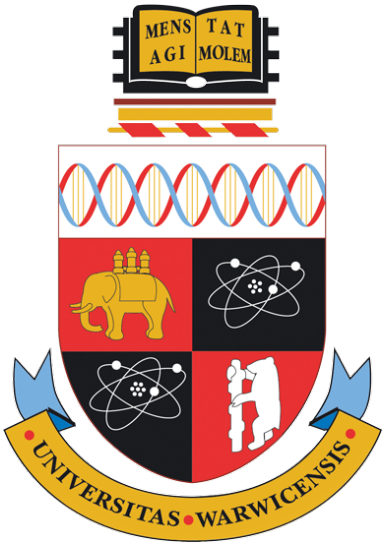
WARWICK
THE UNIVERSITY OF WARWICK



Near Optimal Routing Protocols for Source Location Privacy in Wireless Sensor Networks: Modelling, Design and Evaluation

by

Matthew Bradbury

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

May 2018

Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgments

I would first like to thank my supervisor and mentor, Arshad Jhumka. We have now been working on SLP and WSNs for 8 years and the time has simply flown by. Your advice, guidance, and support has been outstanding and I would not be in the position I am today without your help.

I would also like to thank my fellow PhD students, Jack Kirton and Chen Gu, for our collaborations on SLP. It has been great to work with you during this PhD, getting your different viewpoints and ideas on ways to solve SLP.

For maintaining Flux (the Department of Computer Science's cluster) on which many simulations were run, I would like to thank: Alasdair Thomason, Stephen Roberts, James Marchant, James Dickson, and Phillip Taylor. For our discussions of a statistical nature I want to thank Kevin Ghirardello and Denis Akkavim. I would also like to thank the Centre for Scientific Computing at Warwick for providing HPC facilities and ETH Zurich for providing the FlockLab testbed on which experiments were run.

I would also like to thank the many PhD students and academics that have made my experience what is was. Whether it was doing outreach or teaching together, discussing issues with protecting pandas, or evenings of boardgames, your friendship and support has been greatly appreciated.

Last but not least, for their support during this PhD I also want to thank my family Phil, Paula and James, plus my extended family. I am incredibly grateful for your generosity and help throughout my PhD.

Declarations

1 Publications

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

[24] M. Bradbury, M. Leeke, and A. Jhumka. A dynamic fake source algorithm for source location privacy in wireless sensor networks. In *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 531–538, August 2015. doi: 10.1109/Trustcom.2015.416

[22] M. Bradbury and A. Jhumka. Understanding source location privacy protocols in sensor networks via perturbation of time series. In *INFOCOM, 2017 Proceedings IEEE*, pages 1611–1619, May 2017. doi: 10.1109/INFOCOM.2017.8057122

[21] M. Bradbury and A. Jhumka. A near-optimal source location privacy scheme for wireless sensor networks. In *16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 409–416, August 2017. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.265

[25] M. Bradbury, A. Jhumka, and M. Leeke. Hybrid online protocols for source location privacy in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 115:67–81, May 2018. ISSN 0743-7315. doi: 10.1016/j.jpdc.2018.01.006

Parts of this thesis are currently in submission:

[23] M. Bradbury, A. Jhumka, and C. Maple. A novel source location privacy-aware duty cycle for wireless sensor networks. Manuscript submitted for publication

Research previously performed by the author influenced the contents of this thesis, but does not form part of the thesis:

[76] A. Jhumka, M. Bradbury, and M. 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)*, pages 760–768, June 2012. doi: 10.1109/TrustCom.2012.281

[179] A. Thomason, M. Leeke, M. Bradbury, and A. Jhumka. Evaluating the impact of broadcast rates and collisions on fake source protocols for source location privacy. In *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 667–674, July 2013. doi: 10.1109/TrustCom.2013.81

[77] A. Jhumka, M. Bradbury, and S. Saginbekov. Efficient fault-tolerant collision-free data aggregation scheduling for wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1789–1801, 2014. ISSN 0743-7315. doi: 10.1016/j.jpdc.2013.09.011

[78] A. Jhumka, M. Bradbury, and M. Leeke. Fake source-based source location privacy in wireless sensor networks. *Concurrency and Computation: Practice and Experience*, 27(12):2999–3020, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3242

Research was performed in collaboration during the development of this thesis, but does not form part of the thesis:

[59] C. Gu, M. Bradbury, A. Jhumka, and M. Leeke. Assessing the performance of phantom routing on source location privacy in wireless sensor networks. In *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 99–108, November 2015. doi: 10.1109/PRDC.2015.9

[93] J. F. Laikin, M. Bradbury, C. Gu, and M. Leeke. Towards fake sources for source location privacy in wireless sensor networks with multiple sources. In *15th IEEE International Conference on Communication Systems (ICCS'16)*, pages 1–6, December 2016. doi: 10.1109/ICCS.2016.7833572

[74] A. Jhumka and M. Bradbury. Deconstructing source location privacy-aware routing protocols. In *Proceedings of the Symposium on Applied Computing*, SAC'17, pages 431–436. ACM, April 2017. ISBN 978-1-4503-4486-9. doi: 10.1145/3019612.3019655

[60] C. Gu, M. Bradbury, and A. Jhumka. Phantom walkabouts in wireless sensor networks. In *Proceedings of the Symposium on Applied Computing*, SAC'17, pages 609–616. ACM, April 2017. ISBN 978-1-4503-4486-9. doi: 10.1145/3019612.3019732

[88] J. Kirton, M. Bradbury, and A. Jhumka. Source location privacy-aware data aggregation scheduling for wireless sensor networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2200–2205, June 2017. doi: 10.1109/ICDCS.2017.171

[61] C. Gu, M. Bradbury, J. Kirton, and A. Jhumka. A decision theoretic framework for selecting source location privacy aware routing protocols in wireless sensor networks. *Future Generation Computing Systems*, 87:514–526, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2018.01.046

[89] J. Kirton, M. Bradbury, and A. Jhumka. Towards optimal source location privacy-aware TDMA schedules in wireless sensor networks. *Computer Networks*, 146:125–137, 2018. ISSN 1389-1286. doi: 10.1016/j.comnet.2018.09.010

2 Sponsorships and Grants

This research was supported by the Engineering and Physical Sciences Research Council (EPSRC) [DTG grant EP/M506679/1].

Abstract

Wireless Sensor Networks (WSNs) are collections of small computing devices that are used to monitor valuable assets such as endangered animals. As WSNs communicate wirelessly they leak information to malicious eavesdroppers. When monitoring assets it is important to provide Source Location Privacy (SLP), where the location of the message source must be kept hidden. Many SLP protocols have been developed by designing a protocol using intuition before evaluating its performance. However, this does not provide insight into how to develop optimal approaches. This thesis will present an alternate approach where the SLP problem is modelled using different techniques to give an optimal output. However, as this optimal output is typically for a restricted scenario, algorithms that trade optimality for generality are subsequently designed. Four main contributions are presented. First, an analysis is performed based on entropy and divergence to gain insight into how to reduce the information an attacker gains via the use of competing paths, and ways to compare the information loss of arbitrary routing protocols. Secondly, the SLP problem is modelled using Integer Linear Programming. The model result guides the design of a generic protocol called *ILPRouting* that groups messages together to reduce the moves an attacker makes. Thirdly, a timing analysis of when events occur is used to dynamically determine fake source parameters for the *Dynamic* and *DynamicSPR* algorithms. These fake sources lure the attacker to their location instead of the real source. Finally, the first SLP-aware duty cycle is investigated, and implemented for *DynamicSPR* to make it more energy efficient. These techniques are evaluated through simulations and deployments on WSN testbeds to demonstrate their effectiveness.

Acronyms

AODV Ad-hoc On Demand Distance Vector.

CP Constraint Programming.

CPS Cyber-Physical System.

CTP Collection Tree Protocol.

FSSP Fake Source Selection Problem.

GPS Global Positioning System.

ILP Integer Linear Programming.

IoT Internet of Things.

LQI Link Quality Indicator.

MAC Medium Access Control.

OS Operating System.

PFS Permanent Fake Source.

RFID Radio-Frequency Identification.

RSSI Received Signal Strength Indicator.

SLP Source Location Privacy.

TailFS Tail Fake Source.

TFS Temporary Fake Source.

WSN Wireless Sensor Network.

WWF World Wide Fund for Nature.

Symbols

q	The sink node
s	The source node
$\mathbb{A}$	The attacker
P_{safety}	The safety period (an amount of time). How long the source is to be protected for
P'_{safety}	The discrete version of the continuous safety period that is split into chunks of a certain size
$\mathcal{TT}$	The amount of time it takes the attacker to capture the source
ϕ	The safety factor, a constant that is multiplied by the time taken to obtain the safety period
P_{src}	The source period (an amount of time). The time between the source sending messages
$D_{TFS}(j)$	TFS Duration — the time the TFS j exists for
$P_{TFS}(j)$	TFS Period — the time between the TFS j sending $\langle$fake$\rangle$ messages
$P_{PFS}(j)$	PFS Period — the time between the PFS j sending $\langle$fake$\rangle$ messages
$\#_{\mathcal{F}}(j)$	Number of $\langle$fake$\rangle$ messages to be sent from j during the TFS duration
α	The average time it takes for one message to be sent by a node and received by its neighbour
1HopN(j)	The set of nodes in the 1-hop neighbourhood of j
$\Delta(n, n')$	The minimum number of hops between the nodes n and n'
$\Delta_{sink}(j)$	The distance in hops between node j and the sink
$\Delta_{src}(j)$	The distance in hops between node j and the source
Δ_{ss}	The distance in hops between the sink and the source
$\mathcal{N}_i$	The i^{th} $\langle$normal$\rangle$ message sent by the source
$\mathcal{A}$	The $\langle$away$\rangle$ message sent by the sink

$\mathcal{C}$	The ⟨choose⟩ message sent by fake sources at their end-of-life
$\mathcal{F}_i$	The i^{th} ⟨fake⟩ message sent by fake sources
$\mathcal{NO}$	The ⟨notify⟩ message used to notify the sink that the source has been detected
$S_n(\mathcal{M})$	The time one of the message types is sent by node n
$R_n(\mathcal{M})$	The time one of the message types is received at node n
$\Delta_{as}(\mathcal{N}_i)$	The attacker's source distance after receiving i ⟨normal⟩ messages

Throughout this work specific types of messages will be referenced. To make it clear that they are message types, the name of the type will be enclosed in angled brackets. Such as, ⟨normal⟩, ⟨fake⟩, ⟨away⟩, ⟨choose⟩, and ⟨notify⟩ messages.

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are a recent innovation that are becoming widely used to solve a variety of issues. They are comprised of many small computing devices, called *nodes*. These nodes have a low powered CPU and some sensors that allow them to measure their environment. They have a wireless radio that allows them to communicate this information to other parts of the network. Finally, sensor nodes are battery powered as this allows them to be easily deployed without requiring infrastructure. It has only been due to the miniaturisation of hardware, improvements in energy efficiency of the hardware, and better battery technology that sensor networks have become feasible.

Wireless Sensor Networks are typically used in monitoring applications, where sensor data is taken from the environment and relayed back to a base station through nodes in the network. As the communication range of the nodes does not typically cover the entire network, this means that reporting sensed events needs to travel over multiple wireless links (also known as hops). The scope of these applications range from monitoring animal behaviour and habitats [17, 32, 47, 80, 166, 173], observing the health of human-made structures such as bridges [33] and other structures [14, 31], measuring air pollution [53, 87], detecting forest fires [66, 67], monitoring volcanoes [189], surveillance systems [7] and many other applications [5].

The advantage of wireless sensor networks in these applications is that there is no need for a wired communication network or power infrastructure to be in place. The nodes can be attached or scattered where desired and the network will self-configure and adapt to any changes. This flexibility leads to a greatly simplified deployment process which means a WSN can be setup and running with minimal effort. The ease of deployment is one of the factors which is driving the usage of WSNs.

However, a wider range of security and privacy issues need to be addressed

compared to traditional wired networks. This is because messages are sent over a wireless broadcast medium [39] in which anyone in range with an adequate radio can detect and eavesdrop messages. Security and privacy issues also tend to be harder to address due to the decentralised nature of WSNs, the limited capabilities of sensor nodes themselves, and trade-offs that are made to save energy.

These security threats are not just limited to WSNs, but also to novel systems such as the Internet of Things (IoT) and Cyber-Physical Systems (CPS). An IoT involves many small devices that are connected to each other and other devices via the internet. A CPS tends to be similar to both a WSN and IoT but involves a deep integration between the physical system and the software controlling it, often due to the CPS controlling physical actuators. As both an IoT and CPS can involve wireless communication and sensing, many of the security threats faced by a WSN are also applicable to an IoT or CPS. As there is much overlap between WSNs, IoT and CPS [169], the remainder of this thesis will focus on WSNs with the intention that the techniques and discussion could be applied to an IoT or CPS.

Some important security aspects include: ensuring message confidentiality via encryption [145, 146], key management and exchange [135], intrusion detection, secure message routing and validation of localisation [94]. Due to the constraints of embedded systems, a number of novel attacks can be performed against large distributed wireless systems, for example, preventing nodes from going to sleep to save energy in a denial of sleep attack [26]. Another example is wormhole attacks [193] where an attacker uses a long-range wireless link to shorten the distances between two nodes. As many routing protocols aim to take the shortest route, messages prefer to be sent over this wormhole. This allows the attacker to eavesdrop on many more packets than it otherwise would.

Threats against privacy can be classified along two dimensions, they are either (i) *content-based* threats, or (ii) *context-based* threats. Content-based privacy threats relate to preventing the content of the messages broadcast by sensor nodes being revealed to an eavesdropping attacker. The content of a message can be protected using encryption. On the other hand, context-based privacy threats relate to the context in which messages are broadcast. This information is available to attackers because messages are sent wirelessly and can be eavesdropped by any attacker present in the network with an antenna capable of receiving the messages. Context is a multi-attribute concept that encompasses situational aspects of broadcasted messages, including environmental and temporal information. Examples include temporal privacy [83] where the times at which events occur needs to be kept secret from an attacker, the location of the base station to which messages are being routed (sink location privacy) [41], and the location of the nodes that detect an event (source

location privacy) [138].

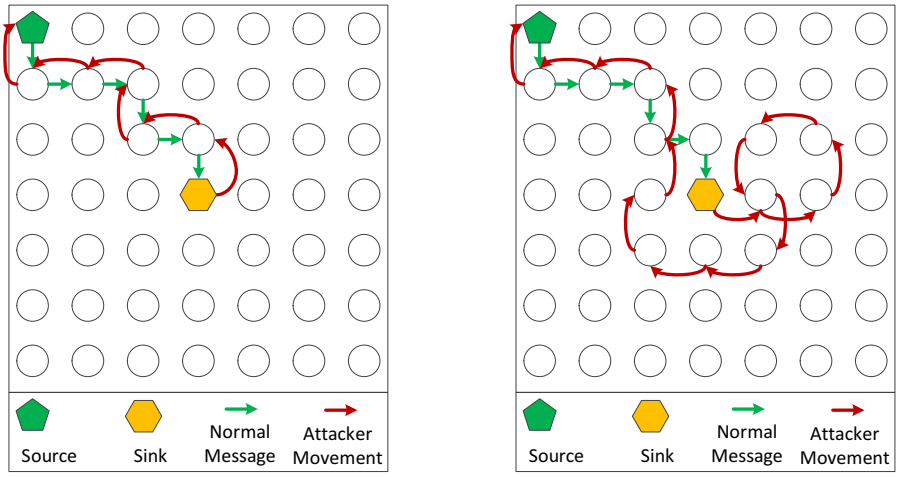
Leaking context information can occur without an eavesdropper needing to read the content of a message. This means that it is insufficient to provide context privacy by protecting the content of messages using encryption. Therefore, other techniques are required to provide context privacy. These techniques must also take into account the restrictions and limitations of the WSN hardware and battery capacity.

1.1 Why Provide Source Location Privacy?

Asset monitoring is one of the applications where WSNs have been used to great effect. There have been a number of theoretical and practical deployments in which WSNs have been used to track assets. In [12] a theoretical technique to monitor and protect tigers in the wild was presented. An actual deployment was undertaken to monitor badger locations in [47] with a WSN deployed to route information and specific nodes in the network designated as badger detection nodes. Another deployment was performed in [176] where movement data was gathered for four sheep. The work aimed to identify certain behaviours based on how the animal's movement changed. Position signals were transmitted every 15 minutes to aid in locating the sheep. A larger deployment was undertaken by the WWF as part of the Wildlife Crime Technology Report [191], where wireless mesh networks, sensors attached to animals and UAVs were deployed to monitor and protect wildlife from poachers. Wireless messages were encrypted [73] to ensure that the *content* of the messages was protected. The Internet of Life[1] project is another real world deployment that monitors animal location in real-time. Two smart parks have been set up in Akagera National Park in Rwanda and Mkomazi National Park in Tanzania. Sensors are embedded in Rhino horns to enable tracking. The smart parks also facilitate other management aspects of conservation alongside tracking. However, none of these examples aimed to protect against *context* attacks, meaning that they could be abused by an attacker to find the location of the animals being monitored or the times at which animals were detected.

Of the different aspects of context that needs to be protected, the source's location is the most important. This is because these networks were originally deployed with the aim of protecting valuable assets that sensor nodes will detect and report information about. If the networks help an attacker find the source they were deployed to help protect, then they have failed in their purpose. In many of these situations the base station is a well known location from which the conservationists

[1] `https://www.theinternetoflife.com/application.html`



(a) Attacker movement in response to messages sent by the source.

(b) The attacker being misled by some SLP scheme, delaying it on its way to the source.

Figure 1.1: Example attacker movement under protectionless and SLP routing

manage the asset monitoring network. Therefore, there is no need to provide sink location privacy, as that location is already well known.

If an attacker is equipped with a directional antenna it can hone in on the source of a signal by moving towards the location where the signal strength increases. It is not expected that the attacker will find the source immediately when it receives a message, but to follow the signal over multiple messages. After receiving a message, the attacker will move in the direction from which the message originated. Eventually it will eavesdrop another message which it will use to refine the direction it is honing in on. Once the node that is the source of the message is found, the valuable asset will likely be nearby, thus allowing an attacker to capture the source using the context of broadcasts in the network [138].

This problem was originally stated in terms of the panda-hunter game [81], where conservationists deployed a WSN across a panda habitat to monitor where the pandas like to gather, in order to better preserve that part of the habitat. When a sensor node detects a panda it floods the network informing all nodes of information about the panda. A poacher who wants to capture the panda uses this information to capture the panda faster than performing an exhaustive search of the habitat.

To provide SLP, novel routing protocols have been designed to reduce the information an attacker can acquire and to increase the time it takes an attacker to capture the source. An example of protectionless and SLP routing is shown in Figure 1.1. The SLP-aware routing protocols typically have messages travel along non-shortest path routes or use fake messages to lure the attacker to a fake

source. The aim of any SLP-aware protocol is to make the attacker using the context information obtained from eavesdropping broadcasts more expensive in terms of time than using traditional tracking techniques. Once an attacker takes longer to find the valuable asset by following messages in an SLP-aware routing protocol its behaviour would revert back to using traditional tracking.

When using the information leaked by the SLP-aware routing protocol becomes the worse option for an attacker, preventing the attacker from capturing the source using traditional techniques remains an issue. A person component is necessary to protect the assets by tracking and arresting the poacher, as was used in the WWF and Internet of Life deployments. Behaviour modelling of attackers [84] can be used to help predict where the attacker will be to enable a better allocation of personnel to patrol those areas. However, protecting the source in this manner is out of scope for SLP-aware routing protocols. The aim of the SLP-aware routing protocols is to make using the broadcast context information too expensive in terms of time for an attacker to use compared to traditional tracking techniques.

1.2 Motivation

Many solutions have been proposed to reduce the in-network attacker's ability to use the context information from message broadcasts to capture the source [40]. The work thus far has approached the development of techniques by first creating the technique and then exploring its efficiency through either simulations or theoretical analysis. While this approach may lead to techniques with good performance, it does not indicate if there are other better techniques that could be used instead or provide a generic model to theoretically compare different techniques. Theoretical analysis that is specific to an individual protocol also has limited usefulness as it is restricted to the technique developed rather than applicable to a class of SLP techniques. To solve this problem this thesis proposes techniques that *first* model the SLP problem and then use that model to produce an optimal way of providing SLP. These techniques should aim for optimality in the sense that the attacker fails to capture the source using the information leaked by the routing protocol. However, as many optimal techniques require global knowledge or only work under limited circumstances, they are infeasible to use in practice. Therefore, intuition is still needed to take the optimal output and devise a technique that trades optimality for generality. The performance of these techniques then needs to be evaluated through experiments. By using different techniques to model and analyse the SLP problem first, insights can be gained into how to structure message routing in novel ways.

The second problem is that parameters for techniques are often specified at

compile time. These statically specified parameters are investigated before deployment, so values that perform well (in terms of SLP provision, or other metrics the network maintainer is interested in) are used in the actual deployment. However, this means that if network circumstances change then these statically specified parameters may no longer perform well (e.g., have a high capture ratio, high number of messages sent, or other poor performance) for the new situation. Techniques should instead seek to have parameters determined on-line using available knowledge. The way to determine how to calculate these parameters should be derived from the modelling of SLP-aware routing. These parameters should encompass configuring how the routing is performed, but should also include additional considerations that are vital to how the protocol will operate. For example, as energy is a constrained resource nodes will need to spend as much time as possible asleep, therefore dynamic parameter calculation should also extend into how to determine when a node should turn on and off in what is known as duty cycling.

1.3 Contributions

The contributions made in this thesis present **models from which optimal source location privacy aware routing protocols can be obtained** and **use these models to design deployable near optimal routing protocols that dynamically determine parameters online**. The focus of optimality in this thesis is in terms of the level of SLP provided, so the aim is to minimise the chance an asset will be captured by the attacker. In support of this thesis the following contributions are made:

- In Chapter 4, the structure of SLP-aware routing protocols is analysed by the entropy (uncertainty) of message routing and a divergence measure between routing protocols. By aiming to minimise the information gain of an attacker, its uncertainty of the source's location can be increased. Techniques are presented to measure the amount of information an attacker loses when comparing two routing protocols, and a heuristic is developed that perturbs a protectionless routing protocol into an SLP-aware routing protocol.

- As a key component of SLP-aware routing is the time at which events occur, a modelling of when broadcasts occur in terms of an integer linear programming (ILP) optimisation problem is performed in Chapter 5. Using an ILP solver, the optimal broadcast schedule is obtained for a specific network topology. This optimal schedule is then used to develop a near optimal routing protocol for delay-tolerant networks that works on a wider range of topologies.

- Furthering the importance of time, a timing analysis of when events occur in the network is undertaken in Chapter 6, and is used to *dynamically determine* the parameters to a fake source based SLP protocol. These fake sources generate $\langle \text{fake} \rangle$ messages to lure the attacker to an area of the network that is away from the source. Two different approaches to the allocation of fake sources are presented, with the latter aiming to reduce the energy usage of the first.

- As energy is also an important consideration when developing SLP-aware routing, in Chapter 7 a duty cycle will be developed for a solution presented in Chapter 6. By extending the timing analysis nodes will be able to calculate when to turn on and off in order to go into a low power sleep mode. By sleeping instead of staying awake, the energy cost of the technique is reduced.

1.4 Organisation

This chapter has introduced the problem of Source Location Privacy in Wireless Sensor Networks and stated the contributions this thesis makes. The remainder of this work is organised as follows:

- Chapter 2 presents a background of the previous work performed in developing techniques for Source Location Privacy and other context privacy solutions for Wireless Sensor Networks.

- Chapter 3 introduces the models used in this work and explains how the models are used in practice when the algorithms are simulated and run on WSN testbeds.

- Chapters 4 to 7 present the main contributions of this work.

- Chapter 8 discusses the implications of this work and also includes a comparison between the techniques.

- Chapter 9 concludes and presents future work in this area.

Chapter 2

Background

Wireless Sensor Networks are comprised of small computers called nodes. These nodes are often limited in their capabilities and resources. As these nodes do not require existing infrastructure to run (such as power cables) they have been used to solve a wide variety of problems. In this chapter the WSN platform will be described, including the limited nature of hardware and the existing techniques used to perform data gathering and reporting in Section 2.1. Security issues that are faced in WSN deployments are explained in Section 2.2. The seminal work on Source Location Privacy will be introduced in Section 2.3 and the remaining sections will cover the work that has been performed thus far on providing context privacy in WSNs.

2.1 Wireless Sensor Networks

The sensor nodes that make up WSNs consist of: a low powered CPU, a small amount of RAM, a small amount of ROM (to store the executable code), a wireless radio (to communicate), a battery (to provide power so the hardware is independent from infrastructure), and sensors relevant to the application being deployed. These sensors are typically temperature, light, and humidity, but other options are available, such as range sensors, motion sensors, RFID, and GPS (to detect geographical location).

A wide variety of hardware platforms are available. The TelosB mote [126] is one such platform that is equipped with a 8 MHz 16-bit CPU (Texas Instruments MSP430), 10 kB of RAM, 1 MB of external flash, and an IEEE 802.15.4 [1] compatible radio that transmits on the 2.4 GHz band. The MICAz mote [124] uses an 8 MHz 8-bit CPU (ATmega128/L), but otherwise has similar capabilities. Both devices are expected to run off two AA batteries for long periods of time. The typical current consumption for TelosB, MICAz and other IEEE 802.15.4 nodes is shown in Table 2.1. For comparison, typical current consumption of IEEE 802.11 (WiFi) and Cellular based IoT nodes are shown in Table 2.2. As the energy cost of leaving the

State	TelosB [126]	MICAz [124]	Z1 [197]	Panstamp NRG3 [139, 178]	LTC 5800-IPM [111]
CPU awake	1800	8000	500 to 10 000	960 to 2880	1300
CPU asleep	5	15	1	3	1
Radio Tx	19 500	17 400	17 400	36 000 to 300 000	9700
Radio Rx	21 800	19 700	18 800	14 000 to 18 000	4500
Radio Idle	21	20	20		
Radio Sleep	1	1	1		

Table 2.1: Typical current consumption (rounded up in µA) for TelosB, MICAz, Z1, Panstamp NRG3, and Linear Technology SmartMesh IP (IEEE 802.15.4 radio)

State	Photon (802.11) [143]	Electron (Cellular) [142]	Waspmote (LoRa EU) [106, 107]
CPU awake	30 000 to 40 000	47 000 to 50 000	17 000
CPU asleep	1000 to 2000	800 to 2000	30
CPU deep sleep	80 to 100	110 to 130	33
Radio Tx	235 000 to 430 000	800 000 to 1 800 000	38 900
Radio Rx	235 000 to 430 000	800 000 to 1 800 000	14 200
Radio Idle	18 000 to 100 000	180 000 to 250 000	2800

Table 2.2: Typical current consumption (µA) for Particle Photon, Particle Electron, and Libelium Waspmote with LoRa

CPU awake is two to three orders of magnitude greater than putting it to sleep, the sensor nodes need to spend as much of their lifetime as possible asleep. Sending and receiving messages can be up to 1000 times more expensive than just leaving the radio idle, which is also 10 times more expensive than putting the radio to sleep. This means that managing the CPU and radio is very important, so protocols should aim to minimise their usage as much as possible in order to maximise network lifetime.

