
0.5	0.25	100	100	4	+4.36 (+188%)	-2708 (-11%)	-3 (-18%)
0.5	0.125	100	100	4	+4.42 (+193%)	-6379 (-25%)	+4 (+34%)
1.0	0.5	80	100	4	+2.66 (+134%)	+2108 (+6%)	-9 (-23%)
1.0	0.5	90	100	4	+3.06 (+171%)	+1410 (+4%)	-8 (-22%)
1.0	0.5	100	100	4	+3.18 (+184%)	+788 (+2%)	-8 (-21%)
1.0	0.25	90	100	4	+3.22 (+188%)	-18569 (-42%)	+11 (+40%)

Table 73: Adaptive: Results for networks of size 15 and configuration Generic2

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	2	-1.50 (-113%)	-15085 (-34%)	-1 (-20%)
0.125	0.0625	80	100	4	-0.98 (-92%)	-15157 (-34%)	-1 (-16%)
0.125	0.0625	90	100	4	-0.16 (-24%)	-17350 (-39%)	+0 (+3%)
0.125	0.0625	100	100	4	+0.26 (+58%)	-19092 (-42%)	+1 (+21%)
0.25	0.125	80	100	2	-0.74 (-80%)	-10236 (-30%)	-2 (-19%)
0.25	0.125	90	100	2	+0.08 (+15%)	-11412 (-33%)	-1 (-8%)
0.25	0.125	80	100	4	-0.46 (-58%)	-10835 (-31%)	-1 (-10%)
0.25	0.125	90	100	4	+0.24 (+55%)	-12059 (-34%)	+0 (+2%)
0.25	0.125	100	100	4	+0.46 (+139%)	-12956 (-36%)	+1 (+9%)
0.25	0.0625	100	100	4	+0.50 (+161%)	-18621 (-49%)	+5 (+82%)
0.5	0.25	80	100	4	+0.04 (+6%)	-9563 (-13%)	-4 (-22%)
0.5	0.25	90	100	4	+0.56 (+156%)	-10829 (-15%)	-3 (-18%)
0.5	0.25	100	100	4	+0.62 (+188%)	-11644 (-16%)	-2 (-14%)
0.5	0.125	100	100	4	+0.64 (+200%)	-21418 (-28%)	+5 (+41%)
1.0	0.5	100	100	2	+2.14 (+166%)	-2600 (-2%)	-7 (-20%)
1.0	0.5	80	100	4	+1.74 (+117%)	+5688 (+5%)	-11 (-28%)
1.0	0.5	90	100	4	+2.12 (+163%)	+3585 (+3%)	-10 (-26%)
1.0	0.25	100	100	1	+2.20 (+175%)	-72023 (-50%)	+17 (+68%)

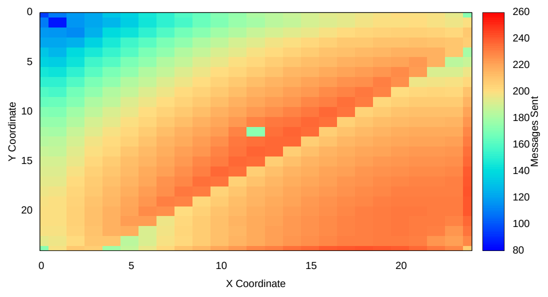
Table 74: Adaptive: Results for networks of size 21 and configuration Generic2

P_{src} (secs)	P_{fs} (secs)	TFS (%)	PFS (%)	D (secs)	Captured (%)	Fake Messages	Received (%)
0.125	0.0625	80	100	4	-1.00 (-109%)	-31047 (-41%)	-0 (-15%)
0.125	0.0625	90	100	4	-0.10 (-21%)	-33945 (-44%)	+0 (+2%)
0.125	0.0625	100	100	4	+0.26 (+90%)	-36674 (-47%)	+0 (+16%)

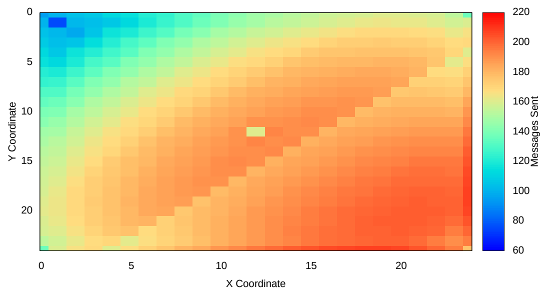
0.25	0.125	80	100	2	-0.84 (-124%)	-23235 (-40%)	-1 (-14%)
0.25	0.125	80	100	4	-0.90 (-127%)	-23161 (-40%)	-1 (-13%)
0.25	0.125	90	100	4	+0.08 (+36%)	-25194 (-43%)	-0 (-0%)
0.25	0.125	100	100	4	+0.22 (+147%)	-26870 (-45%)	+1 (+8%)
0.5	0.25	90	100	4	-0.02 (-18%)	-26172 (-21%)	-2 (-12%)
0.5	0.25	100	100	4	+0.06 (+86%)	-27730 (-22%)	-1 (-9%)
0.5	0.25	80	100	4	-0.56 (-147%)	-23902 (-19%)	-3 (-17%)
1.0	0.5	100	100	1	+1.50 (+165%)	-35404 (-17%)	+1 (+4%)
1.0	0.5	100	100	2	+1.42 (+149%)	-14565 (-8%)	-4 (-10%)
1.0	0.5	80	100	4	+0.90 (+74%)	+554 (+0%)	-7 (-20%)
1.0	0.5	90	100	4	+1.38 (+142%)	-3169 (-2%)	-6 (-17%)
1.0	0.125	100	100	1	+1.56 (+177%)	-166548 (-62%)	+27 (+129%)

Table 75: Adaptive: Results for networks of size 25 and configuration Generic2

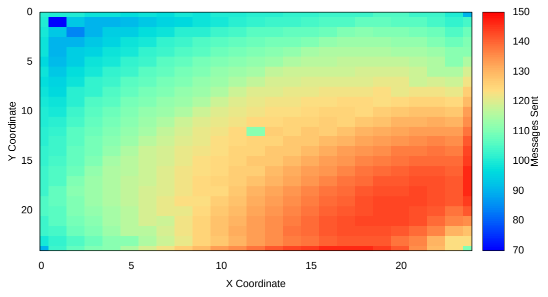
I Adaptive Energy Tables



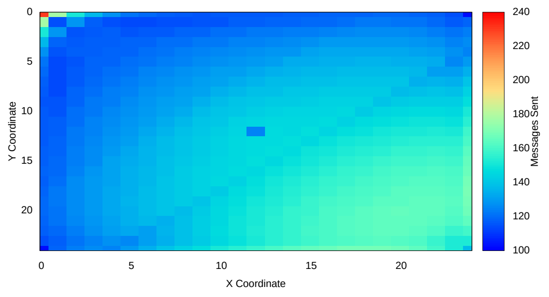
(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.5 seconds

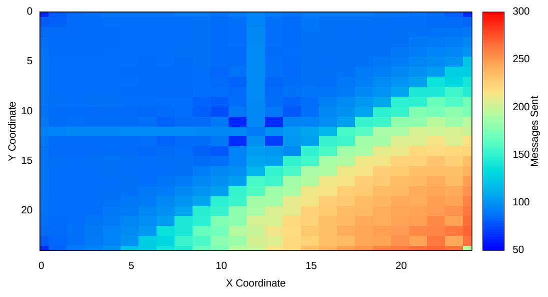


(c) Size 25, Source Period 0.25 seconds

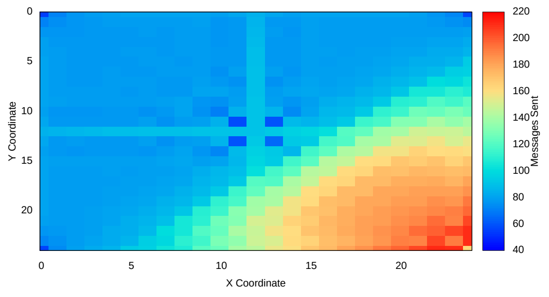


(d) Size 25, Source Period 0.125 seconds

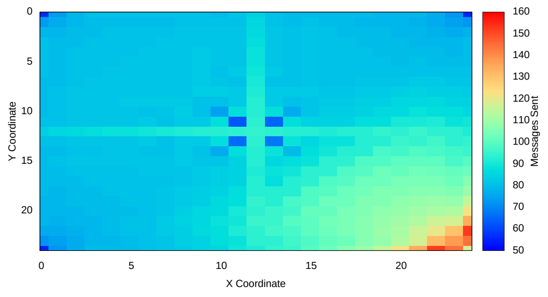
Figure 89: Messages sent while running the Adaptive algorithm for the SourceCorner configuration