Using WiFi or Cellular based communications tends to be more expensive in terms of energy than IEEE 802.15.4 due to the protocols supporting much higher data rates, so many solutions that need low energy consumption will prefer IEEE 802.14.5 based sensor nodes. However, due to the ease in which WiFi and Cellular communications can take advantage of existing infrastructure and easily access the wider internet, many IoT devices tend to prefer communicating over these kinds of technologies. For areas in which this infrastructure does not exist the preferred communications choices include ad-hoc low power IEEE 802.14.5 communication, or LoRaWAN (Long Range WAN) which provide low power communications over long ranges but with low data transmission rates. However, IoT deployments may choose to take advantage of multiple communication technologies in their heterogeneous deployment, depending on the scenario of interest.

Overall, the hardware platforms that comprise wireless sensor networks are highly constrained and these limitations need to be taken into account when designing applications and protocols. The energy consumption of any protocols or applications deployed on the nodes is one of the most important considerations and it needs to be as low as possible in order to maximise network lifetime.

2.1.1 Sensing

Wireless sensor nodes can be equipped with a variety of sensors that allow them to gather information about the environment that they are situated in. Some very common sensors include: temperature, humidity, visible light and IR light. Many of these are very cheap and can provide interesting information about events occurring near the sensor node. As well as sensing the environment, the sensor node will need to sense information about itself. For example, the voltage levels of a sensor node's batteries will be monitored in order to estimate the amount of energy remaining to the node. A sensor node is also equipped with a number of implicit sensors. For example, the wireless radio can act as a sensor to detect noise on a specific frequency. The radio also records information about received signals in the form of the Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI). WiFi technology has even been used to detect motion, as the sensed signal changes when people move between the sender and detector [62].

Depending on the problem that is being solved with WSNs, specific sensors might also be used. For example, in the Oxford Flood Network deployment[1] sonar sensors are used to measure the height of rivers and streams. If sensor nodes need access to highly accurate time synchronisation or position information, then a GPS unit might be applicable. In industrial scenarios it might be important to monitor the air for the presence of certain gasses such as alcohol, hydrogen, carbon dioxide or others. These sensors could aid in the detection of leaks of these potentially dangerous gasses. Another example is detecting the presence of animals in the wild. These animals might be equipped with a short range radio collar, which a sensor node would need to be able to detect and communicate with. Alternately, sensors could detect vibrations in the ground or air to locate the animals [4].

Overall, the sensor nodes in a WSN may be equipped with a wide range of sensors. These sensors will need to be carefully selected as there are multiple costs to including more sensors. On one hand it is important to consider the monetary cost of the sensor node, because cheaper hardware allows for a larger deployment of sensor nodes. On the other hand there is also an energy cost to power the sensors. Some sensors (such as GPS and gas monitoring [152, Table. 1]) can consume a large

[1] `https://flood.network/news/2015/9/2/our-sensors`

quantity of energy, meaning that including them would lead to a reduction in the lifetime of a sensor network. For example, Libelium Waspmotes can be equipped with a GPS unit that consumes 36 mA [105, pg. 8], if deployments can avoid using GPS then there is more energy available for other activities. Therefore, selecting the sensors that nodes are equipped with is an important consideration to make when deploying a WSN.

2.1.2 Network Stack

Wireless sensor networks are comprised of many sensor nodes that can communicate with their neighbours. The network is typically deployed such that it is not fully connected, i.e., every node can communicate with every other node. Due to this, messages sent from a node to a target *sink* node will often need to travel over multiple wireless links (or *hops*) to reach its destination. A variety of routing protocols have been developed to solve the problem of sending a message from a node to the sink in an energy efficient manner [140]. A selection of routing protocols will be briefly described as the SLP techniques presented in this work provide SLP via novel routing protocols.

One of the simplest routing protocols is flooding, where a message is broadcasted by the source node to all its neighbours and other nodes will rebroadcast messages they receive that have not been previously seen. This is a very energy intensive protocol as all nodes in the network are involved with sending the message. It is also a way for a node to send a message to all nodes in the network. A variant of flooding is known as polite gossiping and is intended to reduce the number of messages sent. In polite gossiping, if a node has a message to send it waits a short interval and if no other node sends the same message in that interval then the node broadcasts the stored message.

Alternate routing protocols aim to send fewer messages than flooding and gossiping. Two of the most widely used protocols are the Collection Tree Protocol (CTP) [57] and Ad hoc On-Demand Distance Vector (AODV) Routing [144]. In CTP the network is arranged into a tree structure rooted at the sink. When a message is generated anywhere in the network it is forwarded down a branch to the sink. The path does not aim to take the fewest hops, but the path that would require the fewest transmissions. So reliable links will be preferred over unreliable links. The designated parent node is periodically checked, if the link is no longer present or a better parent is available then the parent will be recalculated. CTP can also be used to form aggregation trees where data is aggregated (such as averaging temperature data) at nodes where two branches meet. AODV is different to CTP as it supports arbitrary node-to-node communications, whereas in CTP all communications must be to the

sink. Routes are discovered when required and discarded after some time of not being used. Loops are avoided by using sequence numbers to detect if the message has arrived to a previous location on the path. The route discovery algorithm explores outwards from the sending node and once the target is found a message is sent back along the path to inform the sender. Maintenance is performed on the route to detect link failures and alter the route around the failed link. Overall, these two protocols provide a high level of reliability at a relatively low energy cost.

These routing protocols are a small selection of techniques that route messages from one node to another node or multiple nodes. Many other routing protocols exist with different aims. Providing context privacy is one such aim that routing protocols can be used to achieve. When considering location privacy routing protocols need to be able to consider the location, which can either be performed by geographic routing, hop-based routing or routing based on information revealed. Routing protocols that choose child nodes based on the energy cost lack sufficient information which could be used to perturb the route messages take and are therefore unsuitable to provide SLP unless modified to gather this extra information.

2.1.3 Energy Consumption

The power consumption of a WSN application is important to consider due to the limited energy available in a sensor node's battery. In Table 2.1 the energy cost of different activities are shown, with the lowest power consumption being when the node is asleep. So to minimise power consumption a sensor node needs to spend as much time as possible asleep. However, the sensor nodes will need to wake up to perform activities such as sensing, transmitting and receiving messages. There have been many techniques developed to duty cycle wireless sensor nodes [30, 161]. These approaches are typically categorised based upon how nodes synchronise their wakeups and transmissions, either via (i) global time synchronisation, (ii) synchronising time across a subset of nodes (semi-synchronous), or (iii) not attempting to synchronise time (asynchronous).

Approaches such as TDMA provide a straightforward way to assign slots of global time in which nodes are allowed to broadcast. This slot assignment can also be used to determine when a node should sleep. When the current slot is not the node's slot (node might transmit) or any of its neighbour's slots (node might receive), the node can sleep. Existing work has already been performed to allocate a TDMA DAS schedule that provides SLP by assigning slots in a specific way [88]. However, a downside of using TDMA is that there is a high latency between a message being sent and received at the sink. Alternatively, nodes could be clustered, using techniques such as LEACH [68], in order to synchronise time among a cluster of nodes. The

Name	Symbol	Description
Local Wakeup	t_s	How long the radio sleeps for after a wakeup.
Remote Wakeup	t_{tx}	How long a packet's retransmission will be attempted for.
Delay After Receive	t_d 100 ms	Keep the radio on for this period after receiving a message.
Maximum CCA Checks	cca 400	The maximum number of CCA checks that a receiving node performs.
Min Samples Before Detect	$msbd$ 3	The minimum number of detection while performing CCA before a packet is detected.
ACK Wait Delay	t_{ack} 256 jiffies	The amount of time a sender waits for an acknowledgement packet.

Table 2.3: Parameters to TinyOS' DefaultLPL component and their default values

cluster heads then interact with each other asynchronously.

Due to the cost of synchronising time, the majority of duty cycling techniques are asynchronous. TinyOS's Low Power Listening (LPL) [125] performs periodic sampling for a preamble signal, if the preamble is detected then radio remains on to receive the message after the preamble. Alternatively, the receiver can periodically broadcast a beacon to indicate it is willing to receive a message [171]. One of the main downsides to both of these approaches is the high latency introduced. Since the radio only needs to be duty cycled to address the high cost of idle listening, if another radio has a low energy cost to perform idle listening, then it can be used to detect wakeup signals [18]. The problem is that this type of technique requires non-standard hardware that sensor nodes are not usually equipped with. Finally, if a WSN is sufficiently dense, then nodes can randomly wakeup and there will be a high probability that a node will be awake when a message is being sent.

In TinyOS's LPL there are multiple stages to the duty cycling. Initially, when the radio is listening for messages a number of Clear Channel Assessment checks (CCA) are performed. These CCA checks aim to detect a high energy signal being emitted by the radio on the sending node. Multiple checks are needed to ensure that a message is being received and to avoid staying awake when noise is detected. Once a certain number CCAs have been detected then the radio switches to receive mode. Once the radio is done with receiving then it will turn off for a period. When sending a message, the radio will be turned on and packets will be retransmitted for a period specified at compile time. Once an acknowledgement is received the radio has the option to turn off immediately.

One of the important things that needs to be considered when deploying a

duty cycle is what impact will it have on the routing protocol that is being used? The duty cycle may cause links to be unavailable compared to if no duty cycle is being used, which could cause messages to take different routes. This is usually not an issue with routing protocols, as they focus on delivering a message with minimum energy cost. A different route will be very important for SLP, as the space through which messages are routed is an important consideration when trying to protect SLP. Therefore, duty cycle protocols need to take how SLP is provided into account.

2.2 Security in Wireless Sensor Networks

There has been much work performed on providing source anonymity in networks. One technique is to send the data to a third party that then forwards it onto the target after removing the source's identity from metadata known as a Chaum Mix [35]. In this scenario the data is encrypted so that the third party cannot read the data. Onion Routing [58, 153] is an extension of this which sends the data through multiple anonymisers. Each onion router along the path only knows the previous node and discovers the next node by removing a layer of encryption. By performing this multiple times it increases the difficulty of the onion routers from being able to infer the source and destination. However, these techniques do not apply well to the problem of Source Location Privacy in Wireless Sensor Networks because the attacker is capable of eavesdropping the wireless communication at any point in the path. Onion Routing would be suitable against a stationary attacker, but, the attacker model assumed in the SLP problem can involve a mobile attacker or an attacker with a complete view of the network. As the attacker can see multiple parts of the route and can change the section it can see, the attacker is capable of tracing back from the destination to the source using just the direction from which messages arrive.

In [13], Benenson et al. performed a thorough investigation of the attacks that could be performed against a sensor network. The authors categorised an attacker's ability along two dimensions, *presence* and *actions*. It was also noted that the length of time it takes an attacker to perform an attack is an important consideration.

$$\text{local} \rightarrow \text{distributed} \rightarrow \text{global}$$

Figure 2.1: Attacker presence

Presence describes the area of the network that the attacker is capable of influencing. An attacker with *local* presence can only influence a small area of the network that does not change. Such an attacker might be a wireless device fixed at one location. A stronger variant would be an attacker with *distributed* presence.

This attacker would either consist of multiple stationary entities, or mobile entities that can move through the network. The strongest presence would be *global*, where an attacker is capable of seeing or influencing the entire network.

eavesdrop → crash → disturbing → limited passive → passive → reprogramming

Figure 2.2: Attacker actions

Actions describe the ways in which an attacker can interact with the network, where eavesdropping is the weakest and reprogramming is the strongest. An *eavesdropping* attacker is only capable of listening to network traffic. A *crashing* attacker is capable of causing a node to cease operating, this could be through physical attacks that destroy the node or by attacks that cause high energy consumption (such as denial of sleep [26]). A *disturbing* attacker is able to impact the network operation, such as by broadcasting messages or fooling sensors. The *limited passive* attacker can retrieve all information held on a sensor node. This attack may require removing the node from the network for some time. A *passive adversary* can access all information on a sensor node without removing it from the network. It is also capable of modifying data on the node. The most powerful action, *reprogramming*, allows an attacker to run arbitrary code on a sensor node.

While some of these actions can be very strong, the practicality of performing them is related to the length of time it would take to execute the attack. For example, passive attacks such as eavesdropping can be performed continuously, so the time to capture individual messages is low. However, depending on the information an attacker needs to collect, the attacker could spend a long time eavesdropping to gather sufficient information. Depending on how the attack is performed will also impact the time it takes. If the attacker simply needs to plug in a cable, then the attack can be performed quickly. However, if the attacker needs to de-solder components or invoke hardware bugs then the attack would take much longer. When attacks are time sensitive it means certain actions become unsuitable to perform when they take a long time to execute.

2.2.1 Encryption

As WSNs have simple CPUs and limited power, encryption can be a costly expense, however, *content* privacy is something that needs to be provided in certain applications. A number of different approaches with different properties have been developed to provide energy efficient encryption for WSNs.

One of the simplest forms of encryption is to use symmetric encryption with a Pre-Shared Key (PSK). Messages can then be encrypted and decrypted using this

key. However, every node in the network will need to be aware of this key in order to process received messages, or to send new messages. If an attacker compromises one node to obtain the key, then it has compromised the entire network. A way to reduce the impact of node compromises is to use pairwise keys, where every pair of nodes shares a key [113]. This technique encounters a greater energy cost compared to symmetric encryption, as a message will need to be encrypted differently and rebroadcasted for each neighbour a node wants to send a message to. Also, setting up public key techniques tends to be expensive in terms of energy, due the number of messages that need to be sent to obtain the shared secrets and the computation costs. This energy cost needs to be considered when using public key encryption, although there exist some techniques that mitigate this cost [135].

A different alternative to symmetric and asymmetric cryptography is homomorphic encryption, where operations can be performed on encrypted data without decrypting it [146]. This type of cryptography is useful in data aggregation, which is one of the common tasks WSNs perform. Examples of this aggregation function are finding the minimum, maximum or mean. The sensor nodes do not need to know what the currently aggregated value is, and can use homomorphic encryption to simply include its data in the aggregated data.

So encryption is a technique usable for content privacy on WSNs. However, there is an energy cost involved with encrypting plaintext and also setting up the shared secret. Storing the shared secret also makes the memory of a sensor node a useful target to an attacker.

2.3 Seminal Work

Much work has been undertaken investigating SLP in WSNs [40, 97, 155]. This section will first introduce the seminal work [81, 138]. The rest of the chapter that follows will survey much of the existing work performed on location privacy in WSNs.

The seminal work by Ozturk et al. and Kamat et al. first introduced the panda hunter game and proposed a working solution to the SLP problem. The panda-hunter game involves a wireless sensor network being deployed to monitor the location of pandas so conservationists know which part of their habitat to focus on preserving. A node is designated the source when it detects a panda and will broadcast until the panda is no longer detected, the authors assume there is only ever one source. Another node is designated the sink, this is the node messages are being routed to. Finally, there is an attacker present that traces the messages from a random starting location. The attacker is physically present in the area being monitored by the WSN and is equipped with a directional antenna so it is aware of

the direction that messages are received from.

The authors then investigated flooding's ability to provide SLP, however, they found that the attacker would always be able to capture the source from the information leaked by this protocol. Many SLP techniques use *protectionless* flooding as a baseline routing protocol to compare against, as it provides no SLP. It also leaks maximal information to the attacker as no matter its location, the attacker will always eavesdrop a message from the source under flooding (assuming perfectly reliable links).

The second solution Ozturk et al. proposed was that of phantom routing, where the ⟨normal⟩ message being sent to the sink first takes a random walk to a *phantom node* and then floods the network. The aim was to have the phantom node appear to be the source. A purely random walk was found to provide poor SLP, so two different directed random walks were provided: a sector-based approach and a hop-based approach, with the walk being directed to a certain sector of the network or along a hop gradient respectively. This technique was found to provide an increase in the time it took an attacker to capture the source compared to protectionless flooding. An optimisation of the technique was presented [81] which instead of using flooding in the second phase used single-path routing. This alteration to the second stage aimed to decrease the energy cost by reducing the number of messages sent. Many SLP techniques have since used the idea of Phantom Routing to build alternative SLP techniques.

2.4 Routing-based Techniques

An extension to Phantom Routing was introduced by [192], where the authors develop a variation called Greedy Random Walk (GROW). GROW works on a very similar principle to Phantom Routing as they both used random walks to an intermediate node which then floods the network. The difference with GROW is that the next node in the random walk is chosen with a weighted probability biased towards neighbour nodes that have not participated in the random walk. The aim is for the random walk to take very different paths each time a send is initiated. By doing so the attacker should miss receiving messages in the first phase as it is not in the correct location to eavesdrop them.

Phantom Walkabouts [60] is another variant of Phantom Routing where the phantom node is created at the end of a sequence of long and short directed random walks. Short walks are walks that are less than or equal to the sink-source distance and long walks are longer than that distance. The results show that just performing short walks leads to a high capture ratio, and just performing long random walks

leads to a low capture ratio. As an optimisation alternating between long and short random walks helps reduce the number of messages sent by the protocol.

In STaR [108] the set of phantom nodes is constrained to be in a toroidal region around the sink. The source routes to a node within this region, before that message is forwarded to the sink via single-path routing. The toroidal region is divided into a grid, allowing nodes to be assigned coordinates. This allows the source to randomly choose the target phantom node based on calculating the coordinates in the grid from a distance from the sink and an angle. Every message will be sent to a different randomly selected node. The author's compare their technique to the "totally random RRIN scheme [103]", but expect STaR to consume less energy and have a smaller latency due to the restrictions on the size of the toroidal region.

In Phantom Routing, messages often take different routes to different phantom nodes. This type of technique was investigated in [183] where sensor traffic is split to take multiple routing paths. In doing so the attacker is delayed, because when a message is sent on a path that the attacker is not on, the attacker misses eavesdropping that message. The problem is that it is hard to find routes that are far enough away from each other such that the attacker will not be able to eavesdrop on multiple paths. Two techniques are presented, the first called Random Parallel (RP) routing uses global knowledge to assign a number of physically separate parallel routes. This technique does reveal an approximate location of the source as the attacker can wait to receive multiple messages at the sink. As all routes are parallel, they will all lead in the same direction. The second technique called Weighted Random Stride Routing (WRS) uses local information to pick a forwarding angle and distance to travel, which is repeated multiple times. The angle selected is in the direction of the sink. The authors found RP used less energy, but that WRS provided a higher safety period.

The Cyclic Entrapment Method (CEM) was proposed in [137]. CEM works by allocating certain nodes to broadcast a loop of fake messages before a source sends any normal messages. The idea is that an attacker can become trapped in these loops instead of following the message path. The loops get activated when a node in a loop receives a normal message, the authors also investigated probabilistically activating a loop, noting that an energy reduction can be achieved by trading off the privacy provided. Longer loop lengths (after a certain cut-off) were found to provide less privacy as it appears to be harder to generate longer loops. In the paper the authors studied CEM with shortest path routing and noted that the delivery ratio was 100%, this implies that the study was done without considering the effects collisions might have.

A variety of other techniques have been proposed that modify Phantom

Routing to obtain a benefit. For example, in [37] a limited flooding of messages h hops away from the source is performed to help allocate phantom nodes that are widely distributed from one another. Another example is [181] which is based on [19, 85] and aims to create diverse routes with few zigzags or back and forth behaviour, but still delay the attacker on its way to the source. Paths aim to detour around alert zones in which assets are detected. Simulations show that it performs better than Phantom Routing both in terms of having a higher safety period and lower latency. This is also one of a few solutions that is capable of handling multiple sources.

2.4.1 Angle-based Phantom Routing

A common issue with phantom routing is that a poor route from the source to the phantom source can lead to poor SLP performance. One of the popular techniques used to improve the directed random walk is by directing the path based on the angles between source, sink, and intermediate node. The first work that modified the directed random walk to use angle-based routing was [188]. The authors identified that the directed random walk limits the possible nodes it can select. The authors proposed a technique called PRLA to selected better intermediate nodes on the random walk. PRLA uses a directed random walk based on a node's inclination angle with respect to the source and sink. This angle is used to determine the probability to forward messages.

A geographic routing scheme called Angle-based Dynamic Routing Scheme (ADRS) was proposed in [168]. The technique uses angles between nodes to determine the route that a packet should take in a directed walk. The algorithm uses a Request-To-Send/Clear-To-Send handshake to determine the angles needed. The handshake also allows the algorithm to detect nodes leaving and joining the network and adapt the path to those changes. Simulations showed a larger safety period and decreased packet latency compared to standard Phantom Routing.

The work in [119, 120] uses multiple preselected phantom nodes to provide SLP. The phantom nodes are selected by the base station. All the nodes selected to become phantom nodes are outside α degrees either side of the direction the source is in. The set of phantom nodes will be periodically recomputed at the base station and informed of the phantom status. An issue with this approach is that it relies on a centralised allocation of phantom nodes.

An issue with Phantom Routing is that the source node has no control over the location of the phantom node. To solve this the authors of [102, 103, 154] propose the Routing through a Randomly selected Intermediate Node (RRIN) technique, where the source node selects the area of the network the message should be routed

to before sending the message. Once a message reaches one of these phantom nodes it could either be routed to the sink, or a subsequent RRIN could be selected and routed to. Two approaches to selecting multiple RRINs were presented. An angle-based approach and a quadrant-based approach.

The angle-based routing in [79] also aimed to ensure that phantom sources were geographically dispersed across the network. However, it also aimed to utilise energy rich sections of the network. This means that phantom nodes were preferred to be allocated in areas that had more remaining energy compared to areas with less energy in their batteries.

Overall, angle-based techniques for performing the directed random walk phase of phantom routing are capable of better distributing the phantom sources that are allocated. This better allocation leads to an increase in the time it takes an attacker to capture the source, and in certain cases also improves other metrics such as the message latency or the energy cost of the solution.

2.4.2 Ring-based Routing

Similar to Phantom Routing is the use of routing rings. In this class of technique a set of nodes is selected that form a ring structure in the network. When a source detects an event a message is generated and sent to the ring. This message then travels around the ring for a certain number of loops and is finally forwarded to the sink. By travelling around the ring the original direction from which the message originated is obscured from the attacker. This approach was initially proposed in work [100, 101] that extended the concept of routing to a randomly selected intermediate note [102], where the authors used the RRIN selection technique before routing the message to a ring around the source. Once the message reaches the intended area of the RRIN phase, in the Network Mixing Ring (NMR) phase, the message is first routed to a ring node, secondly routed around the ring for a random number of hops, before finally being sent to the source. A downside of this technique is that the path a message needs to travel is long, causing the latency to be high.

Rather then relying on a single ring, [196] expects multiple rings to be present. Sensor nodes on the same rings are all the same distance to the sink in hops. When a message is sent by the source it is first routed to a ring further from the sink before routed to a ring closer to the sink than the source. The direction the message is routed in is randomised (clockwise or counter-clockwise). Eventually the message is routed to the sink along the shortest path. To provide diversity $\langle$fake$\rangle$ messages are injected and forwarded around a ring for a limited number of hops.

Continuing with the idea of multiple rings [131, 132] proposed the Dynamic Optimal mix-ring-based Source Location Anonymity Protocol (DORing protocol).

A ring is formed of nodes that are all the same distance from the sink in hops. A mix-ring is selected from the available rings to mix packets from the network together. As part of DORing, at a certain time interval every node sends a message to the mix ring. If no ⟨normal⟩ messages are available to send, then ⟨fake⟩ messages are sent instead. ⟨Fake⟩ messages are dropped after a certain number of hops. In the mix-ring packets are routed both clockwise and counter-clockwise, ⟨normal⟩ messages are forwarded to the sink after a certain distance has been travelled. To reduce the energy impact on a specific set of nodes the mix-ring eliminates duplicate packets travelling around it. The mix-ring also changes location to spread out its energy impact.

2.4.3 Phantom Routing Vulnerabilities

Whilst Phantom Routing has been a popular area of research, a number of issues have been encountered with this style of SLP technique.

One of the key components of phantom routing is the *directed random walk* which is used partly due to the *memoryless* nature of these walks on a graph. The directed random walk is memoryless in the sense that the next node in the walk depends on the current node, but not the previous nodes in the walk. However, if an attacker is monitoring the nodes on the network boundary it is possible to analyse the distribution of where the walk first hits the boundary to infer the source's location [165]. Techniques need to be developed to avoid leaking information in this way.

The majority of work assumes a single source, in [59] multiple source configurations were investigated where the sources were close together. This configuration is interesting as it may occur when monitoring flocks of animals that have grouped together but are detected by different nearby sensor nodes. However, Phantom Routing performed poorly, because the multiple individual phantom routes are unaware of each other. So, when the sources separately try to allocate a phantom node, they instead lead to the attacker being *funnelled* towards to group of real sources.

2.5 Fake Source Based Techniques

Fake source-based techniques are different to routing-based in that instead of altering the route of the ⟨normal⟩ messages, ⟨fake⟩ messages are instead used to provide SLP. ⟨Fake⟩ messages are encrypted and padded to be indistinguishable from ⟨normal⟩ messages [81]. Fake sources broadcast the ⟨fake⟩ messages in such a way that lures the attacker towards them instead of the real source.

The seminal work [81] was the first work to investigate using fake sources to

provide SLP. Two techniques were proposed the *Short-lived Fake Source* strategy and the *Long-lived Fake Source* strategy. In the short-lived strategy when a node receives a ⟨normal⟩ message it chooses to send a ⟨fake⟩ message with some probability and flood it to the rest of the network. The persistent strategy has nodes continue to generate and flood ⟨fake⟩ messages after the first one. Neither technique performs well and the authors dismiss the usefulness of fake sources.

A rebuttal was published in [75] showing that fake sources can provide SLP. The authors introduced a fake source allocation algorithm based on the observations that permanent fake sources outperform temporary ones and that the fake source should be allocated at a similar distance from the sink as the real source [81]. Two variants were proposed, FS1 and FS2. In FS1 ⟨fake⟩ messages are flooded from the intermediate nodes with a certain probability, whereas in FS2 the ⟨fake⟩ messages are only sent from the nodes a target number of hops away from the sink. Results show that the attacker's capture ratio can be reduced to 40% compared to its performance against protectionless flooding.

This algorithm was improved in [76, 78] to use temporary fake sources to lure the attacker away from the source to a point in the network where a permanent fake source was to be allocated. The work also proved that fake source selection problem (FSSP) — the problem of selecting which nodes should become fake, at what time and for how long — is NP-complete. The authors also looked at a wide selection of parameter values: temporary fake source duration, fake source rate and fake source selection probability. Different network layouts were also investigated, such as when the sink was located in a corner and the source in the centre, forcing temporary fake nodes to be allocated around the source. In these situations different responses were required compared to when there are nodes adjacent to the sink that are further from the source than the sink is.

The work in [179] showed that it is important to consider what rates nodes will broadcast at. The more frequently real and fake sources broadcast the greater the chance of collisions which can lead to a decrease in the level of SLP provided. Finally, the authors showed that deceasing the rate at which the source sends messages to increase delivery ratio and reduce energy usage, can lead to reduced provision of SLP.

The Path Extension Method (PEM) was presented in [174]. PEM creates fake sources dynamically when the sink receives its first message from the source, these fake sources are then used to form multiple fake paths. Initially fake sources are allocated on the shortest path between the sink and source. After some delay new fake sources are chosen from an existing fake source's 1-hop neighbourhood, only nodes not on the shortest path between sink and source and nodes that are a

certain distance from the real source are eligible. ⟨Fake⟩ message broadcast rates are initially high to capture the attacker and are then slowed down to save energy. The source will send messages to the sink on the shortest path and fake sources will send ⟨fake⟩ messages along the path of nodes previously allocated to be fake sources.

The main complaint that comes with fake source solutions is the increase in energy cost [170]. This is because there can be a large number of fake sources generating ⟨fake⟩ messages this leads to more radio activity and less sleeping compared to routing-based SLP protocols. Another issue with fake source is that when multiple sources are present it becomes harder to provide SLP due to the increase in collisions [93]. However, these complaints are typically against specific fake source techniques and each approach's energy cost and reliability should be individually investigated with respect to the intended application [61]. For example, a short-lived WSN deployed on a battlefield may want to prioritise SLP provision over low energy usage, so a high energy cost is not a significant negative aspect.

2.6 Hybrid Techniques

While many techniques focus on providing SLP through just the routing protocol or just by using fake sources, others use a hybrid of these two techniques. These hybrid techniques can achieve better performance because SLP provision can come from both the routing protocol and ⟨fake⟩ messages. However, these techniques often encounter both costs: a high latency from the way messages are routed and a high energy cost by using ⟨fake⟩ messages.

In [115] a tree structure was overlaid on the network along which ⟨normal⟩ and ⟨fake⟩ messages travel. Initially, a backbone path is established from the sink to the network border. Along the border numerous branches are created, at the end of the branch a fake source is allocated which broadcasts ⟨fake⟩ messages along the branch and to the backbone which sends them to the sink. When the asset is detected the source node will send a ⟨normal⟩ message along the branch it is on to the network's edge and then the message will be sent to the source via the branch and then the backbone. This approach is similar to CTP, but uses ⟨fake⟩ messages to obscure which branch the source is located down.

A hybrid of fake sources and Phantom Routing was proposed in [159] with the technique called Source Location Privacy Using Fake Source and Phantom Routing (FSAPR). To allocate the fake sources the real source first floods a `fake_req` message to nodes h hops from it. Nodes are chosen to be a fake sources probabilistically based on the frequency they have acted as fake sources in the past and their energy level. Phantom routing works as it normally does. The author's fail to describe the

rate at which the fake sources broadcast or the period of time they last for, both of which have been shown to be important [179].

Another popular technique is that of fog routing [44], where messages are routed from source to sink via a number of fogs. A fog is a group of adjacent nodes in which messages are routed between themselves. Fogs can either be routing real messages or fake messages between their member nodes. Messages will be passed between fogs and real messages use these inter-fog paths to be routed from the source to the sink. Fogs are created around areas of real sources and also areas without any sources. Two downsides of this technique are that the time it takes a message to reach the sink is increased, and a large number of nodes are required to support multiple fogs. However, greater privacy is provided compared to Phantom Routing.

A similar technique to fog routing is to create clouds of fake messages [117]. The main aim of this technique is to prevent the attacker from being able to locate groups of assets. As these multiple assets are grouped together it causes an inconsistency in the traffic pattern compared to if a single asset was being reported about. Multiple clouds are allocated in which fake messages are sent. A real source then sends a message to a fake source before it is forwarded to the sink, meaning the attacker will be unable to discern which node in the cloud the message originated from. The packets appearance is also changed at each hop by encrypting it with different keys. This further decreases the attacker's ability to follow specific packets.

2.7 Other Approaches

So far the solutions presented have focused on developing new routing strategies to avoid leaking the location of the source to an attacker. These techniques typically assume limited capabilities in terms of the hardware available to sensor nodes and thus limited actions they can perform. However, other approaches have added specific capabilities to sensor nodes or to the network that allows SLP to be provided using different mechanisms.

Under a different attacker model, [175] developed a linear programming (LP) model that varied node transmission range to avoid the attacker being able to eavesdrop messages. The authors argued that an attacker would make observations from afar before attempting to get closer to capture the source. This is to reduce the risk of detection from the attacker's perspective. When the LP model attempted to maximise the lifetime of the network (the time the first sensor node exhausted its power supply) the vulnerable area can be up to three times the size of the sensing area. However, with a reduction of network lifetime by 10% the vulnerable area could be reduced to the sensing area.

The work in [133] presented a technique that uses coordinated jamming to prevent an adversarial localisation system from gaining useful information about the radio transmissions. The idea in this technique is to prevent an attacker from being able to localise an intermediate node along a path. If an attacker is incapable of doing that, then it cannot trace the entire path back back to the source. However, this technique does decrease the throughput of the network and increase the energy cost. This work assumes the attacker uses information such as the time of arrival and received signal strength to localise an intermediate source, so the solution may not be sufficient to defend against an attacker equipped with one or more directional antennas.

A network coding approach was used in [51, 52] to provide SLP. Fake sources and ⟨fake⟩ messages are used to generate streams of traffic to mislead the attacker. Network coding is used to absorb ⟨fake⟩ messages at certain nodes to prevent an explosion in the quantity of traffic. Another useful property of this solution is that by using homomorphic encryption ⟨fake⟩ messages are made indistinguishable from ⟨normal⟩ messages.

In [134] the SLP problem was recast as the "Panda tracker problem", where a robot was deployed to track pandas moving through their habitat. The attacker tried to locate the source by monitoring the wireless traffic from the robot, or by compromising the robot's memory. In this situation there was no network to route messages over, and instead the authors focused on how much information the robot gathered. By lowering information gained by sensors that detect assets, better privacy was able to be provided.

2.7.1 Data Mules

One of the capabilities that WSNs may use to provide SLP are data mules. Typically data mules are used to assist with the delivery of messages between two disconnected areas of the network that are incapable of communicating via wireless links, or to reduce the energy cost of transmitting [86]. The paper does not state the form data mules take, but they may be in the form of UAVs or other mobile agents. The Mule-Saving-Source (MSS) protocol was introduced in [98, 150], where messages are initially routed away from the source to some phantom node a data mule collects the packets. The data mule then moves to a different location in the network and broadcasts the message along the shortest path to the sink. The location is chosen to confuse the attacker as to the direction of the source. An alternative called Direct Delivery (DD) has the mules deliver the messages directly to the sink with the trade-off being a higher latency due to the time cost of it moving through the network. The downside of using data mules is that additional hardware is required

as part of the network deployment. One the key aspects of WSNs is that they are cheap to deploy, adding a UAV, another kind of robotic mule, or having people drive a vehicle around to gather messages adds to the cost of deploying and maintaining the sensor network.

An alternate work [167] investigated the impact of multiple data mules passing data between themselves when they are near. The network was initially subdivided and data mules patrol a specific area. When the mules are near one another they forward messages. A comparison to DD was performed and the technique proposed in this work provided a lower message latency.

Another approach that is similar to data mules is PASSAGES [141], where "permeability tunnels" (PT) comprised of wormholes and mobile ferries obscure the source's and sink's locations. A wormhole is a link between two nodes that cannot be eavesdropped by an attacker, such as a wired cable or a long range directional wireless signal. Mobile ferries are conceptually similar to data mules, an example would be the XM1216 robot [2]. By routing messages through different PTs the source and destination can be obscured from an attacker with global visibility. However, the wormhole component of this work is unlikely to be practical in real-world scenarios. Deploying infrastructure like a cable may often be impractical (on a battle field, or in a large forest). Also an attacker would likely be capable of eavesdropping on a long range wireless link, making parts of this work unlikely to be usable in practice.

2.7.2 Cross Layer

Rather than solely relying on message routing, some SLP techniques have also used aspects of the MAC layer to provide SLP. For example, the cross-layer approach presented in [164] used the periodic beacon messages sent by the 802.15.4 MAC layer to propagate messages. As the beacons occur periodically, it is impractical to send data all the way to the sink via beacons due to the high delay incurred and limit imposed upon the data rate. The cross-layer solution aims to reduce the latency cost by first propagating messages short distances and then switching to routing the messages as usual. To improve privacy further, rather than routing the message direct to the sink, another round of propagating (in a different direction) via MAC beacons can be performed. This technique can be viewed as a hybrid of the Periodic Collection global privacy solution in which every node periodically broadcasts and phantom routing.

To eliminate collisions Time Division Multiple Access (TDMA) has been used as a way to allocate each node a time slice in which they can broadcast. In [88] an algorithm to setup a TDMA slot assignment was presented, where the slots were

[2] army-technology.com/projects/xm1216-small-unmanned-ground-vehicle-sugv

allocated in such a way that following the messages would trap the attacker rather than lead it to the source. An extension was presented in [89] where the TDMA slot assignment was solved using genetic algorithms. The algorithm has each node always broadcast a message in its time slot, even if there are none in its queue to send. This kind of solution is very energy efficient as nodes are aware of when to sleep and when to wakeup because neighbouring TDMA slots are known. However, the technique imposes a high latency because a node must wait for its slot to come around before sending another message. Other techniques can instead just send the message immediately, which leads to a low latency.

2.7.3 Cryptographic Approaches

Whilst many techniques have either relied on adjusting the routing protocols or using fake sources to provide SLP, there have been a number of different techniques.

In [128] a technique called DCARPS was proposed. It uses layered cryptography (similar to onion routing) to prevent an attacker from being able to detect if a packet is new or not. This is because a layer of encryption is removed in order to send the packet on the next link. Label switching is also used to obscure the identity of the source and sink. When sufficient traffic flows are present, DCARPS is capable of protecting against both an attacker with local and global network visibility.

SPENA [148] uses cryptographic techniques, such as one-way hash chains and mapping functions, to provide SLP. The attacker assumed is more capable than typical, as it has a longer eavesdropping range compared to the sensor nodes and is capable of compromising the sensor node's hardware to obtain all information stored on the node. SPENA works by hashing the source information (which can be retrieved by the sink using its knowledge of the nodes in the network and the hash function). When a packet is received its hash can be modified before being broadcasted after some delay. This approach prevents an attacker from being able to link a broadcast with a source detection event.

2.8 Global Source Location Privacy

The solutions presented thus far have focused on a mobile in-network attacker with a small eavesdropping range, which is the attacker assumed in this work. An alternative capability is when the attacker has global visibility over the entire network. This can be achieved through using a single very powerful receiver or a network of many low power receivers. The important aspect is that the attacker can see all events as they occur in the network. This means that a different approach needs to be taken compared to the solutions against a local attacker that have been covered thus far.

This is because a global attacker can see where a message originates from, if flooding or single-path routing is used, as soon as a message is sent.

The first solution has been presented as the Periodic Collection algorithm [122] and the Constant Rate algorithm [163]. Instead of just the node that has detected a source generating a message, every node will generate a message. If a source has been detected then a ⟨normal⟩ message will be added to a queue of messages to be sent every generation period. Every broadcast period a message will be sent, if the queue is not empty then the ⟨normal⟩ message at the top of the queue is popped and sent. If the queue is empty a ⟨dummy⟩ message is to be sent instead. Like with ⟨fake⟩ messages, the ⟨dummy⟩ message must be encrypted and padded to be indistinguishable from the ⟨normal⟩ messages. With this scheme every node will send a message every broadcast period no matter if there is a real message to send or not. This prevents the attacker from finding any information about the location of the source as it cannot be sure that the first broadcast detected was actually from a source node.

An extension proposed to the Constant Rate algorithm was the Fitted Probabilistic Rate algorithm [163, 195]. Instead of using a constant rate of message broadcasts the FitProbRate algorithm broadcasts messages every period, where the period is chosen with respect to a probability distribution. As long as the messages follow the probability distribution then the attacker cannot perform any statistical analysis on the broadcast timings to gather information about a source's location. The benefit of this solution over the ConstRate algorithm is that the period between a node sending messages is not fixed. As the period varies it allows the algorithm to prioritise the ⟨normal⟩ messages over the ⟨dummy⟩ messages. This allows the FitProbRate to send a ⟨normal⟩ message at the earliest time that would not skew the probability distribution. Compared to ConstRate, FitProbRate can achieve a much lower latency between the source sending the message and the sink receiving ⟨normal⟩ messages because they can be prioritised.

Both the ConstRate and FitProbRate algorithm are essentially flooding algorithms, however, the way they introduce time delays, ⟨dummy⟩ messages, and prevent the attacker from being able to perform analysis on the time at which messages are sent allow them to provide SLP. An alternative solution that was proposed by Mehta et al. [122, 123] was Source Simulation. The problem with Periodic Collection was that it consumes a large amount of energy and also has a high latency. Source Simulation achieves lower energy usage and a lower latency in exchange for a higher capture ratio. The algorithm provides SLP by creating multiple fake source traces through the network. These traces are created by modelling the behaviour of the asset and trying to match its behaviour. The fake sources will be

sending ⟨fake⟩ messages as they move throughout the network. This is similar to the fake source solution proposed in Chapter 6 except that it is designed to mislead a global attacker rather than a local one, making the design approach and behaviour different.

Since these initial pieces of work, there has been a lot of focus on developing techniques to protect against a global attacker. As ConstRate and Periodic Collection provide optimal SLP, many of these solutions aim to improve another metric (such as energy consumption or latency) in a trade-off with the SLP provided. In [130] OSAP, a cluster based solution, can trade-off between the amount of network traffic and message delivery latency. Another solution used data aggregation alongside broadcasting messages with a period determined by an exponential probability distribution in [194]. By aggregating data fewer data packets needed to be sent, saving energy.

2.9 Additional Aspects

In the literature the majority of the solutions against a local attacker assume that there is only a single one present in the network. The fake source solution in [75] is one work that investigated multiple attackers, where the network was divided into four quadrants and one attacker was assigned to each quadrant. Once an attacker receives and responds to a message, no other attackers will respond to that message again. The authors found that these multiple attackers cause an increase of 100% in the capture ratio (from roughly 40% to 80%). This was associated with one attacker receiving many ⟨fake⟩ messages in one quadrant and another focusing on the ⟨normal⟩ messages in another quadrant.

There have also been a number of works that do not attempt to introduce new techniques but present insights into the problem. In [74] it was proposed that SLP protocols have three components: (1) selection of decoys (such as the location of phantom nodes or fake sources), (2) use and routing of control messages, and (3) use and routing of decoy messages (such as ⟨fake⟩ messages). These components then either have a temporal, spatial or hybrid nature. Not all of these components need to be implemented by SLP-aware protocols. For example, Phantom Routing does not implement the third component. The conclusion of this work is that there needs to be some form of redundancy in the network to allow SLP to be provided. Either in terms of space (such as an area to lure the attacker to), or in terms of time (such as delaying the attacker by causing it to miss receiving messages).

As is evident by the number of techniques presented thus far, there is a large number of SLP protocols to choose from. The work in [61] presented a

decision theoretic methodology to select which protocol is appropriate for different deployments. The authors focused on four metrics: the capture ratio, the delivery ratio, the number of messages sent, and the delivery latency. Users of the methodology need to supply custom utility functions that specify how to weight the metrics and rank the goodness of the results. Results in the paper are obtained through simulation, but could be supplied by real world experiments. A variety of techniques were investigated: *Protectionless* flooding, CTP [57], *DynamicSPR* [25], *ILPRouting* [21], Phantom Walkabouts [60], and Phantom Routing [81]. Sample utility functions were demonstrated that focused on different metrics, showing certain algorithms outperforming others.

In the future it is likely that SLP-aware routing protocols will take advantage of new hardware developments. Most techniques thus far assume basic hardware without additional functionality, but new capabilities would allow for new SLP protection strategies. For example, GNSS radios that consume little power or other localisation strategies would allow SLP-aware geographical routing that takes advantage of the physical layout rather than just the network communication topology. By doing this SLP-aware routing could aim to take an attacker over areas that are slower to traverse compared to existing solutions. Another example would be low power wake-up radios [18], which could be used to optimise the broadcast schedule of an SLP-aware protocol. Alternatively, they could be potentially used to detect an attacker's presence with a low energy cost and have messages routed around the attacker. New kinds of low power hardware will enable new classes of SLP protection strategies.

2.10 Other Context Privacy

Along with source location privacy there are two other main types of context privacy that are important to consider in WSNs. Some solutions aim to just provide this type of context privacy, while others provide multiple kinds of context privacy within the same solution.

2.10.1 Sink Location Privacy

Similar to source location privacy, *sink* location privacy is another aspect of context privacy that instead focuses on protecting the location of the sink. In this work the sink is assumed to be a well known location, such as a military base or a camp operated by conservationists, so the solutions presented by this thesis will not focus on sink location privacy. However, there are some situations where the sink's location is not obvious and it becomes important to protect that location. Many of the

techniques developed aim to protect both the sink and the source nodes.

One of the first works to investigate sink location privacy was in [41, 42]. A capable attacker that undertook *rate monitoring* and *time correlation* of messages routed through a tree structure was assumed. The authors showed an attacker was easily able to detect its proximity to the sink based on the volume of messages that were being forwarded. The solutions presented take multiple approaches to provide the context privacy. To provide sink location privacy a combination of: (i) nodes forwarding packets to their parents instead of children (in the tree), (ii) a random walk is introduced on the way to the sink, (iii) fake paths are used to lure the attacker away from the sink, and (iv) areas of high communication volume are created to increase uncertainty as to how far from the sink the attacker is. The technique is rationalised as the energy cost is only increased by a factor of 2 to 3, but the time taken for the attacker to capture the sink is increased by a factor of 19.

There are many other techniques to provide Sink Location Privacy [36, 99, 129, 184–187], and overall these techniques use similar approaches to source location privacy techniques except that the focus is on protecting the location of the sink. These approaches include, random walks, fake paths, uniform energy consumption, and delay. Several techniques also provide sink and source location privacy. Depending on the scenario, providing sink location privacy may or may not be a requirement. For example, if the base station is a well known location then there is no need to protect it when messages are being routed.

2.10.2 Temporal Privacy

The other type of context privacy that has been investigated is *temporal* privacy, where instead of attempting to protect the *location* of events the *time* at which the event occurs is protected. Aspects of temporal privacy are very useful in providing source location privacy as will be shown in this thesis in Chapter 5.

The first work on temporal privacy was presented in [70]. When an event is detected a message is sent immediately, but is delayed along its route to the sink by a random amount of time that is exponentially distributed. An alternate approach was then presented in [151], where rather than immediately sending a message it was cached locally for some time. After the n^{th} event is detected all stored messages are broadcasted together. The value for n could be a fixed value for all nodes, or determined probabilistically. By storing messages for some time, the time at which older messages occur is obscured.

The authors of the seminal work on SLP then presented their take on temporal privacy in [82, 83], with a solution called RCAD which adaptively delays the packets from being forwarded onwards. The delay increases the attacker's uncertainty of

when the event occurred. Four delaying strategies were presented: (i) LDF which delays the packet that has been in the buffer the longest, (ii) LRDF which delays the packet with the longest remaining delay time, (iii) SDTF which delays the packet with the shortest delay time, and (iv) SRDF which delays the packet with the shortest remaining delay time. Results showed that the LRDF strategy performed the best in terms of privacy against a baseline adversary, but SRDF performed better against an adaptive adversary when traffic rates were high. The adaptive adversary changes the way it estimates event time depending on the rate of traffic.

In [162] a sub-problem of temporal privacy was identified, *Rate Privacy*, where the rate at which events occur is important to protect. Two techniques that the authors identified as being inadequate to provide rate privacy are: (i) deterministic message generation at a specific rate, and (ii) randomised fake event messages. The first greatly increases the delay of being informed that an event occurred and the second increases the energy usage of the network. The solution proposed by the paper maintains a buffer of messages to forward. When a message is received, it is placed in a random position that is empty in the message buffer. After a random delay a node will retrieve the first message in the buffer and broadcast it. This reorders the messages and provides temporal privacy.

2.11 Out of Scope

The solutions presented tend to avoid the capability of detecting an attacker's location and re-routing messages to avoid the attacker. This is because the technique is easily defeated if the attacker finds a better way to evade these sensors, such as by changing the frequency it broadcasts on or by simply not broadcasting at all. However, there exists the problem of adversarial location, where the sensor network tries to locate the attacker itself whilst minimising information leaked to an attacker about sensor node locations [46]. Other approaches have investigated changing how messages are routed when an attacker is detected [156]. While such information would be useful to an animal tracking deployment, its use is out of scope for the SLP protocols developed in this thesis.

Another problem is ensuring that the event location reported is correct, as an attacker may want it to seem like events are occurring at different locations than they actually are [72]. For example, the attacker may want to mislead the conservationists as to where animals actually are to give themselves more time to steal the animals. Techniques have been presented to detect fake or inconsistent event location claims, however, the work in this thesis focuses on protecting the location of a message broadcasting node even if the event it has detected is not real. Ensuring that the

event the node has detected is real is left to secure localisation techniques.

Some sensor networks will contain anchor nodes that broadcast messages containing their location which other nodes in the network will use to triangulate their position. In [48] techniques are developed to prevent untrusted nodes from being able to use the signal strength to locate anchor nodes. The technique dynamically changes the power used to transmit beacon frames based on a discretised exponential distribution. However, because nodes are not aware of their physical location in this work, it means that anchor nodes would not be present. So providing protection for anchor nodes is unnecessary. However, changing transmit power levels can be a useful way to help obscure location information [175].

It is not just in WSNs where context privacy needs to be provided. A wide variety of other situations also require it. Onion routing is one example that has already been discussed, where the context information about the actual sender and source is obscured from onion nodes along the route packets take. The idea of *path confusion* was introduced in [69] where a sequence of GPS coordinates is perturbed in such a way that the introduced error is bounded by a quality of service requirement. Such obfuscation would have been useful to prevent situations such as when the mobile app Strava leaked the physical layout of military bases[3]. Another example is protecting a user's location in location based services [112], where a user's actual geographical location is obscured but the user is still able to use the mobile application effectively. This sort of privacy preservation could be extended to a number of other situations, such as hiding the identity of rumour sources on social media, privacy preserving data mining [16], and others.

2.12 Summary

Overall, there are a large number of techniques that provide SLP. Two of the biggest categories are (i) routing messages via a phantom node and (ii) using fake sources. The aim in both of these techniques is to lure the attacker to a different position in the network other than the real node. These routing protocols have a variety of performance characteristics, including how well they protect the source's location, the amount of energy they consume, how long it takes a message to reach the source, and others. Other categories of techniques also exist, but these usually assume additional network capabilities (such as data mules) that limit their use.

[3] `http://www.bbc.co.uk/news/technology-42853072`

Chapter 3

Models and Experimental Setup

In order to investigate ways to model SLP and develop techniques to provide SLP, the problem first needs to be formally stated. In this chapter the formal definition of the SLP problem will be described in terms of the Panda-Hunter Game. This definition describes the problem at a high level and contains a number of components that are described in more detail. The attacker model and privacy model explain assumptions about how the attacker behaves and what the aims are when providing SLP.

This chapter also describes how the abstract models are applied when testing solutions experimentally, which is undertaken to quantify the performance of the solutions. There are two main options to perform the testing: simulation and deployment on real world hardware. There are advantages and disadvantages to both approaches. For example, simulation allows a wide array of scenarios to be tested, but running on real hardware allows the actual behaviour to be examined. As both approaches have merit, both techniques will be used to evaluate the developed algorithms.

In order to simulate and deploy these algorithms a number of choices will need to be made. The four main choices are: (i) what Operating System will the algorithms be written in, (ii) what configurations will they be tested for, (iii) what simulator will be used, and (iv) what testbeds will the algorithms be deployed on. This chapter will detail these choices, as well as describing how the experimental environment was set up, and how the experiments were performed. Finally, an example execution of *Protectionless* flooding will be examined to give a practical example of the information available to the attacker and how it responds to eavesdropped messages.

3.1 Panda-Hunter Game

The Panda-Hunter Game model is based on the model proposed by [81]. The aim is for the attacker to find the location of the source s by tracing back the messages sent by some routing protocol $\mathcal{R}$. The aim of network maintainers is to modify or replace $\mathcal{R}$ such that the attacker fails to capture the source. The implementation of an SLP-aware routing protocol can occur in a number of ways which will be investigated in this thesis. The game represented by the five-tuple $(G, \mathcal{Q}, \mathcal{S}, \mathcal{R}, \mathbb{A})$, where:

$G = (V, E)$ defines the graph where $V \subset \mathbb{Z}$ is the set of sensor nodes in the network and $E \subseteq V \times V$ defines the communication links between nodes.