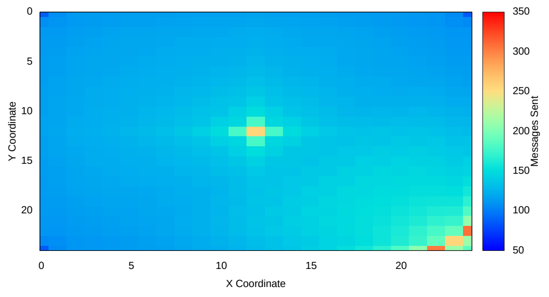
(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.5 seconds

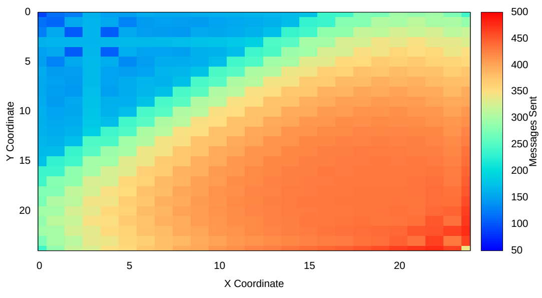


(c) Size 25, Source Period 0.25 seconds

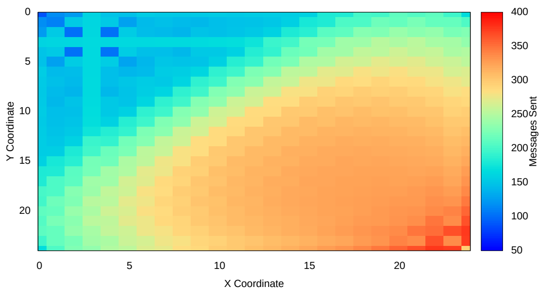


(d) Size 25, Source Period 0.125 seconds

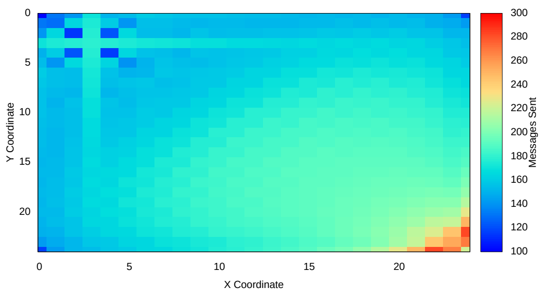
Figure 90: Messages sent while running the Adaptive algorithm for the SinkCorner configuration



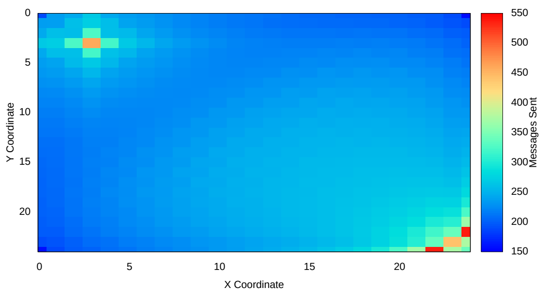
(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.5 seconds