$\mathcal{Q} \subset V$ are the network sinks, to which all communication is ultimately routed to. Typically there is a single sink q.

$\mathcal{S} \subset V$ are sensor nodes that have detected an asset. Typically there is a single source s. Assets are characterised by a mobility pattern $\mathcal{M}_s$. Nodes cannot be both a sink and a source $\mathcal{Q} \cap \mathcal{S} = \emptyset$.

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

$\mathbb{A}$ is the attacker, who seeks to acquire or capture a source in $\mathcal{S}$ through a set of movement rules $\mathcal{M}_{\mathbb{A}}$.

The following sections will expand on this representation and explain aspects of the Panda-Hunter Game further. Section 3.2 will detail the network model. The focus of the routing protocol will be briefly described in Section 3.3 as the different implementations in Chapters 5 and 6 will describe their respective routing protocols in depth. When SLP is considered to have been provided is detailed in Section 3.4, and how the attacker behaves is described in Section 3.5. These sections will also state the assumptions made that the SLP-aware routing protocols depend upon.

3.2 Network Model

A wireless sensor node is a device with a unique identifier that has limited computational capabilities and is equipped with a radio transmitter for communication. A WSN is a set of wireless sensor nodes with communication links between pairs of nodes. The sensor network is modelled as a graph $G = (V, E)$ where V represents the set of nodes and E is a set of unordered pairs that represent bidirectional links between the nodes. In some cases the network will be modelled as $G = (V, A)$ where A is a set of ordered pairs that represent unidirectional links. When a link exists

between two nodes m, n, then m and n can directly communicate with each other. Nodes that can communicate are called neighbours and the set of neighbours of a node n is represented by $1\text{HOPN}(n)$ for networks with bidirectional links.

There exists a distinguished node in the network called a *sink*, which is responsible for collecting data and which acts as a link between the WSN and the external world. Other nodes sense data and then route the data via $\langle \text{normal} \rangle$ messages along a computed route to the sink for collection. It is expected that there are multiple hops between the source and sink. Any node, except for the sink, can be a data source. It is assumed that the network is event-triggered, i.e., when a node senses an object, it starts sending messages periodically to the sink for a certain amount of time.

The messages sent are encrypted and the source node includes its identifier in the encrypted messages. The type of encryption, be it end-to-end, pairwise or some other scheme is left undefined. Using the identifier, the sink can infer an asset's location as it is assumed that the network administrators will record where they put nodes. Nodes are not assumed to know their geographical location by being equipped with Global Navigation Satellite System (GNSS) chips (such as GPS), which is due to the increase in energy cost that would be incurred.

The modelling performed in later chapters will assume that network links are reliable and bidirectional to simplify the analysis. However, these assumptions will not hold in practice, so the implementation of the techniques that will be presented in this thesis will be developed to handle unreliable links and other reliability issues such as message collisions. The impact to the translation of the individual modellings to an implementation will be small and shall be described in the chapters that perform the modelling.

3.3 Routing Protocol

In WSNs, a routing protocol is required so data can be transferred from a source node to the sink node. It is important for the routing protocol to deliver a high percentage of messages from the source to the sink over multiple hops. The routing protocol is considered to be a set of paths (a path is a sequence of communication links between pairs of nodes) and a message will travel along one of the paths to the sink. Each message may follow the same path or messages may follow different paths to the sink. The way the routing protocol executes depends on the topology of the network, as messages can only travel over links present in the network. The attacker will make use of the routing protocol to locate the asset.

There are two approaches when considering the routing protocol. Either

the protectionless routing protocol remains the same and is augmented to provide SLP, or a new SLP-aware routing protocol is developed. Both approaches will be investigated in this thesis to provide SLP.

3.4 Privacy Model

The overall objective of any WSN-based SLP solution is to ensure that the asset (at a specific location) is *never* captured through information leaked by the WSN. However, if the asset is stationary, then the attacker can perform an exhaustive search of the network to find the asset. In this case the SLP problem is irrelevant as the attacker will always capture the source. If there exists no time bound on the capture time, then an exhaustive search is a trivial yet effective solution. On the other hand, if the asset is mobile, then performing an exhaustive search of the network is unsuitable, as the attacker may hone in on a given location only to find out that the asset has moved.

Thus, the SLP problem can only be considered when it is time-bounded. This captures the maximum amount of time a mobile asset will spend at a given location. This notion of time bound has been termed as *safety period* in the literature. There are two alternative definitions of safety period: The first, used primarily by routing-based techniques, e.g. [81], is where the safety period is defined as the time required to capture the asset. The aim of these techniques is to maximise the safety period, i.e., the higher the time to capture, the higher the SLP level provided.

The second notion of safety period is used where it is desirable to bound the amount of time SLP is being considered for, i.e., if an attacker fails to capture a source within the specified safety period, then SLP is said to have been provided. That notion of safety period intuitively captures the maximum time an asset will be at a given location before its next movement. Often, this can be obtained from previous data gathering to know more about such mobile assets.

This second notion of safety period is more generic than the first one in that, rather than attempting to maximise the amount of time an asset isn't captured (as under the first definition), the second definition captures the fact that the asset can't be reached before a certain time limit, i.e., setting the time limit to be ∞ in the second instance results in the first definition.

In this thesis, the time bound model of safety period is used. In order to calculate the safety period (P_{safety}), for some parameters ($\mathcal{P}$), the average time the attacker takes to capture the source ($\mathcal{TT}$) under *Protectionless* flooding is multiplied by a safety factor ϕ. In this work the safety factor is set to 2. Different safety factors

have been investigated in other work [61].

$$P_{safety}(\mathcal{P}) = \mathcal{TT}(\mathcal{P}) \times \phi \tag{3.1}$$

3.5 Attacker Model

It was proposed in [13] that the strength of an attacker for WSNs could be factored along two dimensions, namely *presence* and *actions*. Presence captures the network coverage of the attacker, while actions capture the attacks the attacker can launch. For example, presence could be local, distributed or global, where a *local* attacker has a view of a subset of the network that does not change, a *distributed* attacker views a subset of the network that can change as it moves location, and a *global* attacker can see the entire network (such as by deploying its own WSN, or by using long range antennas). Some examples of actions could be eavesdropping, reprogramming, or emitting wireless interference among others.

The attacker assumed in this thesis is a *distributed eavesdropper* (distributed due to mobility) based on the patient adversary, introduced in [81]. A distributed attacker is chosen because the attacker can move around the network to gather information. The only other action that it can perform is eavesdropping. Such an attacker is reactive in nature and proceeds as follows:

1. The attacker initially starts at the sink.
2. When the attacker is co-located at a node n and eavesdrops a new message that has not been received before, from a neighbour node m, the attacker will move to m. Thus, in a normal setting, the attacker is geared to moving closer to the source as it only follows unique messages.
3. Once the source has been found, the attacker will no longer move.

Previous work [81] has assumed that the attacker has the ability to identify whether a message has been previously received and moved in response to. This assumption is also made in this work. If this assumption is not made, then a routing protocol such as flooding would lead the attacker away from the source. The attacker is assumed to have the capability to perfectly detect which direction a message arrived from, that it has the same radio range as the nodes in the network, and also has an unlimited amount of memory to keep track of information such as messages that have been heard. This history of received messages is used to determine if a message is new or not.

In this work, the attacker starts at the sink because the sink is the one location in the network where the attacker is *guaranteed* to eavesdrop a message from the

source node, irrespective of the routing protocol used. The attacker could potentially start at any location in the network, however, the attacker may not receive messages due to their location not being on the route from the source to the sink. The sink is assumed to be located at a base known to the attacker such as a military base or a field station used by scientists monitoring wildlife. A routing protocol (such as flooding [81]) that provides little protection to the source location is called a *protectionless* routing protocol.

3.5.1 Capabilities

This attacker model lies on the bottom of total order of attacker capabilities shown in Figure 3.1. This is because some stronger attacks would weaken the the attacker's ability to capture the source by leaking information to the WSN regarding the attacker's position. For example, if the attacker attempted to disrupt the functioning of the network (e.g., by a DoS attack), it would reduce the amount of useful information the attacker could gather. If an attacker attempted to broadcast messages to influence the SLP or routing protocol, then the WSN could potentially detect an intrusion attack and respond by ceasing to broadcast around the attacker (similar to [127]).

eavesdrop $\rightarrow$ *crash* $\rightarrow$ *disturbing* $\rightarrow$ *limited passive* $\rightarrow$ *passive* $\rightarrow$ *reprogramming*

Figure 3.1: Attacker capability hierarchy proposed in [13].

Performing certain attacks such as breaking into a sensor node to obtain encryption keys (i.e., passive attacks) *are* good strategies for an attacker trying to defeat SLP. The problem with such an attack is that it is also time consuming. For example, [13, p. 11] predicts that a key stealing attack will take around 30 minutes to perform in the field (not counting preparation time elsewhere or the time it takes to find, obtain and open a sensor). As the solutions presented in this work aims to provide a high level of SLP within a specific safety period, if the time taken to obtain encryption keys is larger than the safety period then the attacker will have failed to capture the source within the safety period. This means the attacker would have achieved better results by simply eavesdropping. Thus, in the context of SLP in WSNs, one of the most powerful type of local attackers that can be had is the distributed eavesdropper which is assumed in this work.

3.5.2 Reasons Against Global Visibility

As this work focuses on a distributed attacker with a small visibility of the network at a given point in time, the techniques are not designed to protect against a global

attacker. The reason for this is that an attacker physically present in the network is a much more likely threat to need to defend against. For an attacker to gain global visibility they will need to expend significant resources to either: (i) deploy their own WSN to monitor the WSN that is monitoring the assets, (ii) build towers with sensitive long range directional antennas, or (iii) deploy many attackers present in the network each with a small visible range. Each of which comes with downsides. For example, deploying a sensor network capable of triangulating the source node will be expensive, building towers with long range antenna is going to be noticed by personnel monitoring the assets, and having more attackers in a network increases the chance of detection and capture of the attackers. A single attacker with distributed visibility is much more likely as cheap commodity components (a laptop, a SDR dongle [147], and a directional antenna) can be used to sufficiently equip the attacker. Therefore, this work focuses on a single attacker with visibility that changes over time as the attacker moves through the network.

3.5.3 Simulation of the Attacker

The attacker is simulated by parsing log messages outputted by either the simulation or testbed. The attacker moves from one node's location to another node's location when required by its movement rules. It cannot be simulated by an independently moving node as neither the simulator nor testbeds support this. This is why the attacker must be co-located with another sensor node in the network. When a node n receives a message, if the attacker is co-located with n, then the attacker will also receive that message. This is why the attacker is assumed to have the same range as the wireless sensor nodes. When a new message that has not been previously seen is received by the attacker, it will move in response to that message. If the attacker were to move in response to messages it has previously received then it can be defeated by *Protectionless* flooding, as the attacker will follow the flood away from the source. Once the source has been found the attacker will no longer move, as the attacker has obtained its objective of capturing the source.

3.5.4 Summary of Attacker Assumptions

As a number of assumptions about the attacker model have been presented in this section, they are summed up below for clarity. In the seminal work [81] assumptions 1–10, 12, and 14 are made, assumption 11 is not discussed, and assumption 13 is investigated along with other attacker eavesdropping rages. Similar assumptions are made in other work that addresses location privacy against a local attacker [21, 38, 41, 60, 63, 78, 120, 127, 137, 168, 174].

1. The attacker is a person physically present in the network
2. The attacker is aware of the location of the sink node that messages are routed to
3. The attacker starts at the sink as that is the one location guaranteed to receive messages from the source
4. The attacker has sufficient directional antennas to detect the direction from which a message originates
5. The attacker is capable of determining if a message has been previously seen before
6. The attacker has unlimited memory to record the messages it has moved in response to
7. The attacker moves to the location of the proximate source when it receives a new message
8. The attacker's visibility of the network changes over time as it moves through the network
9. The attacker cannot tell the difference between an encrypted ⟨normal⟩ message and an encrypted and padded ⟨fake⟩ message
10. The attacker does not have a global view of the network
11. The attacker does not transmit any wireless signals, such as jamming the network (as that would reveal its position)
12. The attacker does not do complicated traffic analysis or physically attack the network (as this takes time to perform, delaying its progress towards the source within the safety period)

Due to limitations in the simulator and the way a mobile attacker is emulated on the testbed, the following assumptions are made:

13. The attacker has the same wireless range as the sensor nodes
14. The attacker must be co-located with a sensor node, it cannot be located at arbitrary coordinates

3.5.5 Discussion on Attacker Assumptions

One of the assumption made is that the attacker must be co-located with a node, due to simulator limitations. This means that the attacker will not exist at positions between nodes. The implications of this are that an attacker that can exist at positions between nodes will not receive the same messages at the same times as an attacker that can only be co-located with nodes. This pattern of movement will not greatly affect which messages the attacker receives (and thus the decisions it makes) as the attacker will be in a similar area of the network when it either moves discretely between nodes, or when it can move in the space between nodes.

Another assumption is that the attacker has infinite memory and the ability to perfectly tell if a message has been received before. It is unlikely that sufficient headers would be exposed, or that the encrypted contents would reveal sufficient information for an attacker to be able to perfectly detect if a message is new or not. A reason for including this behaviour is because if the attacker responds to all messages then *Protectionless* flooding would provide SLP, as the attacker would follow the wave of ⟨normal⟩ messages away from the source. One possible way to prevent this behaviour is to have the attacker keep a travel history and not go back to the previously visited nodes, however, the attacker can possibly become stuck if it travels in a loop. An alternative would be for the attacker to ignore messages for a short time period after moving, however, this means that the solutions would be limited to protecting against attackers of a specific speed and potentially causes attackers to miss receiving new messages they would want to respond to. By giving the attacker this stronger assumption, techniques can be tested against a larger class of attackers than using other alternatives.

In terms of the attacker movement rules, this attacker moves in response to receiving new messages that have not been previously received before. Alternatively, the attacker could wait at one location to gather information in contradiction of assumption 12. After receiving a number of messages it could choose to move based on the information gathered. A problem with this approach is that waiting reduces the number of moves that the attacker could potentially take. If the attacker waits a too long then the safety period will expire and it will fail to capture the source. Note that a proactive attacker (which makes move independently of receiving messages) may be able to benefit from waiting to gather information and then moving independently of receiving messages. However, this thesis focuses on a reactive attacker, so it is out of scope. As a proactive attacker is likely to perform well, it is worth investigating the techniques developed in this thesis against this kind of attacker.

3.6 Operating System and Simulation Tool

In order to implement these models to simulate and test SLP algorithms, a WSN operating system and simulation tool needed to be chosen. There are several OSes and simulation tools for WSNs [172]. Certain simulators have support for several OSes, but other simulators support only one. As two of the most prominent OSes come with their own simulator, the OSes and simulators will be reviewed together. In choosing an OS and simulation tool there were a number of requirements:

1. Code written for an OS should be able to run in a simulator without changes. This avoids the possibility of bugs being introduced while porting code and reduces development time.

2. The simulator should be scalable and fast. As the WSNs that need SLP provided for are expected to contain a large number of sensor nodes, the simulator should be capable of supporting similar configurations.

3. The OS should be widely supported among different testbeds.

4. The OS should come with support for common hardware used by testbeds.

5. The OS should provide direct support for broadcasting and receiving messages. As routing protocols are being developed, the OS should not require that I use one of their protocols.

3.7 Simulators

A variety of WSN simulators exist, many with different features. This section covers a number of the simulators considered for the simulations performed in this work. Then the choice made is stated and why it was chosen is explained.

3.7.1 TinyOS and TOSSIM

TinyOS [96] is one of the oldest WSN OSes and has a number of unique design decisions that allow it to run on hardware with very limited resources. TinyOS focused on (i) running on hardware with limited resources, (ii) reactive concurrency, (iii) flexibility and, (iv) low power consumption. These features are supported because the OS is written in a C dialect called nesC [56]. The main additions to the language are the ability to declare modules and interfaces and specify how they interact. Once the program is compiled how the components interact is fixed and cannot be changed.

TinyOS was initially created as a "platform for innovation" [96] with the aim of being flexible and to enable innovation, rather than being the *right* OS from the

beginning. Whilst this certainly led to high usage in its early years, the complexity caused by that flexibility has hindered its adoption in recent years. One of the major features of TinyOS — nesC and the component architecture — lead to greater development time, as there is a high time cost when learning this new language. This seems to have put many people off learning TinyOS, or at least relegating it to academia. Many new OSes have since been developed that aim to be simpler to understand and use, such as Contiki and RIOT.

Code written for TinyOS can be run in the TOSSIM [95] simulator and also on a variety of hardware. However, TOSSIM only supports code written for TinyOS as it replaces the default ActiveMessageC component with its own that provides an abstraction for how messages are sent and received across a wireless medium. As TOSSIM is deeply integrated into TinyOS it can simulate large networks of up to 1000 nodes quickly. However, the lack of accuracy caused by not simulating a CPU or the associated clock leads to several disadvantages:

- The duty cycle technique Low Power Listening (LPL) is not simulated, meaning the radio is always on and the CPU never goes to sleep.
- Only one hardware platform (MICAz) is supported.
- TOSSIM can only simulate programs written using TinyOS.

3.7.2 Contiki and COOJA

Contiki [45] is an alternate OS written in C. One of the main benefits of Contiki is that it supports the notion of multiple *processes* running concurrently that can be replaced at run time. However, the implementation of certain functionalities are hidden behind macros, which makes comprehending the code harder.

Code written for Contiki can be simulated in COOJA [136]. But COOJA also supports running binaries from other OSes as long as it can be supplied in the correct ELF binary format. Special support is present in COOJA to handle setting the nodes identifier for both the Contiki and TinyOS OSes [50]. As COOJA executes arbitrary binaries it thus needs to simulate the node's CPU. Simulations are therefore very accurate but are slow. This makes running simulations for a large number of nodes impractical. COOJA also supports executing binaries for multiple platforms with different CPU architectures[1]. It supports the MSP430 and ATmega CPUs in the form of support for the TelosB (Sky) and MICAz motes.

Whilst COOJA has proved to be very useful there are some issues with it that do not seem to have been resolved. As reported in [158] there is a timing bug

[1] `https://github.com/contiki-os/contiki/wiki#Platforms`

involving the SPI bus, where the time it takes the hardware to transmit data was being greatly over predicted. This does not seem to have been reported to the Contiki or MSPSim developers. There are also issues with the compiler for the TelosB motes[2] (MSP-GCC 4.6.3) where instructions are generated for the MSP430X CPU instead of the MSP430 CPU. This is the default compiler provided by TinyOS and the Debian OS. Updating to a more recent version of the compiler introduced its own set of issues, so code needs to be written to avoid generating MSP430X instructions.

However, even with issues such as these, COOJA offers the opportunity to test the entire code stack on a relatively accurate simulation of the hardware. A major advantage compared to TOSSIM is that executing code advances a node's clock. Whereas in TOSSIM a very expensive calculation could occur, but the time interface would report no change in the time before and after that calculation.

3.7.3 RIOT

RIOT [9, 10] is a relatively new IoT OS. It supports many more features compared to TinyOS and Contiki such as (i) support for writing in C and C++, (ii) real multithreading support, (iii) modularity, and (iv) a real time kernel. RIOT has a number of downsides that come with providing these features. The first is that it is more resource intensive than TinyOS, meaning there is less space available for user code. The second is that RIOT focuses on higher level routing protocols than developers would typically require, such as RPL and IPv6. As this work involves creating routing protocols it may be harder to reach the broadcast primitives compared to TinyOS or Contiki.

Overall RIOT aims to be much more developer friendly. This was an issue with TinyOS and Contiki, as TinyOS required developers to use their custom nesC language which was based on the C89 version of the C language and Contiki uses macros to hide implementation details. By using Unix paradigms software developers are used to, RIOT is much easier to work with. However, RIOT has no provided simulator. This means that cycle accurate simulators like COOJA or Avrora [180] would need to be used to simulate code. This comes with the same downsides as using Contiki with COOJA, including low scalability and slow simulation speed.

3.7.4 Choice

A variety of other WSN simulators are available, such as OMNeT++ [182] and NS2 or NS3[3]. However, these simulators are unable to have the same code run in both the

[2] `github.com/contiki-os/mspsim/blob/47ae/se/sics/mspsim/core/MSP430Core.java#L455`

[3] `https://www.nsnam.org`

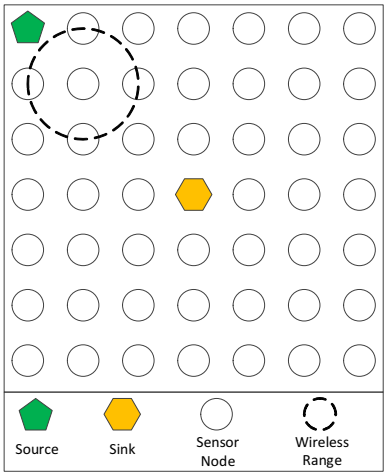


Figure 3.2: Source Corner for a size 7 grid network, with example wireless range of one sensor node

simulator and on real hardware, thus were unsuitable. There was also a simulator designed for experimenting with SLP protocols in WSNs [27], but as the source code could not be found, it could not be evaluated.

The operating system chosen to implement this work using was TinyOS. This was due to the operating system's maturity and stability. Also important is TOSSIM's simulation scalability to a large number of nodes. This is because WSNs will be deployed to monitor animals over a large area with a large number of nodes. Also important is the ability for the same code to be simulated and run on real hardware, as no errors will be introduced porting it from a simulator to hardware.

Previous experience with Contiki and COOJA had shown the simulator was limited, both in terms of the size of the network that could be simulated and the time it would take to simulate that network. However, as TOSSIM lacks the ability to simulate TinyOS' Low Power Listening layer, COOJA will be used to simulate the network when this is necessary.

3.8 Simulation

In this section the simulation environment and protocol configurations that were used to generate the results are described. A modified TOSSIM (version 2.1.2) simulation environment was used for most experiments [95]. TOSSIM is a discrete event simulator capable of accurately modelling sensor nodes and the modes of communications between them. The modifications introduce no backwards incompatible changes. The changes improve the performance of the simulations and add additional features.

3.8.1 Simulation Environment and Network Configuration

A square grid network layout of size $n \times n$ was used in all TOSSIM experiments, where $n \in \{7, 11, 15, 21, 25\}$, i.e., networks with 49, 121, 225, 441, and 625 nodes respectively. In COOJA experiments a network size of $n \in \{7, 9, 11\}$, i.e., networks with 49, 81, and 121 nodes respectively, were used instead due to the slow speed of the simulator. These sizes were chosen because they are practical sizes for deployments. Another consideration is to investigate smaller network sizes because they tend to be harder to provide SLP for, as shown by previous results [78] where the capture ratio tends to be higher for smaller networks. So it is important to validate developed techniques on small networks to see how performance varies.

A single source node generated messages and a single sink node collected messages. These nodes were assigned positions in the SourceCorner configuration shown in Figure 3.2, where the source is in the top left corner and the sink at the centre of the grid. This configuration was chosen as the networks that the techniques developed are also expected to be deployed on relatively sparse networks, in which every node can reach another node via multi-hop communications. The techniques developed are not designed to require grid shaped network and are capable of running on non-grid networks. The rate at which messages from the real source were generated was varied at $P_{src} \in \{2.0, 1.0, 0.5, 0.25\}$ messages per second. In some instances additional experiments were run for $\{8.0, 4.0\}$ messages per second.

Nodes were assigned a unique integer identifier. The identifiers were allocated randomly, so neighbouring nodes were unlikely to have an adjacent identifiers. Nodes were located 4.5 m apart. The node separation distance in TOSSIM was determined experimentally, based on observing the pattern of transmissions in the simulator. This separation distance ensured that messages (i) pass through multiple nodes from source to sink, (ii) can move only one hop at a time, and (iii) will usually only be passed to horizontally or vertically adjacent nodes. However, there is a small probability of diagonal links being present in the network.

3.8.2 TOSSIM Noise and Communication Model

The node neighbourhoods were generating using LinkLayerModel with the parameters in Table 3.1. LinkLayerModel is a tool provided with TOSSIM that extrapolates link strengths from node coordinates based on experimental data. These parameters were chosen based on *Football Field* settings published online [198]. As animal monitoring will occur in outdoor spaces these parameters are more representative of that environment compared to settings discovered when indoors.

Two noise sample files are included with TOSSIM, meyer-heavy and casino-lab.

Name	Value
PATH_LOSS_EXPONENT	4.7
SHADOWING_STD_DEV	3.2
D0	1.0
PL_D0	55.4
NOISE_FLOOR	-105
S	$\begin{bmatrix} 0.9 & -0.7 \\ -0.7 & 1.2 \end{bmatrix}$
WHITE_GAUSSIAN_NOISE	4

Table 3.1: LinkLayerModel Parameters for TOSSIM

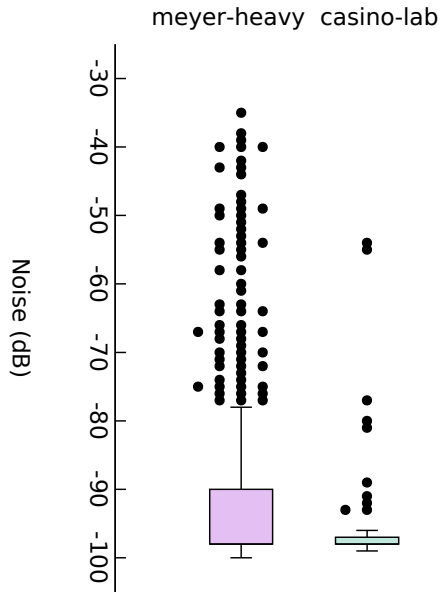


Figure 3.3: Box and whisker plot of noise levels in meyer-heavy and casino-lab

The two noise models have different characteristics that represent noisy (meyer-heavy) and quiet (casino-lab) conditions. As the focus of this work is with a WSN that is deployed over a large outdoors area to monitor assets, noise levels are likely to be low. Therefore, the casino-lab noise model is used to perform the simulations in TOSSIM. The first 2500 lines from casino-lab were used in the simulations, because including the entire file led to slower simulation speeds. The first 2500 lines of both noise models are displayed in the box plots in Figure 3.3.

3.8.3 COOJA Communication Model

When the COOJA simulator is used to generate results, a unit disk graph model (UDGM) represents the links between nodes. As nodes were positioned 4.5 m away from each other the transmit range was set to 4.75 m to ensure nodes could communicate and the interference range was also set to 4.75 m. This setup means that each node has a link between the nodes above, below, left, and right of itself (if they are present). There are no diagonal links present using this communication model. Note that COOJA does not specify a noise model like TOSSIM does.

3.8.4 Safety Period

Following the definition of safety period in Section 3.4, the average time it takes an attacker to capture the source under *Protectionless* flooding is calculated. These

	Network Size	Source Period (seconds/message) 2.0	1.0	0.5	0.25
casino-lab	7×7	17.60	8.81	4.61	2.55
	11×11	33.76	16.85	8.64	4.57
	15×15	50.32	25.14	12.80	6.64
	21×21	75.45	37.73	19.14	9.80
	25×25	92.79	46.26	23.37	11.89

Table 3.2: Safety Period (in seconds) for SourceCorner on TOSSIM.

Network Size	Source Period (seconds/message) 8.0	4.0	2.0	1.0	0.5	0.25
7×7	78.13	39.06	19.53	9.77	5.03	2.70
9×9	110.00	54.85	27.39	13.76	7.07	3.72
11×11	140.63	70.31	35.16	17.58	9.01	4.65

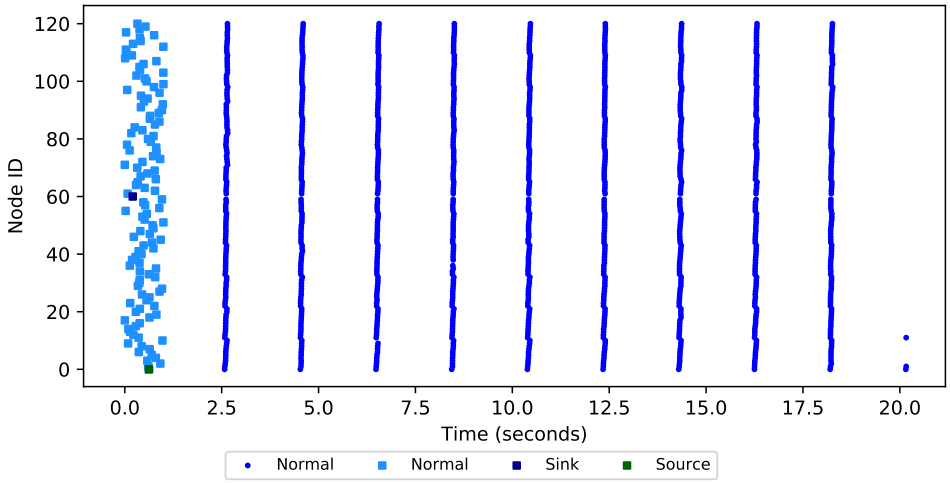
Table 3.3: Safety Period (in seconds) for SourceCorner on COOJA.

values are then doubled due to the safety factor being set to 2, to obtain the safety periods that are shown in Table 3.2 for TOSSIM and in Table 3.3 for COOJA. Two extra source periods (8.0 and 4.0 seconds per message) are run for COOJA as the results for this simulator use these extra values to demonstrate a trend in certain results. The two simulators produced similar safety periods for the same parameters, but COOJA tended to produce slightly higher values.

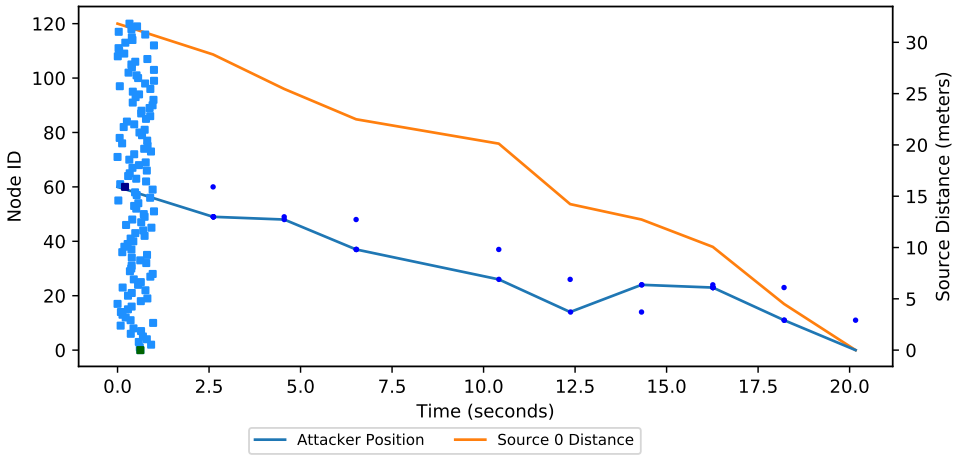
3.8.5 Simulation Experiments

An experiment constituted a single execution of the simulation environment using a specified protocol configuration, network size and safety period. When routing protocols that aim to provide SLP are run, if the attacker does not capture the source within this safety period then it is considered to have failed to capture the source. The simulation terminates when it has been running for a duration equal to the safety period. If the attacker captures the source before the safety period has been reached then it terminates immediately.

To ensure reliable results, a large number of repeats will need to be performed for each parameter combination. Differences in simulation instances with different seeds is expected due to randomness involved with the simulation from the noise and communication models. Sufficient simulation repeats were performed for each combination of parameters to obtain small confidence intervals which can be found in Appendix E. Fewer repeats were performed for COOJA due to the simulator being slower and limited cluster time available.



(a) Messages Sent



(b) Attacker messages received with attacker position and source distance

Figure 3.4: A demonstration of the behaviour of the Protectionless algorithm and how the attacker moves in response to the messages sent.

The results presented are an average of all the values produced from one set of simulation parameters. One result, the capture ratio, is calculated differently. It is defined by the number of simulation instances that led to a capture divided by the total number of simulations executed. A large number of repeats are needed to obtain a reliable capture ratio as the precision depends on the number of repeats. For n repeats the capture ratio will have a resolution of $1/n$.

3.8.6 Example Protectionless

The behaviour of the *Protectionless* Flooding routing protocol and the attacker's response to it is shown in Figure 3.4. Each event occurs on a specific node shown on

the left y-axis. The circles indicate messages being sent, the squares indicate when a node changed type. Nodes do not all become normal nodes at the same time because they boot up at a random time between 0 and 1 second. In these figures the source generated a ⟨normal⟩ message every 2 seconds ($P_{src} = 2$), hence why there is a 2 second gap between waves of ⟨normal⟩ messages. The final wave did not flood the entire network as the simulation terminated when the attacker reached the source.

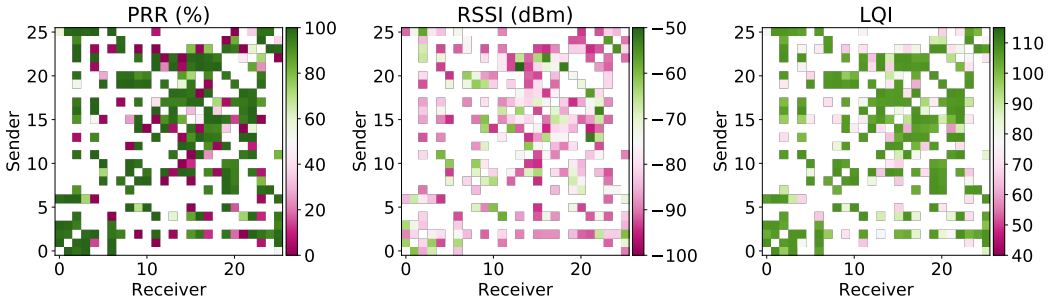
Figure 3.4b shows the attacker's movements and the messages that the attacker eavesdropped. For example, at about 9 seconds the attacker failed to receive a message, indicated by the lack of a blue ⟨normal⟩ message circle. All node change events are included, but an attacker would not be aware of these. The *attacker position* line shows how the attacker moves from node-to-node on its way to the source. The line represents the attacker's movement and where it intersects with message broadcasts shows which messages it responded to. The *source distance* line shows how the distance to the source changes over time. This graph makes it clear what information is available to a local attacker present in the network during its attempt to find the source. The attacker only has a small visibility, so few of the total number of messages sent will ever be eavesdropped by it.

As these graphs provide a useful insight into the behaviour of the algorithm and the attacker's behaviour, they will be used to help analyse the protocols developed in future chapters. Future protocols will include different types of nodes and messages, which will be included in the legend in the messages sent graph.

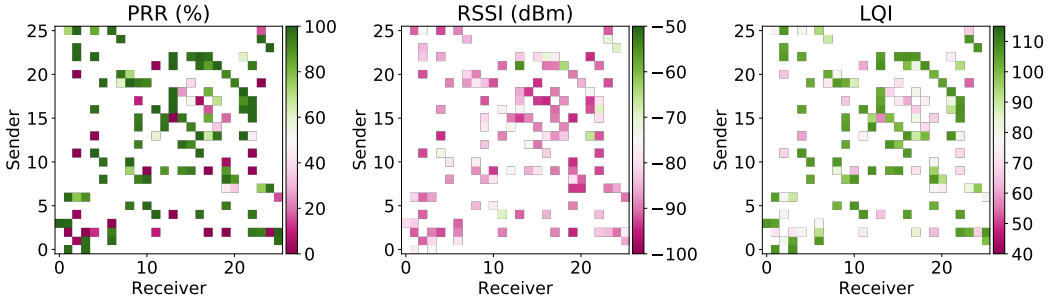
3.9 Testbed

As wireless sensor networks have matured as a technology, testing techniques solely using simulations have become insufficient. Therefore it is important to test developed techniques on *testbeds* which are large deployments of real hardware. Testbeds have a variety of features and vary in the hardware they offer, the metrics that can be gathered about the software, what sensor network OSes are supported, and how jobs are submitted to those testbeds. A number of testbeds were identified as candidates to run experiments on. This section will detail the testbeds identified and explain why FlockLab was selected to run experiments on.

Indriya [43] is a testbed whose infrastructure is designed to support TinyOS. However, Indriya has no support for automation, making deploying jobs a tedious manual task. TWIST [64] is another alternative. However, the testbed was unavailable for use due to construction work on the building the testbed was based in. Both FlockLab [109] and FIT/IoT-LAB [2] supported automatic submission of jobs and automatic retrieval of results. FlockLab has 27 TelosB nodes available across



(a) Broadcast power 31



(b) Broadcast power 7

Figure 3.5: Link quality results for FlockLab.

a single floor. However, queued jobs cannot exceed 1 hour of queued testbed time. FIT/IoT-LAB has a number of different sites with a large number of nodes present. The only site compatible with TinyOS and capable of running data collection scripts was Euratech. The Euratech site has over 200 custom MSP430 nodes.

Both FlockLab and FIT/IoT-LAB were selected to be used for experiments. However, both networks had a very dense topology when using the default (maximum) transmit power of the nodes. This transmit power was decreased to power level 7 to decrease the topology density. However, even with this transmit power, FIT/IoT-LAB had a very small diameter. So for testbed experiments only FlockLab was used to test the developed algorithms under real world conditions.

To understand the results from the testbed the environment in which the testbed exists must first be understood. Therefore, experiments were performed to obtain the noise floor (when no node is broadcasting) and other experiments performed to explore the quality of the wireless links between nodes. All experiments were performed on the default CC2420 channel 26. As SLP techniques are expected to be applied in large and sparse networks, the network degree needed to be reduced. The CC2420 datasheet specifies the power levels that correspond to the transmit power [177, sec 28] as shown in Table 3.5. By decreasing the power level the wireless range of the sensor nodes decrease. With a lower range, fewer nodes will be connected to one another, increasing the diameter of the network.

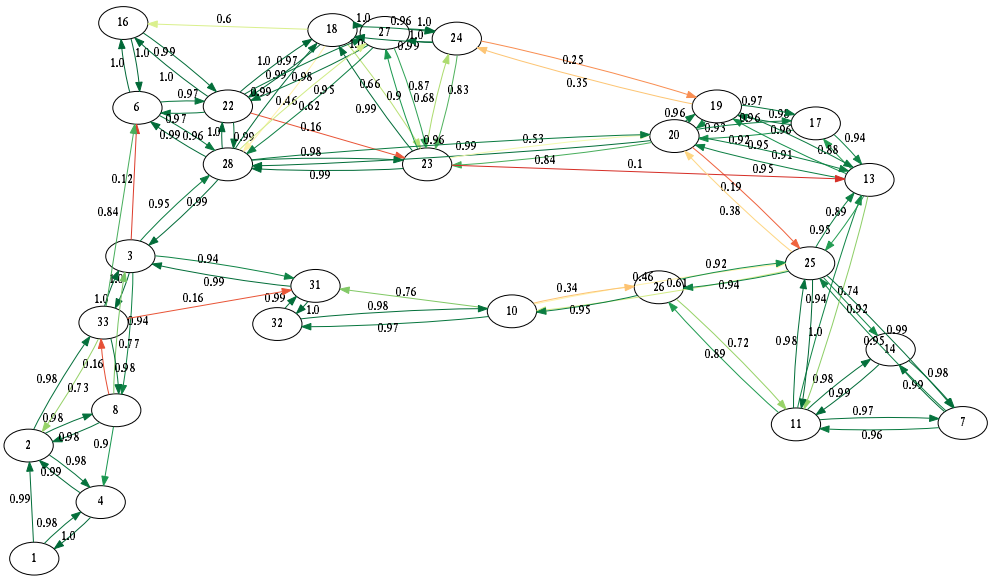


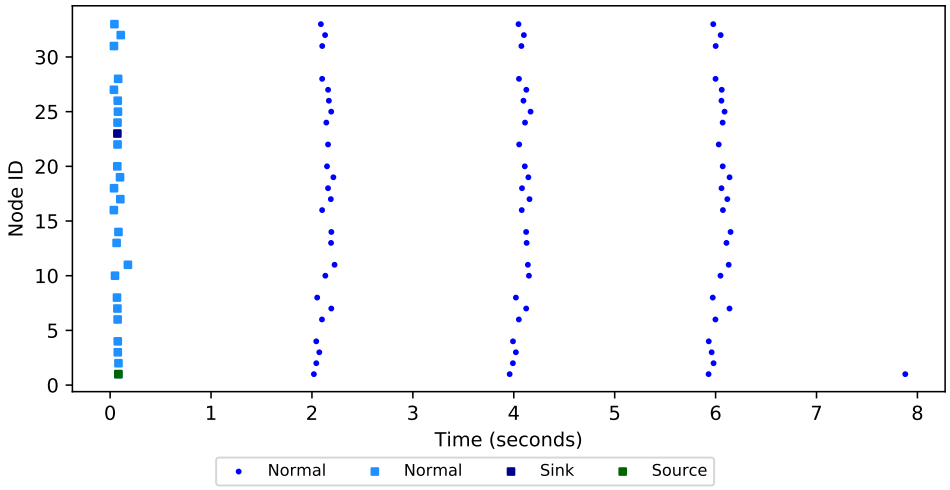
Figure 3.6: The probability of a message being delivered along the links between nodes on FlockLab with a broadcast power of 7

Source Period	0.5	1.0	2.0	8.0
Safety Period	4.44	8.82	17.02	76.18

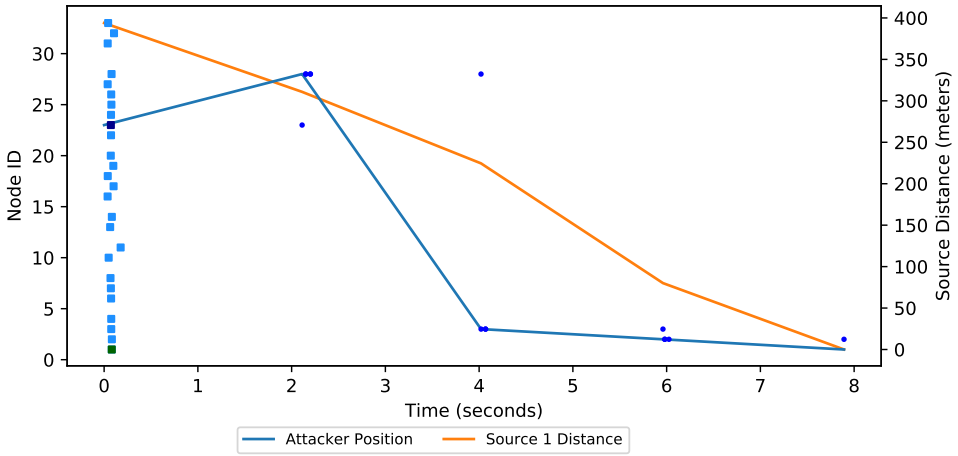
Table 3.4: Safety Period (in seconds) for FlockLab.

The graphs in Figure 3.5 show the percentage of messages that a node received directly from another node (PRR), the received signal strength indicator (RSSI), and the link quality indicator (LQI). Higher transmit powers led to a much denser network with a smaller diameter and lower transmit powers produced a sparser network. The topology when the transmit power was set to 7 is shown in Figure 3.6. This provides a multi-hop topology for the algorithms to be tested on. However, it is still small compared to the networks SLP algorithms are expected to be used on.

An analysis of an instance the attacker's behaviour in Figure 3.7 on Flocklab for *Protectionless* flooding looks very similar to *Protectionless* on TOSSIM. The attacker makes a step towards the source for each message it receives. The attacker does not always move to nodes with lower identifiers because these nodes are on its route to the source. Only four moves are needed for the attacker to capture the source and no moves away from the source are made.



(a) Messages Sent



(b) Attacker messages received with attacker position and source distance

Figure 3.7: A demonstration of the behaviour of the Protectionless algorithm and how the attacker moves in response to the messages sent.

Power Level	**Output Power** (dBm)	**Current Consumption** (mA)
31	0	17.4
27	−1	16.5
23	−3	15.2
19	−5	13.9
15	−7	12.5
11	−10	11.2
7	−15	9.9
3	−25	8.5

Table 3.5: CC2420 Power levels [177, sec 28]

Chapter 4

Analysing the Structure of Source Location Privacy Routing Protocols

There has been much work on providing SLP [40, 157] with many new techniques having been developed, and the performance typically being evaluated through large-scale simulations. Several works [6, 104, 123, 128, 141, 149, 195] have developed models to analyse the privacy provided by their technique or protocol. Many of which tend to be for SLP techniques that provide privacy against an attacker with global visibility. There are two issues here, the first is that the modelling performed is for a specific SLP protocol meaning that its results are not useful in analysing other protocols. The second is that analyses that are more general are for SLP techniques that defend against an attacker with global visibility of the network. This means that there is a lack of analysis of arbitrary routing protocols against local attackers who are present in the network.

To resolve this problem, in this chapter a novel approach based on entropy and divergence is developed to analyse the properties of routing protocols. As this thesis focuses on a distributed eavesdropper attacker that is present in the network, the analysis will aim to quantify how much information is leaked to this kind of attacker. The attacker's position is modelled as a stochastic process (clear data time-series) and its movement is modelled by a Markov chain which is obtained from another Markov chain that specifies the behaviour of a routing protocol. Then, similar to privacy-preserving data mining, an approach for perturbing the clear data time-series to obtain a noisy time-series is developed. However, unlike in privacy-preserving data mining where there is trade-off between information loss and privacy loss [16], no such trade-off is required here. This means that information loss can be maximised,

if possible, to minimize privacy loss.

A suitable definition of information loss is used together with entropy and divergence of random variables to determine the properties of an SLP-aware routing protocol. Specifically, a perturbation model is created such that the normal routing protocol, which is not SLP-aware, is transformed into an SLP-aware routing protocol whereby (i) a source can still do convergecast communication, and (ii) the attacker cannot reach the source within a specified time limit (the safety period). In essence, the SLP-aware routing protocol applies the perturbation technique to the clear time series which generates a noisy time series. The perturbation occurs in such a way that the attacker can learn little information about identifying the source from the noisy time series. Such an approach is beneficial as (i) it does not make any assumptions about the nature of the attacks, and (ii) it does not make any assumptions about any particular protocol implementation.

Furthermore, a novel concept called *proper competing paths* is proposed that captures the problem of whether the attacker can be *stalled* when moving towards the source. Proper competing paths are central to the perturbation model proposed in that (i) wherever proper competing paths exist there is an increased entropy at that point, and (ii) noisy time series made up of proper competing paths are more likely to have very small mutual information with the corresponding clear data time series. The analysis approach developed will be applied to the techniques that are presented in this thesis. The following contributions are made in this chapter:

- The design of an SLP-aware problem is formalised as a transformation problem.
- Using entropy and divergence, the requirements necessary to minimise the amount of information leaked by a noisy time series is derived.
- The concept of proper competing paths is developed, which underpin the perturbation model proposed.
- Two heuristics are proposed to (i) compute the set of proper perturbation paths, and to (ii) transform a normal routing protocol into an SLP-aware routing protocol.

The remainder of this chapter proceeds as follows: Section 4.1 contains a description of the network, privacy and attacker models. Section 4.2 outlines the problem statement and Section 4.3 details the analysis used to guide SLP protocol development. The perturbation model is described in Section 4.4 and an example case study is presented in Section 4.5. In Section 4.6 some of the issues raised with the model is discussed and finally the chapter is concluded in Section 4.7.

4.1 Additions to Models

In this section, a number of extra concepts based on those introduced in Chapter 3 are presented that are required specifically for this chapter.

4.1.1 System Model

A *path* is a finite sequence $\langle n^1 \cdot n^2 \cdots n^j \rangle$ of transitions, where each transition is an ordered pair of nodes $n = (n_1, n_2) \in V \times V$. For a path to be valid, it must be the case that $n_2^i = n_1^{i+1}$. A *source-converging* path is a path that ends at the source, i.e., $n_2^j = s$. In this chapter, unless specified otherwise, a path means a source-converging path. A finite path can be converted into an infinite path through the introduction of loops. For example, when a (finite) path terminates at a source, it can be augmented through the infinite repetition of the final node, i.e., self-loop at the source $\langle n^1 \cdot n^2 \cdots (s,s) \cdot (s,s) \cdot (s,s) \cdots \rangle$. For a given (finite) path p, the number of transitions is denoted by $|p|$. The prefix of path p of length l is denoted by $l \uparrow p$ (resp. the suffix of path p of length l by $p \uparrow l$).

The nodes sense the environment, and when a node detects an event of interest (i.e., the presence of an asset) the node broadcasts information about the event to the network and is called a *source* node. This modelling focuses on the case where there is only a *single* source node in the network (i.e., there is only a single asset to protect). The source is denoted by s and it transmits a single message in every source period P_{src} time units. When a node transmits a new message, its neighbours forward the message in the following time unit. This process is repeated until the message is eventually collected at the sink. This implies that all nodes take exactly one time unit to receive, process and forward a new message. The message is routed towards the sink using a multi-hop routing protocol $\mathcal{R}$. The routing protocol $\mathcal{R}$ is modelled as a matrix, with $\mathcal{R}_{ij}$ representing the probability that node j receives a message from node i *first*, after the message is transmitted by the source. This means that a node can only potentially receive a message first from a neighbour that is closer to the source node. The reason to focus on the first new message is that in real world scenarios the message will usually travel along the shortest path from the source to the sink. Nodes will usually receive a message at least twice, once when it is received for the first time and again by the subsequent node that is forwarding it onwards. In this modelling, a node *cannot* receive a message first along a route which is not the shortest path from the source to that node. A routing protocol that provides little protection to the source location is called a *normal* routing protocol $\mathcal{R}_{\mathcal{N}}$.

4.1.2 Attacker Model

Wherever the attacker is located, upon eavesdropping the *first* new message at that location, the attacker moves to the neighbour who relayed the message. Thus, when the attacker hears a new message, it makes a step towards the source. This process can be repeated a number of times until the attacker reaches the source node, whereby it captures the asset.

In this chapter the distributed eavesdropper attacker is memoryless due to the way its movement is modelled as a Markov chain. This means the attacker does not keep track of history information and it may revisit a node that it has previously visited. Thus, the path an attacker takes to capture an asset may contain loops. Also, since only one message is sent in a single time unit and SLP is being provided within a time limit of the safety period P_{safety}, paths must contain at least P'_{safety} transitions. P'_{safety} is the discrete version of P_{safety} which is obtained by rounding P_{safety} up to the next integer. So, if a path ending at the source s has a length that is less than P'_{safety}, it is extended to contain sufficient number of repetitions of (s, s) until the path has a length of P'_{safety}.

4.2 Problem Statement

When an attacker is initially located at the sink and starts eavesdropping messages along the route between the source and the sink, an important problem is to determine the (maximum) probability that the attacker will be able to reach the source and capture the asset within a specified maximum time bound. The attacker is assumed to use the routing protocol to achieve its objective of reaching the source node. This means that the attacker will not randomly choose moves to take that are not possible as part of the routing protocol. Formally, the problem specification is as follows. Given:

- A network $G = (V, E)$,
- A distributed eavesdropping attacker $\mathbb{A}$ that is initially located at the sink q,
- A single source location $s \in V$,
- A safety period P_{safety},
- A maximum capture threshold δ that determines the SLP level required, and
- A normal routing protocol $\mathcal{R}_{\mathcal{N}}$,

The objective is to transform $\mathcal{R}_{\mathcal{N}}$ into $\mathcal{R}_{\mathcal{S}}$ such that:

- There exists a path from the source s to the sink q using $\mathcal{R}_{\mathcal{S}}$.
- $\mathbb{A}$ reaches s with probability of at most δ within P_{safety} using $\mathcal{R}_{\mathcal{S}}$.