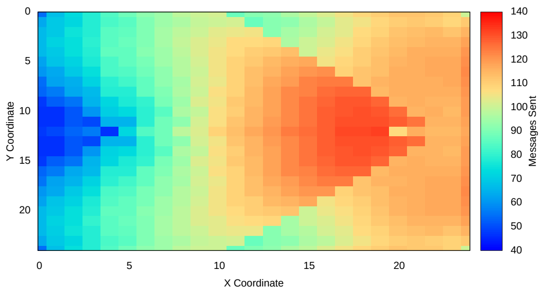


(c) Size 25, Source Period 0.25 seconds

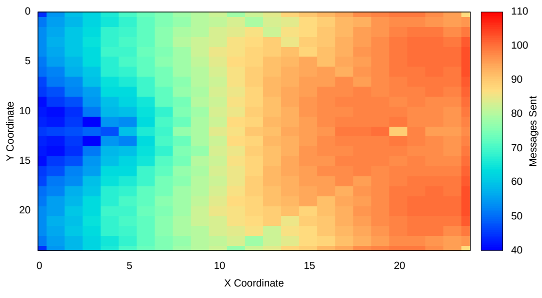


(d) Size 25, Source Period 0.125 seconds

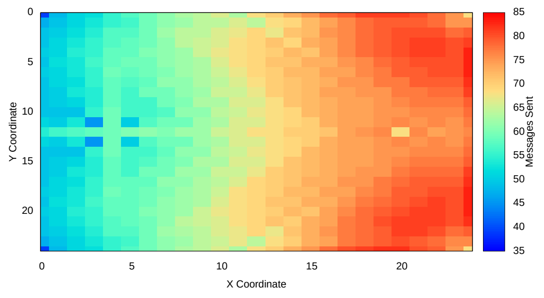
Figure 91: Messages sent while running the Adaptive algorithm for the FurtherSinkCorner configuration



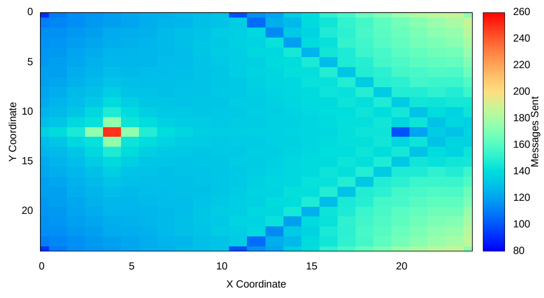
(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.5 seconds

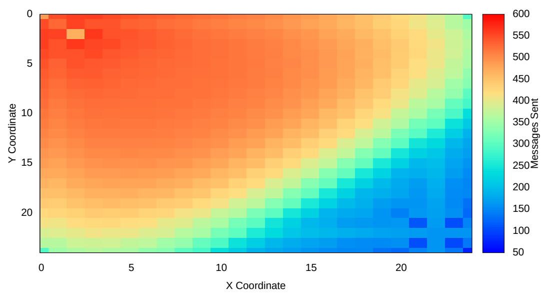


(c) Size 25, Source Period 0.25 seconds

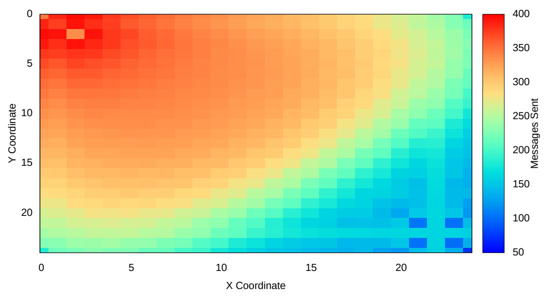


(d) Size 25, Source Period 0.125 seconds

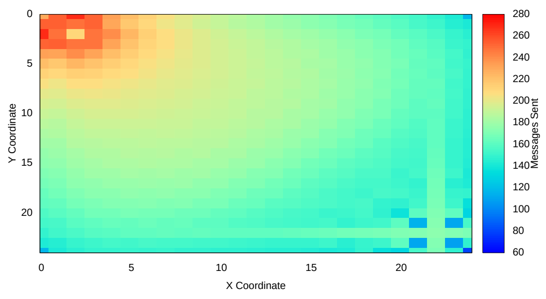
Figure 92: Messages sent while running the Adaptive algorithm for the Generic1 configuration



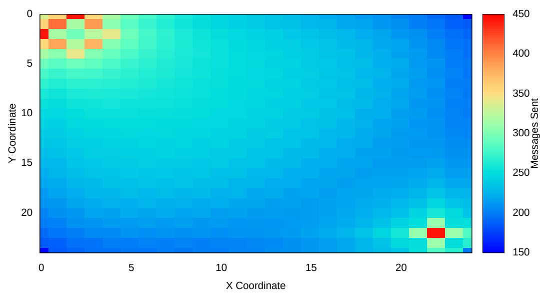
(a) Size 25, Source Period 1 second



(b) Size 25, Source Period 0.5 seconds



(c) Size 25, Source Period 0.25 seconds



(d) Size 25, Source Period 0.125 seconds

Figure 93: Messages sent while running the Adaptive algorithm for the Generic2 configuration

J References

[1] BOINC, November 2011. URL `https://boinc.berkeley.edu/`.