A routing matrix $\mathcal{R}_{\mathcal{S}}$ is called a P_{safety}-δ-SLP routing protocol (or simply an SLP-aware routing protocol) if it prevents the attacker from finding the source within P_{safety} moves with probability δ. Specifically, the objective is to understand the steps required to transform a normal (non SLP-aware) routing protocol into an SLP-aware routing protocol, i.e., to determine the properties that underpin a P_{safety}-δ-SLP routing $\mathcal{R}_{\mathcal{S}}$. Observe that $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ do not need to specify routing protocols with similar transitions, with the exception that a source still needs to be able to send messages to the sink.

One way towards solving this problem is to first develop a protocol and then perform a performance analysis of the protocol to determine its efficiency, for example [6]. Such an analysis identifies the level of SLP the algorithm is capable of providing, and also allows the protocol under analysis to be refined based on the results of the analysis. However, while this technique is effective in demonstrating the performance of a specific technique, it is not suitable in investigating what an optimal technique looks like.

Given that an attacker takes a step along a single hop within a given time unit, its movement is modelled as a time series. This allows the SLP problem to be analysed from the perspective of privacy protection of time-series data. By structuring the analysis in this way, routing protocols can be abstractly considered in order to determine the necessary properties to provide SLP. This chapter will thus consider the problem of quantifying the protection of time-series data (the routing protocol) which has been perturbed by some arbitrary perturbation model. As the perturbation model is related to the transformation of $\mathcal{R}_{\mathcal{N}}$ into $\mathcal{R}_{\mathcal{S}}$, this then provides the ability to determine the development of a SLP routing protocol that potentially minimises privacy loss, i.e., a P_{safety}-0-SLP routing protocol. The remainder of this chapter will focus on the case of $\delta = 0$.

4.3 Problem Analysis

This section will now analyse source location privacy routing protocols using entropy and divergence.

4.3.1 Definitions

The notations and definitions used in the rest of the chapter will now be described before the analysis of the routing matrix transformation required is subsequently

presented.

- The discrete time domain is denoted by $\mathcal{T}$.
- $\mathcal{R}$ is a $|V| \times |V|$ routing matrix, where $\mathcal{R}_{ij}$ represents the probability node j receives a message from i first. There are two variants, one for normal protectionless routing $\mathcal{R}_{\mathcal{N}}$ and another for SLP-aware routing $\mathcal{R}_{\mathcal{S}}$.
- The matrix that specifies the transitions an attacker could take is the transpose of the routing matrix with transitions (n, n) added with probability 1 for each node n when there are no edges that leave n in $\mathcal{R}$. There is an attacker transition matrix for both protectionless routing ($\mathcal{R}^{\mathbb{A}}_{\mathcal{N}}$) and SLP-aware routing ($\mathcal{R}^{\mathbb{A}}_{\mathcal{S}}$), the $\mathcal{X}$ indicates that this could be calculated for either $\mathcal{R}_{\mathcal{S}}$ or $\mathcal{R}_{\mathcal{N}}$.

$$\left[\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right]_{ij} = \begin{cases} 1 & \text{if } i = j \wedge 0 = \sum_{k \in V} \left[\mathcal{R}^{\top}_{\mathcal{X}}\right]_{ik} \\ \left[\mathcal{R}^{\top}_{\mathcal{X}}\right]_{ij} & \text{otherwise} \end{cases} \tag{4.1}$$

- The set of all possible transitions an attacker could take is are a set of ordered pairs of nodes $\Gamma \subseteq V \times V$. The set of possible transitions that an attacker could take in $\mathcal{R}^{\mathbb{A}}_{\mathcal{N}}$ is $\Gamma_{\mathcal{N}} = \{\, (i,j) \mid (i,j) \in \Gamma \wedge \left[\mathcal{R}^{\mathbb{A}}_{\mathcal{N}}\right]_{ij} > 0 \,\}$ and the set of possible transitions in $\mathcal{R}^{\mathbb{A}}_{\mathcal{S}}$ is $\Gamma_{\mathcal{S}} = \{\, (i,j) \mid (i,j) \in \Gamma \wedge \left[\mathcal{R}^{\mathbb{A}}_{\mathcal{S}}\right]_{ij} > 0 \,\}$.
- $\mathcal{N}_\lambda$ is a random variable of attacker transitions $\Gamma_{\mathcal{N}}$, that occur at time λ, under a protectionless routing protocol $\mathcal{R}_{\mathcal{N}}$.
- $\mathcal{S}_\mu$ is a random variable of attacker transitions $\Gamma_{\mathcal{S}}$, that occur at time μ, under an SLP routing protocol $\mathcal{R}_{\mathcal{S}}$.
- The trace of clear time-series data of an attacker movement under a protectionless routing $\mathcal{R}_{\mathcal{N}}$ is a stochastic process $\mathbb{A}_{\mathcal{N}} = \{\mathbb{A}_{\mathcal{N}i}\}_{i \in \mathcal{T}}$, where the $\mathbb{A}_{\mathcal{N}i}$'s form a sequence of random variables of attacker positions in the network ($\mathbb{A}_{\mathcal{N}i} \in V$).
- Thus, the trace of noisy time-series data generated by an SLP routing protocol $\mathcal{R}_{\mathcal{S}}$ is a stochastic process $\mathbb{A}_{\mathcal{S}} = \{\mathbb{A}_{\mathcal{S}i}\}_{i \in \mathcal{T}}$, where the $\mathbb{A}_{\mathcal{S}i}$'s form a sequence of random variables of attacker positions in the network ($\mathbb{A}_{\mathcal{S}i} \in V$).
- There is a safety period P_{safety} which is the number of time units it takes the attacker to reach a source (on average) from the time it first receives a normal message under a protectionless routing protocol $\mathcal{R}_{\mathcal{N}}$.

Note that these routing matrices are different to those in [6] which contain the probability that the routing algorithm chooses the next node. The routing matrices in this chapter instead contain the probability a message is received from a node.

Thus, in this setting, the objective can be recast as follows: develop a transformation ψ that transforms $\mathcal{R}_{\mathcal{N}}$ into $\mathcal{R}_{\mathcal{S}}$ such that, using a measure of *dissimilarity*, the dissimilarity between $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ captures the SLP level imparted by ψ, which implements the SLP-aware protocol.

4.3.2 Analysis

In this section, the characteristics of a routing protocol that can provide a high level of SLP are identified. To do this, a measure of privacy is needed to evaluate the level of SLP enhancement provided by a given solution. There are several potential definitions for privacy metrics and [16] provides a survey of some of these metrics. Initially, the entropy of a single random variable is focused on $H(\mathcal{X}_\lambda)$ and then the divergence between two random variables $\mathcal{N}_\lambda$ and $\mathcal{S}_\mu$ is used to understand how the two routing protocols differ. Note that $\mathcal{X}$ can be replaced with $\mathcal{N}$ or $\mathcal{S}$ when calculating entropy as it only deal with calculating information about one routing protocol.

The entropy of a random variable $H(\mathcal{X}_\lambda)$ indicates the uncertainty of taking transitions at time λ. In order to calculate the entropy the starting location q of the attacker needs to be specified. This means that the entropy at λ will be different depending on where the attacker starts.

$$H(\mathcal{X}_\lambda \mid \mathbb{A}_{\mathcal{X}0} = q) = -\sum_{n \in \Gamma} \Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) \log_2 \Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) \quad (4.2)$$

In order to calculate the entropy, the probability function is now defined. As the attacker movement is represented as a Markov chain, the probabilities will be defined based on Markov chain definitions. The probability that the attacker starting at node i reaches j in exactly λ-steps is $\Pr\left(\mathbb{A}_{\mathcal{X}n+\lambda} = j \mid \mathbb{A}_{\mathcal{X}n} = i\right)$.

$$\Pr\left(\mathbb{A}_{\mathcal{X}t+\lambda} = j \mid \mathbb{A}_{\mathcal{X}t} = i\right) = \left[\left(\mathcal{R}_{\mathcal{X}}^{\mathbb{A}}\right)^{\lambda}\right]_{ij} \quad (4.3)$$

The attacker is assumed to start at the sink, but this chapter will also look at the attacker starting at different locations. Starting at locations with different probabilities is not investigated, this chapter assumes that there is one starting location with a probability of 1.

$$\Pr\left(\mathbb{A}_{\mathcal{X}0} = q\right) = 1 \quad (4.4)$$

The probability that an attacker takes a transition $n = (n_1, n_2)$ at time λ when its starting location is q can be calculated as shown in Equation 4.5. The intuition is to

calculate the probability the attacker reaches node n_1 at time $\lambda - 1$ and then takes the transition n at λ.

$$\begin{aligned}
&\Pr\left(\mathcal{X}_{t+\lambda} = n \mid \mathbb{A}_{\mathcal{X}t} = q\right) \\
&= \Pr\left(\mathbb{A}_{\mathcal{X}t+\lambda} = n_2 \mid \mathbb{A}_{\mathcal{X}t+\lambda-1} = n_1\right) \Pr\left(\mathbb{A}_{\mathcal{X}t+\lambda-1} = n_1 \mid \mathbb{A}_{\mathcal{X}t} = q\right) \\
&= \left[\left(\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right)\right]_{n_1 n_2} \left[\left(\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right)^{\lambda-1}\right]_{q n_1}
\end{aligned} \tag{4.5}$$

To measure how much $\mathcal{R}_{\mathcal{S}}$ diverges from $\mathcal{R}_{\mathcal{N}}$ a divergence measure can be used. Measures such as the Kullback–Leibler (KL) divergence [91] cannot be used as $\forall n \in \Gamma : \Pr\left(\mathcal{S}_\mu = n\right) = 0 \implies \Pr\left(\mathcal{N}_\lambda = n\right) = 0$ does not hold in all cases which is required to use KL divergence. Instead an alternative measure such as the Jensen–Shannon (JS) divergence [110] can be used as it does not require this property. The definition for JS divergence is shown in Equation 4.6 with $\mathcal{N}_\lambda$ and $\mathcal{S}_\mu$ weighted equally.

$$\begin{aligned}
&JSD(\mathcal{N}_\lambda \parallel \mathcal{S}_\mu) = H\left(\frac{\mathcal{N}_\lambda + \mathcal{S}_\mu}{2}\right) - \left(\frac{1}{2}H(\mathcal{N}_\lambda) + \frac{1}{2}H(\mathcal{S}_\mu)\right) \\
&= -\sum_{n\in\Gamma} L\left(\frac{\Pr\left(\mathcal{N}_\lambda = n\right) + \Pr\left(\mathcal{S}_\lambda = n\right)}{2}\right) - \left(-\frac{1}{2}\sum_{n\in\Gamma} L\left(\Pr\left(\mathcal{N}_\lambda = n\right)\right) - \frac{1}{2}L\left(\Pr\left(\mathcal{S}_\lambda = n\right)\right)\right) \\
&= \sum_{n\in\Gamma} \frac{L\left(\Pr\left(\mathcal{N}_\lambda = n\right)\right)}{2} + \frac{L\left(\Pr\left(\mathcal{S}_\lambda = n\right)\right)}{2} - L\left(\frac{\Pr\left(\mathcal{N}_\lambda = n\right) + \Pr\left(\mathcal{S}_\lambda = n\right)}{2}\right)
\end{aligned} \tag{4.6}$$

Where: $$L(x) = x \log_2 x$$

Using the JS divergence indicates how effective the transformation ψ from $\mathcal{R}_{\mathcal{N}}$ to $\mathcal{R}_{\mathcal{S}}$ is at specific points in time. As the log base used to calculate entropy is 2 and the divergence is being calculated for two probability distributions, the divergence is bounded as follows $0 \leq JSD(\mathcal{N}_\lambda \parallel \mathcal{S}_\mu) \leq \log_2(2)$. This means that the upper bound of the divergence is 1. A higher divergence means that there are more differences between the two routing protocols. Ideally the JS divergence would equal 1 when $\lambda = \mu$ for a sufficient number of transitions, indicating the two have fully diverged at the same time instance.

Expected Capture Time

A useful application of Markov chains is the ability to calculate the expected hitting time. This translates well to the SLP problem as it is useful to know the expected capture time of a routing protocol. Using the hitting probability (h_{ij}) and expected hitting time of a Markov Chain, the expected capture time t when the attacker starts

at i and wants to end up at j can be calculated as $E\left[\mathbb{A}_{\mathcal{X}t} = j \mid \mathbb{A}_{\mathcal{X}0} = i\right]$. These equations can be calculated for $\mathbb{A}_{\mathcal{N}}$ and $\mathbb{A}_{\mathcal{S}}$.

$$h_{ij} = \begin{cases} 1 & \text{if } i = j \\ \sum\limits_{k \in V \wedge j \neq i} \left[\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right]_{ik} h_{kj} & \text{otherwise} \end{cases} \tag{4.7}$$

$$E\left[\mathbb{A}_{\mathcal{X}t} = j \mid \mathbb{A}_{\mathcal{X}0} = i\right] = m_{ij} = \begin{cases} \infty & \text{if } h_{ij} < 1 \\ 0 & \text{if } i = j \\ 1 + \sum\limits_{k \in V \wedge j \neq i} \left[\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right]_{ik} m_{kj} & \text{otherwise} \end{cases} \tag{4.8}$$

The δ component of a P_{safety}-δ-SLP routing protocol can also be calculated. This is the capture probability when the attacker starts at i and wants to reach j within P'_{safety} hops.

$$\delta_{ij}^{P'_{safety}} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \wedge P'_{safety} = 0 \\ \sum\limits_{k \in V \wedge j \neq i} \left[\mathcal{R}^{\mathbb{A}}_{\mathcal{X}}\right]_{ik} \delta_{kj}^{P'_{safety}-1} & \text{otherwise} \end{cases} \tag{4.9}$$

Information Loss from Privacy Preserving Data Mining

While the Jensen-Shannon divergence works well for calculating the divergence at specific time points, it doesn't provide an overall summary. So a notion of dissimilarity between $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ is needed such that the dissimilarity level is indicative of the SLP level provided by $\mathcal{R}_{\mathcal{S}}$. It was shown in [3] that the notion of information loss varies inversely with privacy loss (see Figure 4.1 for examples representative of different perturbation models), i.e., the higher the information that is lost or the more perturbed the clear data time-series is, the less privacy is lost.

To this end, the following definition of information loss in Equation 4.10, which is used in privacy-preserving data mining [15, 16] is adapted to suit the SLP problem (see Equation 4.11). Here, $D_{\mathcal{N}}$ and $D_{\mathcal{S}}$ represent the clear and noisy domains respectively, and $f_D(i)$ represents the frequency of the data item i in domain D.

$$IL(D_{\mathcal{N}}, D_{\mathcal{S}}) = \frac{\sum_{i=1}^{n} |f_{D_{\mathcal{N}}}(i) - f_{D_{\mathcal{S}}}(i)|}{\sum_{i=1}^{n} f_{D_{\mathcal{N}}}(i)} \tag{4.10}$$

Since transitions do not contribute to information gain to the attacker after the safety period has elapsed, the information loss definition is adapted to only

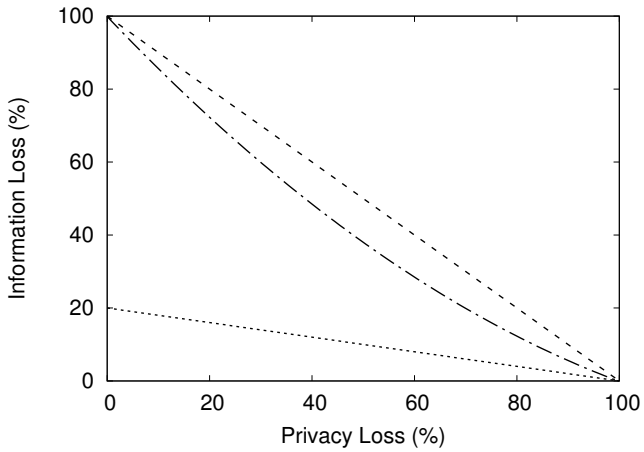


Figure 4.1: Information Loss vs Privacy Loss representative of [3, Fig. 8] depending on the perturbation model used.

include transitions in $\mathcal{R}_{\mathcal{S}}$ that occur before the safety period expires.

$$IL(D_{\mathcal{N}}, D_{\mathcal{S}}) = \frac{\sum_{i=1}^{n} |f_{D_{\mathcal{N}}}(i) - f_{D_{\mathcal{S}}}(i^{P'_{safety}})|}{\sum_{i=1}^{n} f_{D_{\mathcal{N}}}(i)} \tag{4.11}$$

Where $\mathcal{F}_{\mathcal{D}_{\mathcal{N}}}(i)$ and $\mathcal{F}_{\mathcal{D}_{\mathcal{S}}}(i^{P'_{safety}})$ are defined as:

$$\mathcal{F}_{\mathcal{D}_{\mathcal{N}}}(i) = \begin{cases} 1 & \text{if transition } i \text{ is used in } \mathcal{R}_{\mathcal{N}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

$$\mathcal{F}_{\mathcal{D}_{\mathcal{S}}}(i^{P'_{safety}}) = \begin{cases} 1 & \text{if } i \text{ is not taken within } P'_{safety} \text{ steps in } \mathcal{R}_{\mathcal{S}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

Equation 4.10 states that the more dissimilar the set of transitions taken within P_{safety} time units are, the greater is the information loss, hence the lower the privacy loss. If $IL(D_{\mathcal{N}}, D_{\mathcal{S}}) = 1$ (i.e., is maximum), then it implies that $D_{\mathcal{N}} \cap D_{\mathcal{S}} = \emptyset$. In other words, to minimise privacy loss, $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ cannot share any transition. More specifically, it means that, though $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ can share transitions, an attacker cannot take some transition in $\mathcal{R}_{\mathcal{N}}$ under $\mathcal{R}_{\mathcal{S}}$ within P'_{safety} time units. This then means that $\mathcal{R}_{\mathcal{N}}$ has to be transformed in such a way that *for a certain duration*, for any transition (i, j) unique in $\mathcal{R}_{\mathcal{S}}$, an attacker at location j needs to receive a message from i first.

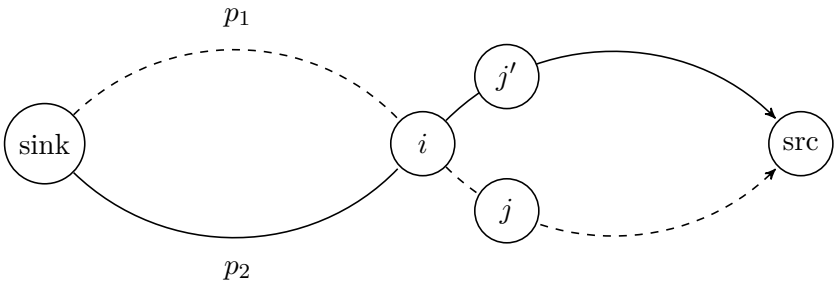


Figure 4.2: Demonstration of two paths p_1 and p_2 the attacker could take that compete at node i.

4.4 Perturbation Model: Proper Competing Paths

To understand the transformation of $\mathcal{R}_\mathcal{N}$ into $\mathcal{R}_\mathcal{S}$, i.e., to understand how $\mathcal{R}_\mathcal{N}$ can perturbed into $\mathcal{R}_\mathcal{S}$, the concept of *competing paths* will now be introduced. A visualisation of Definition 1 is shown in Figure 4.2.

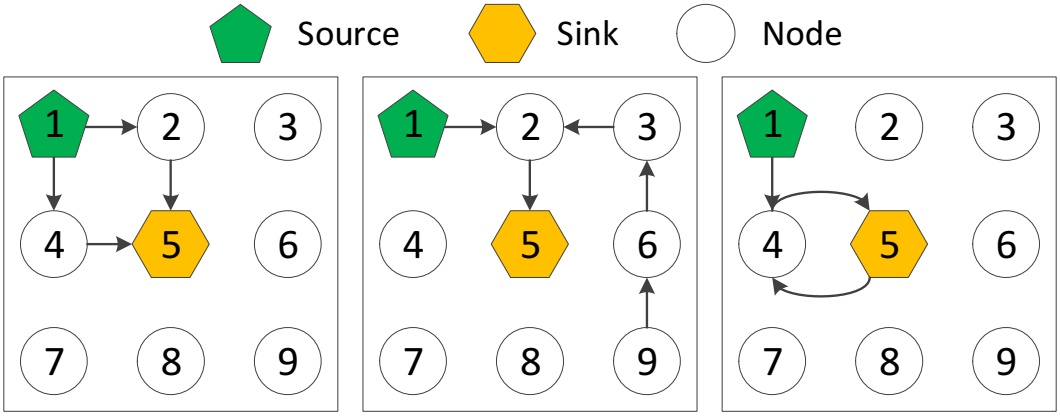
Definition 1 (Competing Paths) *Given a network $G = (V, E)$ and a protection-less routing protocol $\mathcal{R}_\mathcal{N}$, two distinct paths p_1 and p_2 under $\mathcal{R}_\mathcal{N}$ compete at a node $n \in V$ iff the following are satisfied:*

- *p_1 and p_2 are source-converging paths.*
- $\exists (i,j), (i,j') \in E : (i,j) \in p_1 \wedge (i,j') \in p_2 \wedge i = n$
- $\forall j, j' \in V : j \neq j' \implies [\mathcal{R}_\mathcal{N}]_{jn} > 0 \wedge [\mathcal{R}_\mathcal{N}]_{j'n} \geq 0$

Definition 2 (Junction Node) *A node n is a junction node if multiple competing paths compete at n.*

The idea of competing paths is that if one path is part of the clear data time-series, then the other can be used in the noisy data time-series. Specifically, it means that if the attacker is more likely to follow a given path p_1 under $\mathcal{R}_\mathcal{N}$, then the attacker can be made to follow path p_2 under $\mathcal{R}_\mathcal{S}$. In this definition the node n is called a *junction node*, p_1 is called a normal path, and p_2 a perturbed path. The example in Figure 4.3a can be considered. Since $[\mathcal{R}_\mathcal{N}]_{2,5} = 0.5$ and $[\mathcal{R}_\mathcal{N}]_{4,5} = 0.5$, then paths $\langle (5,2) \cdot (2,1) \rangle$ and $\langle (5,4) \cdot (4,1) \rangle$ compete at node 5. Observe that the notion of competing paths increases the entropy at the node they are competing at.

However, not all competing paths are capable of preventing the attacker from reaching the source within the required safety period. For example, in Figure 4.3a the two paths both compete at node 5, but neither prevent the attacker from reaching



(a) Junction Node at 5 (b) Junction Node at 2 (c) Junction Node at 4 and 5

Figure 4.3: Examples of competing paths and their junction nodes

the source. The notion of competing paths is thus strengthened to that of *proper competing paths.*

Corollary 1 *All source-converging paths compete at the sink.*

Definition 3 (Proper Competing Paths) *Given a network $G = (V, E)$ and a protectionless routing protocol $\mathcal{R}_\mathcal{N}$, two distinct paths p_1 and p_2 under $\mathcal{R}_\mathcal{N}$ compete properly at a node $n \in V$ iff the following are satisfied:*

- *p_1 and p_2 are source-converging paths.*
- $\exists (i,j),(i,j') \in E : (i,j) \in p_1 \wedge (i,j') \in p_2 \wedge i = n$
- $\forall j, j' \in V : j \neq j' \implies [\mathcal{R}_\mathcal{N}]_{jn} > 0 \wedge [\mathcal{R}_\mathcal{N}]_{j'n} = 0$

Definition 4 (Proper Junction Node) *A node n is a proper junction node if multiple proper competing paths properly compete at n.*

Here, for two proper competing paths, the attacker cannot receive the message first along one of these paths. Thus, path p_1 should be perturbed into path p_2 in the noisy data time-series. The intuition is that the attacker, at a proper junction node, has two distinct choices and one of those choices is one it would unlikely have made under normal circumstances. As before, p_1 is called a normal path and p_2 a properly perturbed path.

Lemma 1 *Given a network $G = (V, E)$, a protectionless routing protocol $\mathcal{R}_\mathcal{N}$, an attacker $\mathbb{A}$ that starts at the sink, safety period P_{safety}, and a path p_1 under $\mathcal{R}_\mathcal{N}$ with $|p_1| \leq P'_{safety}$, then there exists a path p_2 with $|p_2| > P'_{safety}$ such that $\exists n \in p_1, p_2$ where p_1 and p_2 properly compete at n.*

Proof. It needs to be proven that along p_1 there exists a node n such that p_1 and p_2 properly compete at n and that $|p_2| > P'_{safety}$.

This is proved by contradiction: First it is assumed that there exists no such path p_2. Consider a node $i \in p_1$ and one of its neighbours j which is further from the source. According to the assumption, $0 < [\mathcal{R}_\mathcal{N}]_{ij} \leq 1$ holds. Since it is assumed that no p_2 exists that can properly compete with p_1, i.e., no proper junction node exists, $\forall j, (i,j) \in E : 0 < [\mathcal{R}_\mathcal{N}]_{ij} \leq 1$ holds. However, $\exists k : (i,k) \in E$ is further from the source than i, then $[\mathcal{R}_\mathcal{N}]_{ki} = 0$, which is a contradiction. Thus, such a p_2 exists and the proper junction node is node j. Path p_2 is also of length $|p_2| > P'_{safety}$, by repeating the transition (i,j) P'_{safety} times. □

The intuition is that p_1 is a path that an attacker may follow under the protectionless protocol $\mathcal{R}_\mathcal{N}$ to capture the asset, while p_2 provides a *diversion* via a path that the attacker will not normally follow. This path should be long enough to adequately delay the attacker. Further, p_2 also captures the fact that p_1 can be perturbed into p_2 at the identified proper junction node. Such a path p_2 will need to be guaranteed to occur under a routing protocol.

Using Figures 4.3a and 4.3c, as an example, there are the following two paths p_1 and p_2. Setting $P_{safety} = 4$ the paths are expanded out to p'_1 and p'_2. Notice that the attacker reaches the asset at node 1 in p_1 within P_{safety} steps but does not in p_2.

$$
\begin{aligned}
p_1 &= \langle (5,4) \cdot (4,1) \rangle \\
p'_1 &= \langle (5,4) \cdot (4,1) \cdot (1,1) \cdot (1,1) \cdot (1,1) \cdot (1,1) \cdot (1,1) \rangle \\
p_2 &= \langle (5,4) \cdot (4,5) \cdot (5,4) \cdot (4,5) \cdot (5,4) \cdot (4,1) \rangle \\
p'_2 &= \langle (5,4) \cdot (4,5) \cdot (5,4) \cdot (4,5) \cdot (5,4) \cdot (4,1) \cdot (1,1) \rangle
\end{aligned}
$$

At this point, a proper junction point needs to be selected. Since the attacker starts at the sink, the sink is the first candidate to be considered as a proper junction point. If the sink is not a proper junction point, normal paths are iteratively selected under $\mathcal{R}_\mathcal{N}$ at random and proper junction points are identified. Once a proper junction node is selected, a properly perturbed path needs to be selected to obtain the routing matrix $\mathcal{R}_\mathcal{S}$, i.e., choosing a set of properly perturbed paths.

Hence, the heuristic in Algorithm 1 is proposed to generate the set of properly perturbed paths at a given proper junction node. The SLP-aware routing matrix $\mathcal{R}_\mathcal{S}$ can then be generated using the perturbed time series as shown in Algorithm 2. Note that the heuristic does not define certain methods (such as CHOOSE) as different definitions will lead to different SLP-aware routing protocols being produced. The set of properly perturbed paths that Algorithm 1 generates is shown below, and the

Algorithm 1 Generating set of properly perturbed paths with a length of at least P'_{safety} hops. The safety factor is ϕ.

function ProperlyPerturbedPaths($\mathcal{R}_\mathcal{N}$, ϕ)
  PPP $\leftarrow \emptyset$
  NormalPaths $\leftarrow$ set of all normal paths from sink q to source s in $\mathcal{R}^{\mathbb{A}}_\mathcal{N}$
  $\mathcal{TT} \leftarrow E\left[\mathbb{A}_{\mathcal{N}t} = s \mid \mathbb{A}_{\mathcal{N}0} = q\right]$ ▷ Expected capture time
  $P'_{safety} \leftarrow \lceil \mathcal{TT} \times \phi \rceil$
  for path $\in$ NormalPaths **do**
    used $\leftarrow \langle\rangle$
    for $(i, j) \in$ path **do**
      if $i =$ source **then**
        break
      ▷ Find an unused node n to perturb p though
      for $n \in V : \left[\mathcal{R}^{\mathbb{A}}_\mathcal{N}\right]_{in} = 0$ **do**
        pp $\leftarrow \{\, p \mid$ generate path p from q to s where
            $(|\text{used}| \uparrow p) = \text{used} \wedge$ ▷ Path starts the same
            $p^{|\text{used}|+1} = (i, n) \wedge$ ▷ Path goes via n
            $(\lceil \mathcal{TT} \rceil \uparrow p) \notin \mathcal{R}_\mathcal{N} \wedge$ ▷ Eliminate paths in $\mathcal{R}_\mathcal{N}$
            $|p| > P'_{safety} \,\}$ ▷ Attacker does not reach s within P'_{safety}
        pp $\leftarrow \{\, \langle (n_2, n_1) \mid n \in \text{reverse}(p) \rangle \mid p \in \text{pp} \,\}$ ▷ Reverse the path
        PPP $\leftarrow$ PPP $\cup$ pp
      used $\leftarrow$ used $\frown \langle (i, j) \rangle$
  return PPP

Choose function picks the fourth path in Algorithm 2 in the following examples.

$$\{\, \langle (1,2) \cdot (2,3) \cdot (3,6) \cdot (6,9) \cdot (9,8) \cdot (8,5) \rangle,$$
$$\langle (1,2) \cdot (2,3) \cdot (3,6) \cdot (6,9) \cdot (9,8) \cdot (8,7) \cdot (7,4) \cdot (4,5) \rangle,$$
$$\langle (1,4) \cdot (4,7) \cdot (7,8) \cdot (8,9) \cdot (9,6) \cdot (6,3) \cdot (3,2) \cdot (2,5) \rangle,$$
$$\langle (1,4) \cdot (4,7) \cdot (7,8) \cdot (8,9) \cdot (9,6) \cdot (6,5) \rangle \,\}$$

Theorem 1 (Privacy Loss) *Given, a network $G = (V, E)$, a source location $s \in V$, a sink location $q \in V$, a distributed eavesdropper attacker $\mathbb{A}$ that is initially located at the sink, a safety factor ϕ, and a normal routing protocol $\mathcal{R}_\mathcal{N}$ with expected time to capture $\mathcal{TT}$, then $\mathcal{R}_\mathcal{S}$ generated with Algorithm 2 (if such a routing matrix exists) results in at least $(\phi - 1)\mathcal{TT} + 2$ steps where $JSD(\mathcal{N}_\lambda \parallel \mathcal{S}_\lambda) = 1$.*

Lemma 2 *At least $(\phi - 1)\mathcal{TT} + 2$ steps completely diverge in $\mathcal{R}_\mathcal{S}$ compared to $\mathcal{R}_\mathcal{N}$.*

Proof. By construction. As the paths generated must be longer than $\phi\mathcal{TT}$ (the safety period), the minimum length of a properly perturbed path must be

Algorithm 2 Generating SLP-aware routing matrix

▷ Generate a set of routing paths from the source s to the sink q for messages to follow. The attacker follows the reverse of this path. The safety factor is ϕ.

function PERTURBNORMALROUTING($\mathcal{R}_{\mathcal{N}}$, ϕ)
  $\mathcal{R}_{\mathcal{S}} \leftarrow \mathcal{R}_{\mathcal{N}}$
  PPP $\leftarrow$ PROPERLYPERTURBEDPATHS($\mathcal{R}_{\mathcal{N}}$, ϕ)
  p $\leftarrow$ CHOOSE(PPP) ▷ Choose one path, fails if no paths available
  for $(i, j) \in p$ **do**
    for $k \in V \setminus \{i\}$ **do**
      $[\mathcal{R}_{\mathcal{S}}]_{kj} \leftarrow 0$ ▷ Prevent j from receiving from any node except i
    $[\mathcal{R}_{\mathcal{S}}]_{ij} \leftarrow 1$ ▷ Ensuring attacker follows chosen properly perturbed path
  ▷ Remove paths from source that no longer terminate at the sink
  $\mathcal{R}_{\mathcal{S}} \leftarrow$ REMOVEUNTERMINATINGPATHS($\mathcal{R}_{\mathcal{S}}$)
  return $\mathcal{R}_{\mathcal{S}}$

$\phi\mathcal{TT} + 1$. The maximum number of overlapping steps the perturbed path can have is $\mathcal{TT} - 1$. So the minimum number of diverged steps that the path must have is $\phi\mathcal{TT} + 1 - (\mathcal{TT} - 1) = (\phi - 1)\mathcal{TT} + 2$. □

Lemma 3 *When a transitions diverges at λ, for all n in Γ:*

$$\Pr(\mathcal{S}_\lambda = n) = 1 \implies \Pr(\mathcal{N}_\lambda = n) = 0 \tag{4.14}$$

$$\Pr(\mathcal{N}_\lambda = n) > 0 \implies \Pr(\mathcal{S}_\lambda = n) = 0 \tag{4.15}$$

These properties lead to $JSD(\mathcal{N}_\lambda \parallel \mathcal{S}_\lambda) = 1$.

Proof. There are three cases that need to be considered:

- Case 1: $\Pr(\mathcal{S}_\lambda = n) = 1$, where it is also the case that $\Pr(\mathcal{N}_\lambda = n) = 0$ due to the implication.

$$\frac{L(\Pr(\mathcal{N}_\lambda = n))}{2}^{0} + \frac{L(\Pr(\mathcal{S}_\lambda = n))}{2}^{0} - L\left(\frac{\Pr(\mathcal{N}_\lambda = n) + \Pr(\mathcal{S}_\lambda = n)}{2}\right)^{0.5} \tag{4.16}$$

  So when n is taken under $\mathcal{R}_{\mathcal{S}}$ the expression inside the sum has the value $-L(0.5) = 0.5$.

- Case 2: $\Pr(\mathcal{N}_\lambda = n) > 0$, where it is also the case that $\Pr(\mathcal{S}_\lambda = n) = 0$ due to

the implication.

$$\begin{aligned}
&\frac{L\left(\Pr\left(\mathcal{N}_\lambda = n\right)\right)}{2} + \frac{L\left(\Pr\left(\mathcal{S}_\lambda = n\right)\right)}{2} - L\left(\frac{\Pr\left(\mathcal{N}_\lambda = n\right) + \Pr\left(\mathcal{S}_\lambda = n\right)}{2}\right) \\
&= \frac{L\left(\Pr\left(\mathcal{N}_\lambda = n\right)\right)}{2} - L\left(\frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\right) \\
&= \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\log_2 \Pr\left(\mathcal{N}_\lambda = n\right) - \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\log_2 \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2} \\
&= \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\left(\log_2 \Pr\left(\mathcal{N}_\lambda = n\right) - \log_2 \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\right) \\
&= \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}\left(\log_2 \Pr\left(\mathcal{N}_\lambda = n\right) - \left(\log_2 \Pr\left(\mathcal{N}_\lambda = n\right) - 1\right)\right) \\
&= \frac{\Pr\left(\mathcal{N}_\lambda = n\right)}{2}
\end{aligned} \tag{4.17}$$

As $1 = \sum_{n\in\Gamma} \Pr\left(\mathcal{N}_\lambda = n\right)$ then $0.5 = \sum_{n\in\Gamma} \frac{\Pr(\mathcal{N}_\lambda = n)}{2}$

- Case 3: $\Pr\left(\mathcal{N}_\lambda = n\right) = 0 \wedge \Pr\left(\mathcal{S}_\lambda = n\right) = 0$

$$\frac{L\left(\Pr\left(\mathcal{N}_\lambda = n\right)\right)}{2} + \frac{L\left(\Pr\left(\mathcal{S}_\lambda = n\right)\right)}{2} - L\left(\frac{\Pr\left(\mathcal{N}_\lambda = n\right) + \Pr\left(\mathcal{S}_\lambda = n\right)}{2}\right) \tag{4.18}$$

This leaves the summation to obtain the Jensen-Shannon divergence as:

$$1 = \sum_{n\in\Gamma} \begin{cases} 0.5 & \text{if } \Pr\left(\mathcal{S}_\lambda = n\right) = 1 \\ \frac{\Pr(\mathcal{N}_\lambda = n)}{2} & \text{if } \Pr\left(\mathcal{N}_\lambda = n\right) > 0 \\ 0 & \text{if } \Pr\left(\mathcal{N}_\lambda = n\right) = 0 \wedge \Pr\left(\mathcal{S}_\lambda = n\right) = 0 \end{cases} \tag{4.19}$$

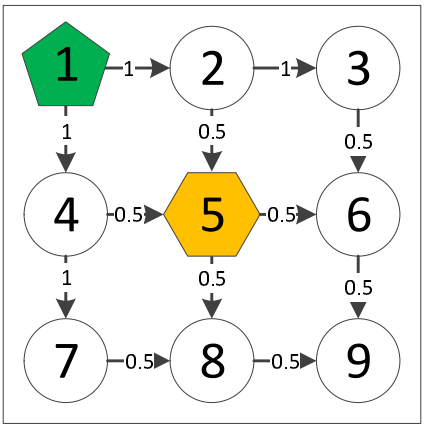
□

4.5 A Case Study

In this section, a simple case study is presented to show the viability of this approach. Using the network shown in Figure 4.4a for $\mathcal{R}_\mathcal{N}$ and Figure 4.5a for $\mathcal{R}_\mathcal{S}$, with the sink at the centre of the network and the source on the border of the network. For example, $[\mathcal{R}_\mathcal{N}]_{1,2} = 1$ means that node 2 will receive a message first from node 1. On the other hand, $[\mathcal{R}_\mathcal{N}]_{2,5} = 0.5$ means there is a 50% chance that node 5 will receive a message first from node 2 (the other possibility is from 4). The set of normal paths is given by $\{\langle(5,2)\cdot(2,1)\rangle, \langle(5,4)\cdot(4,1)\rangle\}$. Here, the safety period P'_{safety} is 4 as

Receiving Nodes

Sending Node	1	2	3	4	5	6	7	8	9
1	0	1	0	1	0	0	0	0	0
2	0	0	1	0	0.5	0	0	0	0
3	0	0	0	0	0	0.5	0	0	0
4	0	0	0	0	0.5	0	1	0	0
5	0	0	0	0	0	0.5	0	0.5	0
6	0	0	0	0	0	0	0	0	0.5
7	0	0	0	0	0	0	0	0.5	0
8	0	0	0	0	0	0	0	0	0.5
9	0	0	0	0	0	0	0	0	0

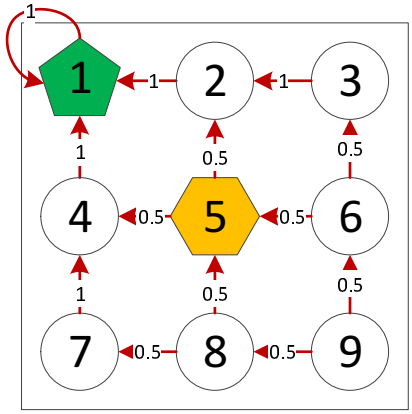


(a) Routing matrix $\mathcal{R}_{\mathcal{N}}$

Sending Node (Moving To)

Recv. Nodes (Moving From)

	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0.5	0	0.5	0	0	0	0	0
6	0	0	0.5	0	0.5	0	0	0	0
7	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0.5	0	0.5	0	0
9	0	0	0	0	0	0.5	0	0.5	0



(b) Attacker movement matrix $\mathcal{R}^{\mathbb{A}}_{\mathcal{N}}$

Figure 4.4: *Protectionless* flooding matrices

the expected capture time is 2 and the safety factor ϕ is 2.

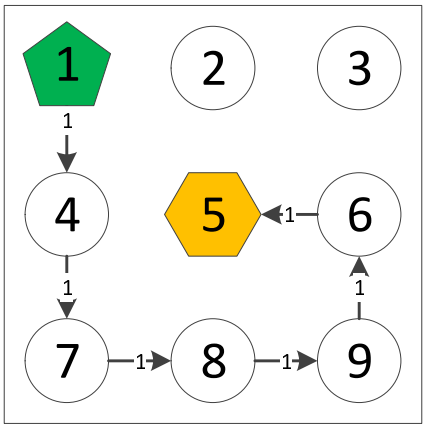
Here, the first proper junction node is node 5, the sink. Any properly perturbed path will start with transition $(5, 6)$ or $(5, 8)$. Applying the heuristic to generate $\mathcal{R}_{\mathcal{S}}$ shows that (i) it is first required to reset the values of node 5 and then set the value of the relevant transition to 1. This is shown in the matrix $\mathcal{R}_{\mathcal{S}}$, where the old value is stricken off and replaced by the new value, shown in bold. For example, it means that $[\mathcal{R}_{\mathcal{S}}]_{6,5}$ needs to be set to 1 to ensure that the attacker moves to node 6 from the sink, rather than move towards either node 2 or 4. This is performed for each transition in one of the properly perturbed paths.

As can be observed, an attacker will now take the following path: $\langle (5, 6) \cdot (6, 9) \cdot (9, 8) \cdot (8, 7) \cdot (7, 4) \cdot (4, 1) \rangle$, meaning that the attacker requires 6 transitions to reach the source, which is more than the safety period. Hence, it means that the attacker cannot catch the source before the safety period expires.

The information loss of $\mathcal{R}_{\mathcal{S}}$ compared to $\mathcal{R}_{\mathcal{N}}$ can be calculated using the

Receiving Nodes

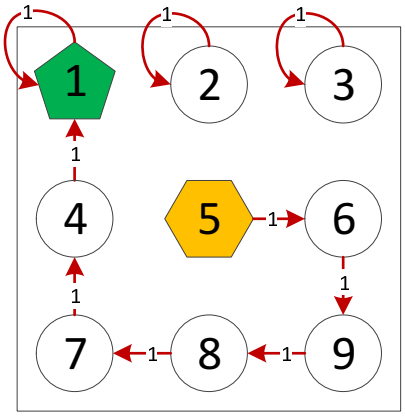
Sending Node	1	2	3	4	5	6	7	8	9
1	0	~~1~~ **0**	0	1	0	0	0	0	0
2	0	0	~~1~~ **0**	0	~~0.5~~ **0**	0	0	0	0
3	0	0	0	0	0	~~0.5~~ **0**	0	0	0
4	0	0	0	0	~~0.5~~ **0**	0	1	0	0
5	0	0	0	0	0	~~0.5~~ **0**	0	~~0.5~~ **0**	0
6	0	0	0	0	~~0~~ **1**	0	0	0	~~0.5~~ **0**
7	0	0	0	0	0	0	0	~~0.5~~ **1**	0
8	0	0	0	0	0	0	0	0	~~0.5~~ **1**
9	0	0	0	0	0	~~0~~ **1**	0	0	0



(a) Routing matrix $\mathcal{R}_{\mathcal{S}}$

Sending Node (Moving To)

Recv. Nodes (Moving From)	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	1
7	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	1	0



(b) Attacker movement matrix $\mathcal{R}_{\mathcal{S}}^{\mathbb{A}}$

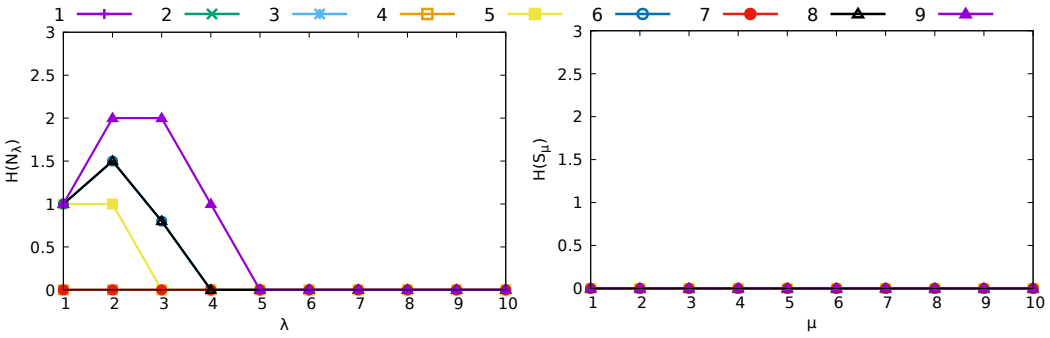
Figure 4.5: Perturbed SLP matrices

equation in Equation 4.10. There are twelve transitions present in the domain of $\Gamma_{\mathcal{N}}$ shown in $D_{\mathcal{N}}$. There are six transitions in the domain of $\Gamma_{\mathcal{S}}$ (shown in $D_{\mathcal{S}}$), of which only four are reachable within the safety period (shown in $D_{\mathcal{S}}^{P'_{safety}}$). Table 4.1 shows which transitions are present in both $D_{\mathcal{N}}$ and $D_{\mathcal{S}}^{P'_{safety}}$. Using this result, (where ✗ is a 1 and ✓ is a 0) the information loss can be calculated in Equation 4.23.

$$D_{\mathcal{N}} = \{(1,2),(1,4),(2,3),(2,5),(3,6),(4,5),\\ (4,7),(5,6),(5,8),(6,9),(7,8),(8,9)\} \tag{4.20}$$

$$D_{\mathcal{S}} = \{(1,4),(4,7),(7,8),(8,9),(9,6),(6,5)\} \tag{4.21}$$

$$D_{\mathcal{S}}^{P'_{safety}} = \{(7,8),(8,9),(9,6),(6,5)\} \tag{4.22}$$



(a) Entropy of $\mathcal{N}_\lambda$ (b) Entropy of $\mathcal{S}_\mu$

Figure 4.6: Entropy at different times based on the attacker's starting position

$D_\mathcal{N}$	$(1,2)$	$(1,4)$	$(2,3)$	$(2,5)$	$(3,6)$	$(4,5)$	$(4,7)$	$(5,6)$	$(5,8)$	$(6,9)$	$(7,8)$	$(8,9)$
$D_\mathcal{S}^{P'_{safety}}$	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓

Table 4.1: Are the transitions from $D_\mathcal{N}$ in $D_\mathcal{S}^{P'_{safety}}$?

$$IL(D_\mathcal{N}, D_\mathcal{S}) = \frac{1+1+1+1+1+1+1+1+1+1+0+0}{12} = \frac{10}{12} = \frac{5}{6} \tag{4.23}$$

These results show that the information loss from the attacker's perspective is 83%, meaning it can only observe 17% of the original routing matrix. By reducing the amount of information the attacker can gain from the routing matrix, it can be prevented from finding the source within the safety period.

4.5.1 Entropy and Divergence

As well as using the measure of information loss from privacy preserving data mining, the entropy can be calculated for these two routing matrices $\mathcal{R}_\mathcal{N}$ and $\mathcal{R}_\mathcal{S}$ individually, and then Jensen–Shannon divergence can be calculated to compare them. Figure 4.6 shows the entropy for the two example routing matrices in Figure 4.4a and Figure 4.5a. Because there is only one path between the sink and source in $\mathcal{R}_\mathcal{S}$, no matter where the attacker starts it will always take the same path, hence the entropy is 0 for all starting locations as there is no uncertainty in its actions. However, when there are multiple paths to the source from the sink in $\mathcal{R}_\mathcal{N}$ the entropy is non-zero. When there are more paths from the starting location to the sink the entropy is greater. This is shown by the attacker starting at node 9 having a greater entropy than when it starts at node 5.

So far the analysis has focused on the individual routing examples, to compare $\mathcal{R}_\mathcal{S}$ to $\mathcal{R}_\mathcal{N}$ the Jensen–Shannon divergence (JSD) can be calculated. Figure 4.7

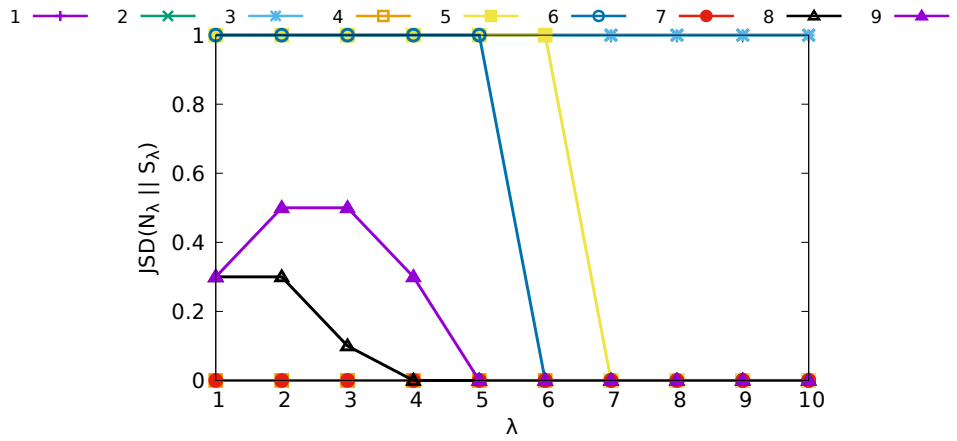


Figure 4.7: Jensen–Shannon divergence between $\mathcal{N}_\lambda$ and $\mathcal{S}_\lambda$ at different starting nodes

shows the JSD at the same times for the two attacker movement matrices, which demonstrates a number of interesting aspects of the two routing matrices. First, is that the actions the attacker takes when starting at 2 or 3 have completely diverged. This is because no messages are routed through 2 or 3 in $\mathcal{R}_\mathcal{S}$. Nodes 1, 4, and 7 have no divergence as the actions the attacker could take when starting at those locations are the same for both routing matrices. The remaining starting points (5, 6, 8 and 9) diverge early on, but converge at a later time when the attacker reaches the source. The most interesting of these four points is when the attacker starts at node 5. The aim in transforming $\mathcal{R}_\mathcal{N}$ into $\mathcal{R}_\mathcal{S}$ was to prevent the attacker from reaching the source within the safety period of 4 steps. Figure 4.7 shows that $\mathcal{R}_\mathcal{N}$ and $\mathcal{R}_\mathcal{S}$ have fully diverged until the 7^{th} step when the attacker under either routing matrix would then be at the source.