[2] JProwler, 2011. URL `http://w3.isis.vanderbilt.edu/projects/nest/jprowler`.

[3] 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: 10.1016/j.adhoc.2003.09.010. URL `http://www.sciencedirect.com/science/article/pii/S1570870503000738`.

[4] Alicia Asn and Manuel Calahorra. Sensor networks to monitor air pollution in cities, September 2010. URL `http://www.libelium.com/smart_cities_wsn_air_pollution`.

[5] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. *Wireless Sensors Networks Security*, chapter Vulnerabilities and Attacks in Wireless Sensor Networks. IOS Press, 2008. URL `http://pi1.informatik.uni-mannheim.de/index.php?inc=staffhome.php3&user_id=24`.

[6] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. In *ACM SIGCOMM Computer Communication Review*, volume 24, pages 212–225. ACM, 1994.

[7] M. Brownfield, Yatharth Gupta, and N. Davis. Wireless sensor network denial of sleep attack. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 356 –364, june 2005. doi: 10.1109/IAW.2005.1495974.

[8] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5):28 –34, oct 2000. ISSN 1070-9916. doi: 10.1109/98.878533.

[9] Jae-Hwan Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(4):609 – 619, aug. 2004. ISSN 1063-6692. doi: 10.1109/TNET.2004.833122.

[10] Pilsoon Choi, Hyung Chul Park, Sohyeong Kim, Sungchung Park, Ilku Nam, Tae Wook Kim, Seokjong Park, Sangho Shin, Myeung Su Kim, Kyucheol Kang, Yeonwoo Ku, Hyokjae Choi, Sook Min Park, and Kwyro Lee. An experimental coin-sized radio for extremely low-power wpan (ieee 802.15.4) application at 2.4 ghz. *Solid-State Circuits, IEEE Journal of*, 38(12): 2258 – 2268, dec. 2003. ISSN 0018-9200. doi: 10.1109/JSSC.2003.819083.

[11] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 113–126. Ieee, 2005.

[12] J. Fagerström. Design and test of distributed applications. In *Proceedings of the 10th international conference on Software engineering*, ICSE '88, pages 88–92, Los Alamitos, CA, USA, 1988. IEEE Computer Society Press. ISBN 0-89791-258-6. URL `http://dl.acm.org/citation.cfm?id=55823.55833`.

[13] Dieter Gollmann. *Computer security (2. ed.)*. Wiley, 2005. ISBN 978-0-470-86293-3.

[14] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1 –6, oct. 2007. doi: 10.1109/MOBHOC.2007.4428702.

[15] J.L. Hill and D.E. Culler. Mica: a wireless platform for deeply embedded networks. *Micro, IEEE*, 22(6):12 – 24, nov/dec 2002. ISSN 0272-1732. doi: 10.1109/MM.2002.1134340.

[16] Arshad Jhumka, Matthew Leeke, and Sambid Shrestha. On the use of fake sources for source location privacy: Trade-offs between energy and privacy. *The Computer Journal*, 54(6): 860–874, 2011. doi: 10.1093/comjnl/bxr010. URL `http://comjnl.oxfordjournals.org/content/54/6/860.abstract`.

[17] Arshad Jhumka, Matthew Bradbury, and Matthew Leeke. Towards understanding source location privacy in wireless sensor networks through fake sources. In *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*, June 2012. URL `http://eprints.dcs.warwick.ac.uk/1607/`.

[18] P. Kamat, Yanyong Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 599 –608, June 2005. doi: 10.1109/ICDCS.2005.31. URL `http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=1437121`.

[19] Kavi Kumar Khedo, Rajiv Perseedoss, and Avinash Mungur. A wireless sensor network air pollution monitoring system. *International Journal of Wireless & Mobile Networks*, 2(2): 31–45, May 2010. doi: 10.5121/ijwmn.2010.2203.

[20] W. Kluge, F. Poegel, H. Roller, M. Lange, T. Ferchland, L. Dathe, and D. Eggert. A fully integrated 2.4-ghz ieee 802.15.4-compliant transceiver for zigbee trade; applications. *Solid-State Circuits, IEEE Journal of*, 41(12):2767 –2775, dec. 2006. ISSN 0018-9200. doi: 10.1109/JSSC.2006.884802.

[21] S.S. Kulkarni and Limin Wang. Mnp: Multihop network reprogramming service for sensor networks. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 7 –16, june 2005. doi: 10.1109/ICDCS.2005.50.

[22] Harvey Maylor. *Project Management*. Financal Times Management, 4th edition, 2010.

[23] K. Mehta, Donggang Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 314 –323, October 2007. doi: 10.1109/ICNP.2007.4375862. URL `http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=4375862`.

[24] S.A. Munir, Biao Ren, Weiwei Jiao, Bin Wang, Dongliang Xie, and Man Ma. Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 2, pages 113 –120, may 2007. doi: 10.1109/AINAW.2007.257.

[25] Celal Ozturk, Yanyong Zhang, and Wade Trappe. Source-location privacy in energy-constrained sensor network routing. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, SASN '04, pages 88–93, New York, NY, USA, 2004. ACM. ISBN 1-58113-972-1. doi: http://doi.acm.org/10.1145/1029102.1029117. URL `http://doi.acm.org/10.1145/1029102.1029117`.

[26] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18 – 27, jan.-march 2005. ISSN 1536-1268. doi: 10.1109/MPRV.2005.9.

[27] Adrian Perrig, John Stankovic, David Wagner, and Caren Rosenblatt. Security in wireless sensor networks. *Communications of the ACM*, 47:53–57, 2004. URL `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.9059`.

[28] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031508. URL `http://doi.acm.org/10.1145/1031495.1031508`.

[29] S. Roundy, E.S. Leland, J. Baker, E. Carleton, E. Reilly, E. Lai, B. Otis, J.M. Rabaey, P.K. Wright, and V. Sundararajan. Improving power output for vibration-based energy scavengers. *Pervasive Computing, IEEE*, 4(1):28 – 36, jan.-march 2005. ISSN 1536-1268. doi: 10.1109/MPRV.2005.14.

[30] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.

[31] Min Shao, Wenhui Hu, Sencun Zhu, Guohong Cao, Srikanth Krishnamurthy, and Tom La Porta. Cross-layer enhanced source location privacy in sensor networks. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.156.8614`.

[32] Victor Shnayder, Mark Hempstead, Bor rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *In Sensys*, pages 188–200. ACM Press, 2004. URL `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.9516`.

[33] Javier Solobera. Detecting Forest Fires using Wireless Sensor Networks with Waspmote, October 2010. URL `http://www.libelium.com/wireless_sensor_networks_to_detec_forest_fires/`.

[34] Robert Szewczyk, Joseph Polastre, Alan M. Mainwaring, and David E. Culler. Lessons from a sensor network expedition. In *EWSN*, pages 307–322, 2004.

[35] Andrew Tanenbaum. *Computer Networks.* Prentice Hall Professional Technical Reference, 4th edition, 2002. ISBN 0130661023.

[36] The University of Warwick Department of Computer Science. CS402 High Performance Computing, Nov 2010. URL `http://www2.warwick.ac.uk/fac/sci/dcs/teaching/syllabi/cs402`.

[37] The University of Warwick Library. Access to E-resources, Oct 2011. URL `http://www2.warwick.ac.uk/services/library/main/electronicresources/passwords`.

[38] R. Vyas, V. Lakafosis, M. Tentzeris, H. Nishimoto, and Y. Kawahara. A battery-less, wireless mote for scavenging wireless power at uhf (470-570 mhz) frequencies. In *Antennas and Propagation (APSURSI), 2011 IEEE International Symposium on*, pages 1069 –1072, july 2011. doi: 10.1109/APS.2011.5996465.

[39] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association. ISBN 1-931971-47-1. URL `http://dl.acm.org/citation.cfm?id=1298455.1298491`.

[40] Jianbo Yao and Guangjun Wen. Preserving source-location privacy in energy-constrained wireless sensor networks. In *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on*, pages 412 –416, June 2008. doi: 10.1109/ICDCS.Workshops.2008.42. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4577819`.