Figure 4.8 shows the Jensen–Shannon divergence at different times in $\mathcal{N}_\lambda$ and $\mathcal{S}_\mu$ (rather than when $\lambda = \mu$ as previously shown) when the attacker starts at nodes 1 to 9. These graphs allow routing matrices to be compared in more detail at specific time points. These figures show how diverged the actions an attacker could take in $\mathcal{R}_\mathcal{N}$ at λ compared to the actions it could take in $\mathcal{R}_\mathcal{S}$ at μ. Where 1 means that the actions have fully diverged and 0 means that the actions are the same. The height of the bottom two rows in Figure 4.8e (when the attacker starts at 5) that diverge is the same as the expected capture time in $\mathcal{R}_\mathcal{N}$, and the width of the first 6 columns is the same as the expected capture time in $\mathcal{R}_\mathcal{S}$. This figure is not symmetric and there is 0.3 divergence at $(\mathcal{S}_6, \mathcal{N}_2)$ because at that point the attacker would take the $(4, 1)$ edge in both routing matrices, but full divergence at $(\mathcal{S}_2, \mathcal{N}_6)$ because by time 6 the attacker has captured the source in $\mathcal{N}$, but at time 2 the attacker is at node 9 in $\mathcal{S}$. While the perturbation $\mathcal{S}$ works well for when the attacker starts at

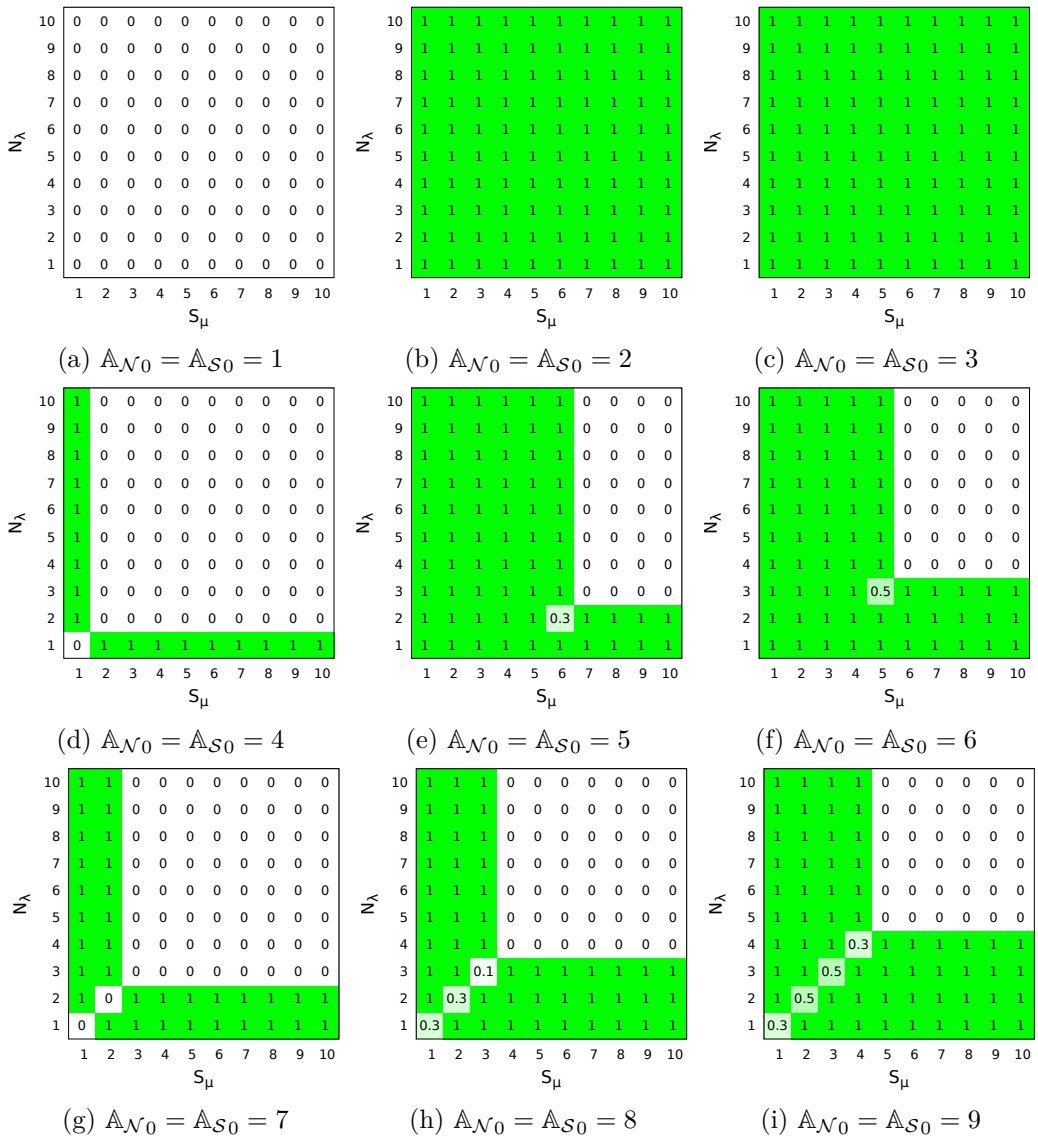


Figure 4.8: Jensen–Shannon divergence between $\mathcal{N}_\lambda$ and $\mathcal{S}_\mu$ at different times

node 5, it does not increase the time to capture when the attacker starts at node 9. This is a weakness of this example perturbation as the algorithm that performs the perturbation focuses on a single starting location. Other SLP techniques may be able to perturb a wider range of nodes sufficiently.

Finally, the expected capture times shown in Table 4.2 can be calculated based on the equations that give the expected hitting time for a Markov Chain. The results for when the attacker starts at the sink (node 5) in the two previous examples is shown in bold font. If the attacker starts at nodes 2 or 3 in $\mathcal{R}_\mathcal{S}$ it would never reach the source as there are no transitions in to either of those nodes that an attacker could use to follow a message.

Attacker Start Location (i)	1	2	3	4	**5**	6	7	8	9
$\mathbb{E}(\mathbb{A}_{\mathcal{N}t} = 1 \mid \mathbb{A}_{\mathcal{N}0} = i)$	0	1	2	1	**2**	3	2	3	4
$\mathbb{E}(\mathbb{A}_{\mathcal{S}t} = 1 \mid \mathbb{A}_{\mathcal{S}0} = i)$	0	∞	∞	1	**6**	5	2	3	4

Table 4.2: Expected capture time t of the source at node 1 for $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$ with the attacker starting at different nodes

4.6 Discussion

In this section, some issues raised by the methodology presented are discussed.

4.6.1 Maximal Information Loss

There are different ways to obtain the relevant proper perturbing paths. For example, a path that contains no loop might be obtained by one method while another path with loops might also be considered. Since the notion of paths captures source-converging paths, it means that Equation 4.10 will never be a maximum as $D_{\mathcal{S}}$ will contain some elements of $D_{\mathcal{N}}$. However, minimizing the number of common elements will result in high information loss. Specifically, it is better to introduce loops in (the non-overlapping elements of) $D_{\mathcal{S}}$ than in $D_{\mathcal{N}}$ as this will reduce the number of common elements.

4.6.2 Selecting Proper Junction Points

In this chapter a heuristic was proposed to select proper junction points to generate the set of properly perturbed paths. This heuristic does not specify how to choose which junction point should be used to generate the perturbed path. This is because there may be several junction points that could potentially be used to perturb the path, and some technique will need to be used to select them. Determining an optimal selection of junction nodes is likely to be a difficult problem to solve as each junction point could be used by multiple paths and produce multiple properly perturbed paths. The perturbed paths generated would need to ensure that they do not compete in a way that would lead the attacker towards the source. Future work will aim to show that selecting proper junction points is an NP-complete problem, a heuristic will also need to be developed to select near optimal proper junction points.

4.6.3 Unreliable Links

In this model, links have been assumed to be bidirectional and lossless, such that the sum of each column that involves nodes that receive messages adds to 1. However, when links become unidirectional or lossy (for example due to message collisions),

the sum may be less than 1. This also means that the domain $D_{\mathcal{S}}$ may be different to when the links are bidirectional. In this case, some of the techniques proposed will have to be adapted to specifically account for unidirectional links. On the other hand, if the unidirectional nature of links is transient, i.e., is short-lived, the current framework can still work if nodes are made to perform retransmissions (at the link-layer level). However, this technique will not work if message collisions occur. Most often, sensor nodes are not equipped with collision detectors and it is entirely possible that the matrix $\mathcal{R}_{\mathcal{N}}$ is different to the one assumed, as a node j may receive a message from node i first (in practice) rather than from node k (as specified by $[\mathcal{R}_{\mathcal{N}}]_{k,j} = 1$ and $[\mathcal{R}_{\mathcal{N}}]_{i,j} = 0$).

4.6.4 Modelling Complex Routing Protocols

The framework proposed is independent of any routing protocol, which was a key aim outlined at the start of this chapter. However, to provide SLP, a routing protocol will need to provide guarantees that $\mathcal{R}_{\mathcal{S}}$ will be supported. The way the routing matrices have been specified so far is that a single matrix applies to each message sent. However, this does not capture the full range of combinations that can occur. A more flexible view of routing protocol is to consider a sequence of routing matrices. A single routing matrix can be viewed as an infinite sequence of that routing matrix. Although only a selection of these routing matrices will be applicable to analysis, as any beyond the safety period do not need to be considered.

Certain classes of protocols can be considered using this modelling of routing matrices. As has been shown in this chapter, flooding and single-path routing can both be specified. In Chapter 5 and Chapter 6 variants of these two types of routing protocol will be demonstrated. Further routing protocols can be implemented by considering sequences of routing matrices. For example, publish/subscribe protocols [71, 92] can be implemented with separate routing matrices for the publisher, subscriber, and broker (if present) and the different actions they would perform. A limitation would be that the position of the sink, source and broker would need to be specified in advance and not dynamically determined.

In some situations nodes may need to generate messages before they would have received any to forward. In this case a possible implementation may require control messages to be sent (independently of the routing protocol) to notify the nodes to send the required messages. This means that the framework provides indication of the requirements to provide high levels of SLP.

4.7 Conclusion

In this chapter, the source location privacy problem in wireless sensor networks has been addressed from an entropy and divergence viewpoint. One major advantage of using such an approach is that it allows specific attacks and protocols to be abstracted away, focusing instead on the amount of information that is leaked by the network, lost by the attacker, or gained by attacker. While several other works have focused on analysing specific routing protocols or privacy metrics, this approach focused on understanding the basis of routing transformations to maximise the routing divergence. The framework is novel in that it allows the SLP-aware routing matrix to be configured in different ways, to give rise to potentially different SLP-aware routing protocols. Overall, the technique is useful as it provides a way to reason about the structure of arbitrary SLP-aware routing protocols against a local attacker, which was previously lacking in the existing work.

Chapter 5

Source Location Privacy Aware Routing for Delay-Tolerant Networks

A problem with modelling SLP routing protocols is that time is a difficult aspect to factor in. But as time is a crucial part of providing SLP-aware routing protocols it is important to consider. If the right events do not occur at the right time, then the routing protocol will fail to keep the attacker away from the source. In the previous chapter time is not modelled when using information theory to analyse the information an attacker gains, and instead the spatial aspect of where messages are routed is focused on. In this chapter SLP-aware routing will be modelled differently to include an aspect of time.

To achieve this temporal modelling constraint programming (CP) will be used. Constraint programming is a technique which applies a set of constraints to decision variables (which represent the state space) with the aim to maximise or minimise an objective function. Generic solvers then process all three parts and output an optimal solution if one is possible. Many problems have been expressed as constraint satisfaction problems (CSPs) such as (i) optimising sensor node deployment locations [20], (ii) energy efficient routing [34], and (iii) others like scheduling and planning [8]. Integer Linear Programming (ILP) is a subset of CP where the relations in the constraints between the variables must be linear.

In this chapter the scheduling of messages being routed from the source to the sink is modelled as an ILP problem. As modelling the time events occur continuously leads to an increase in the complexity of the model, time will instead be discretised into slots. A solver is then run offline to obtain an allocation of broadcast events to those time slots. The moves the attacker takes will be calculated based on those

broadcasts. By applying a suitable objective function that requires the broadcasts to be scheduled in such a way that the attacker does not find the source, the solver will produce an optimal SLP-aware broadcast schedule.

However, as the solution outputted by the model assumes network-wide knowledge and is specific to the scenario provided to the solver, it unsuitable for deployment on arbitrary WSNs. Therefore, a distributed algorithm — called *ILPRouting* — is then developed based on the structure and behaviour observed in the optimal solution. *ILPRouting* follows the same patterns observed in the model output, but scales to larger networks and different topologies. The results show that the protocol is near-optimal under certain parametrisations, independent of network size and application parameters such as the period between the source sending messages. The following three contributions are made in this chapter:

1. The SLP-aware routing protocol from a source to a sink is modelled as an ILP optimisation problem. Allowing an optimal schedule of message broadcasts to be obtained.

2. A novel distributed routing protocol — called *ILPRouting* — inspired from the output of the ILP model that focuses on delaying messages is developed.

3. Simulations of the routing protocol are performed using TOSSIM and results show that the protocol is near optimal in terms of privacy level, i.e., a low 1% capture ratio for certain parametrisations.

The rest of this chapter is structured as follows. The ILP constraint satisfaction model is specified in Section 5.1, the results of the model and how it influenced the *ILPRouting* algorithm's design is described in Section 5.2, and the routing protocol explained in Section 5.3. The routing protocol is then analysed using concepts from Chapter 4 in Section 5.4. The results in are then presented in Section 5.5. Implications are discussed in Section 5.6 and the chapter is finally concluded with a summary in Section 5.7.

5.1 Constraint Programming Model

In this section the ILP model will be described, including the rationale for certain constraints. The ILP model is written in IBM's Optimisation Programming Language (OPL) and a solver present in IBM ILOG CPLEX version 12.6.3 was used to generate an optimal output. There are two ILP solvers provided with ILOG CPLEX. The CP solver was used as it was able to produce optimal results faster and it also scaled to larger network sizes compared to the CPLEX solver. The CP solver proves optimality

Symbol	Description
$\|V\|$	The number of nodes
$\mathcal{W}$	The communication range of the nodes
$\mathcal{S}$	A set of node ids that send messages (sources)
$sink$	The sink id
$\mathcal{C}$	A mapping between node ids and their 2D coordinates
$\mathbb{A}_{start}$	The id of the node the attacker starts at
$\mathcal{W}_{\mathbb{A}}$	The eavesdropping range of the attacker
P_{src}	The time between the source sending messages
P_{safety}	The maximum simulation time
Ω	How many slots a second is divided into

Table 5.1: Integer Linear Programming Model Parameters

by showing that no better solution can be found, while the CPLEX solver obtains a lower bound proof using cuts and linear relaxation.[1]

The network was modelled as a directed graph $G = (V, A)$ where V is the set of nodes and A is the set of arcs. An arc is a 2-tuple (u, v) where u is the origin and v is the target. Each node in V was assigned a 2D coordinate $\mathcal{C} : V \to \mathbb{R} \times \mathbb{R}$ where the x and y coordinate of i can be accessed by $\mathcal{C}(i).x$ and $\mathcal{C}(i).y$ respectively. The euclidean distance $D : V \times V \to \mathbb{R}$ was calculated between each node. If the distance between two different nodes was less than or equal to the communication range $\mathcal{W}$ of the sensor nodes, then that arc is present in A. By using a directional graph more complex networks can be modelled (such as when links become unidirectional), however, in this work if $(u, v) \in A \implies (v, u) \in A$ due to the way range is used to calculate connectivity. The paths that messages can travel along is defined by the arcs in A. The paths the attacker can move along is modelled as a directed graph $G_{\mathbb{A}} = (V, A_{\mathbb{A}})$ rooted at the node with id $\mathbb{A}_{start}$ where the attacker starts. If the distance between two nodes is less than the attacker's eavesdropping range $\mathcal{W}_{\mathbb{A}}$ then $A_{\mathbb{A}}$ contains that arc, this also means the attacker can move along that arc. As the attacker graph is unidirectional there are two 1-hop neighbour functions that either return the neighbours reachable when following out edges from a node i by $\overleftarrow{N}_{\mathbb{A}}(i)$, or the nodes that can reach i when following in edges by $\overrightarrow{N}_{\mathbb{A}}(i)$. Once the attacker finds the source it will remain at that location, so the attacker cannot move along arcs which start at the source. As the attacker moves through the network, it can only move to be co-located with another node. In this work, the attacker's eavesdropping range is set to be equal to the sensor node's transmission range ($\mathcal{W} = \mathcal{W}_{\mathbb{A}}$).

The sink is a special node to which messages are routed and the sources ($\mathcal{S}$) are nodes that generate one message every source period (P_{src}). It is assumed that

[1] `ibm.com/software/integration/optimization/cplex-cp-optimizer/mp-cp`

there is a single sink and source. The model supports investigating multiple sources but only results for a single source will be presented.

There is an upper bound on the duration of the model (P_{safety}), within which routing and attacker movement is considered. Time ($\mathcal{T}$) is discretised by dividing it into slots, there are Ω slots per second. Each node can send a single message in each slot. A node may choose to send no messages in a given slot. The attacker can therefore either move in response to a single message or not move at all in a given slot. To ensure the attacker could reach the source, the attacker's starting position was never set so its distance in hops from the source was greater than $P_{src} \cdot \Omega$. As otherwise the safety period may be reached before the attacker has a chance to capture the source. The attacker can respond to a message a neighbour sends in a time slot if that message had not been previously responded to. If no messages are sent by a neighbour the attacker must remain where it is.

Time 0 is special as it is used to set the attacker's position, no messages are sent at this time. After time 0 when a node broadcasts a message it is assumed network links are perfect and all neighbours receive the message, even if multiple nodes broadcast in that time slot. In this case a collision detection and retransmission strategy is assumed to be used to ensure message delivery.

Messages ($\mathcal{M}$) that are generated by a source are represented by a unique 2-tuple where the first element (src) is the identifier of the source that generated the message and the second element (seq) is the message number.

$$\widehat{P_{safety}} = \lceil \Omega \cdot P_{safety} \rceil \tag{5.1}$$

$$\widehat{P_{src}} = \lceil \Omega \cdot P_{src} \rceil \tag{5.2}$$

$$\mathcal{T} = 0..\widehat{P_{safety}} \tag{5.3}$$

$$\mathcal{T}_1 = 1..\widehat{P_{safety}} \tag{5.4}$$

$$\mathcal{M} = \{\, (src,\, seq) \mid src \in \mathcal{S}, seq \in 1..\lceil P_{safety} \cdot P_{src} \rceil \,\} \tag{5.5}$$

$$D(i,j) = \sqrt{(\mathcal{C}(i).x - \mathcal{C}(j).x)^2 + (\mathcal{C}(i).y - \mathcal{C}(j).y)^2} \tag{5.6}$$

$$A = \{\, (u,v) \mid u,v \in V,\, D(u,v) \leq \mathcal{W} \wedge u \neq v \,\} \tag{5.7}$$

$$\overrightarrow{N}(i) = \{\, j \mid (j,i) \in A,\, i \neq j \,\} \tag{5.8}$$

$$\overleftarrow{N}(i) = \{\, j \mid (i,j) \in A,\, i \neq j \,\} \tag{5.9}$$

$$\begin{aligned} A_{\mathbb{A}} = {} & \{\, (u,v) \mid u,v \in V,\, D(u,v) \leq \mathcal{W}_{\mathbb{A}} \,\} \setminus \\ & \{\, (s,v) \mid s \in \mathcal{S},\, v \in V,\, s \neq v \,\} \end{aligned} \tag{5.10}$$

$$\overrightarrow{N}_{\mathbb{A}}(i) = \{\, j \mid (j,i) \in A_{\mathbb{A}},\, i \neq j \,\} \tag{5.11}$$

$$\overleftarrow{N}_{\mathbb{A}}(i) = \{\, j \mid (i,j) \in A_{\mathbb{A}},\, i \neq j \,\} \tag{5.12}$$

5.1.1 Objective Function

Two decision variables are used to capture the output of the model. These decision variables will have their values set by the ILP model solver during its execution. The first variable is the *broadcasts performed*, which is a three dimensional array of booleans with the node ids, message and time as the dimensions ($\mathscr{B} : V \times \mathcal{M} \times \mathcal{T} \rightarrow \mathbb{B}$). This variable is intended to capture whether a node sends a message at a given time. The other decision variable is the *attacker path*, which is a two dimensional array of booleans with time and the arcs an attacker can take as the dimensions ($\mathscr{P}_{\mathbb{A}} : \mathcal{T} \times A_{\mathbb{A}} \rightarrow \mathbb{B}$). This variable captures whether at a given time an attacker moves along an arc.

The objective function for this model is to maximise the distance between the attacker's final position and the source(s) in the network. As the aim in providing SLP is to prevent the attacker from reaching the source, the further the attacker is from the source the better SLP has been provided within the safety period.

$$\begin{aligned} \text{maximise} \quad & \sum_{\substack{s \in \mathcal{S} \\ e \in A_{\mathbb{A}}}} \mathscr{P}_{\mathbb{A}}(\widehat{P_{safety}}, e) \cdot D(s, e.v) \\ \text{subject to} \quad & \text{Routing Constraints ctR1 to ctR6,} \\ & \text{Attacker Constraints ctA1 to ctA7.} \end{aligned} \tag{5.13}$$

5.1.2 Routing Constraints

Here, the constraints on how messages are generated by the source and how they are routed in the network are described.

ctR1 At $t = 0$, no messages are broadcasted.

ctR2 From $t > 0$, each source node generates a message every P_{src} until the safety period is reached.

ctR3 No node can broadcast more than a single message concurrently. This means that in a given time slot a node must send one message or no messages.

ctR4 Once a message is broadcasted by a node it is not broadcasted by that node again.

ctR5 A node can only forward a message after a neighbour broadcasted that message in a previous time slot.

ctR6 All messages sent by the sources must reach the sink. This is satisfied when every message is broadcasted by at least one neighbour of the sink.

$$\forall n \in V \bullet \forall m \in \mathcal{M} \bullet \mathscr{B}(n, m, 0) = 0 \tag{ctR1}$$

$$\forall n \in \mathcal{S} \bullet \forall m \in \mathcal{M} : m.src = n \bullet \mathscr{B}(n, m, (m.seq - 1) \cdot \widehat{P_{src}} + 1) \tag{ctR2}$$

$$\forall \tau \in \mathcal{T}_1 \bullet \forall n \in V \bullet 1 \geq \sum_{m \in \mathcal{M}} \mathscr{B}(n, m, \tau) \tag{ctR3}$$

$$\forall m \in \mathcal{M} \bullet \forall n \in V \bullet \forall \tau_1 \in \mathcal{T}_1 \bullet \mathscr{B}(n, m, \tau_1) \implies 0 = \sum_{\tau_2 \in \mathcal{T}, \tau_2 > \tau_1} \mathscr{B}(n, m, \tau_2) \tag{ctR4}$$

$$\forall n \in (V \setminus \mathcal{S}) \bullet \forall m \in \mathcal{M} \bullet \forall \tau_1 \in \mathcal{T}_1 \bullet \mathscr{B}(n, m, \tau_1) \implies 1 \leq \sum_{\substack{n' \in \overrightarrow{N}(n) \\ \tau_2 \in \mathcal{T}, 0 < \tau_2 < \tau_1}} \mathscr{B}(n', m, \tau_2) \tag{ctR5}$$

$$\forall m \in \mathcal{M} \bullet 1 \leq \sum_{\substack{n \in \overrightarrow{N}(sink) \\ \tau \in \mathcal{T}_1}} \mathscr{B}(n, m, \tau) \tag{ctR6}$$

5.1.3 Attacker Constraints

This section details the constraints regarding how the attacker moves through the network. These constraints have been constructed so the attacker will follow the messages it receives. If the attacker was allowed to move independently of messages then it would move away from the source due to the objective function. Therefore, it is required that (i) the attacker only moves in response to messages (ctA4, ctA7) and (ii) the attacker will only consider a message once as otherwise it might follow that message as it moves away from the source (ctA5, ctA6).

To simplify attacker constraints four predicates about the attacker's movement are defined. `AM2A` checks if the attacker moved to n at time τ. `AM2NA` checks if the attacker moved to a neighbour of n at τ. `ASM` checks if the attacker remained where it was at τ. `AMBA` checks if an attacker moved because of a message m at τ.

$$\texttt{AM2A}(n, \tau) = 1 = \sum_{e \in A_{\mathbb{A}}, e.v = n} \mathscr{P}_{\mathbb{A}}(\tau, e) \tag{5.14}$$

$$\texttt{AM2NA}(n, \tau) = 1 = \sum_{n' \in \overrightarrow{N}_{\mathbb{A}}(n)} \texttt{AM2A}(n', \tau) \tag{5.15}$$

$$\texttt{ASM}(\tau) = 1 = \sum_{e \in A_{\mathbb{A}}, e.u = e.v} \mathscr{P}_{\mathbb{A}}(\tau, e) \tag{5.16}$$

$$\texttt{AMBA}(m, \tau) = 1 = \sum_{e \in A_{\mathbb{A}}, e.u \neq e.v} \mathscr{P}_{\mathbb{A}}(\tau, e) \wedge \mathscr{B}(e.v, m, \tau) \tag{5.17}$$

ctA1 At $t = 0$ the attacker moves from the attacker's starting position to that same position.

ctA2 The attacker makes exactly one move each time slot. If no message is received then the attacker moves from its current location n to n (stays still).

ctA3 A move must be from the attacker's current location.

ctA4 If the attacker moves to n from m at time τ, then it must be because at time τ the node n broadcasted a message.

ctA5 If the attacker receives a message that it has not previously moved in response to, then the attacker moves in response to that message.

ctA6 If the attacker moved in response to a message at time τ, then at no time $\tau' > \tau$ will the attacker move in response to that message again.

ctA7 If the attacker is at node n and no neighbours send a message, then the attacker moves along the (n, n) edge.

$$\mathscr{P}_{\mathbb{A}}(0, (\mathbb{A}_{start}, \mathbb{A}_{start})) = 1 \tag{ctA1}$$

$$\forall \tau \in \mathcal{T} \bullet 1 = \sum_{e \in A_{\mathbb{A}}} \mathscr{P}_{\mathbb{A}}(\tau, e) \tag{ctA2}$$

$$\forall \tau \in \mathcal{T}_1 \bullet 1 = \sum_{e_1, e_2 \in A_{\mathbb{A}},\, e_1.v = e_2.u} \mathscr{P}_{\mathbb{A}}(\tau - 1, e_1) \wedge \mathscr{P}_{\mathbb{A}}(\tau, e_2) \tag{ctA3}$$

$$\forall n \in V \bullet \forall \tau \in \mathcal{T}_1 \bullet 1 = \sum_{e \in A_{\mathbb{A}},\, e.v = n \wedge e.u \neq e.v} \mathscr{P}_{\mathbb{A}}(\tau, e) \implies 1 = \sum_{m \in \mathcal{M}} \mathscr{B}(n, m, \tau) \tag{ctA4}$$

$$\forall n \in V \bullet \forall m \in \mathcal{M} \bullet \forall \tau \in \mathcal{T}_1 \bullet \mathscr{B}(n, m, \tau) \wedge \texttt{AM2NA}(n, \tau - 1) \wedge 0 = \sum_{t_2 \in \mathcal{T},\, 0 < \tau_2 < \tau} \texttt{AMBA}(m, \tau_2) \implies \texttt{AM2A}(n, \tau) \tag{ctA5}$$

$$\forall \tau_1 \in \mathcal{T}_1 \bullet \forall m \in \mathcal{M} \bullet \texttt{AMBA}(m, \tau_1) \implies 0 = \sum_{\tau_2 \in \mathcal{T},\, \tau_2 > \tau_1} \texttt{AMBA}(m, \tau_2) \tag{ctA6}$$

$$\forall \tau \in \mathcal{T}_1 \bullet \forall e \in A_{\mathbb{A}} \bullet \mathscr{P}_{\mathbb{A}}(\tau - 1, e) \wedge 0 = \sum_{\substack{n \in \vec{N}_{\mathbb{A}}(e.v) \\ m \in \mathcal{M}}} \mathscr{B}(n, m, \tau) \implies \texttt{ASM}(\tau) \tag{ctA7}$$

5.2 Model Results

This section describes the output from the ILP model for one combination of parameters. The model was run for a variety of different parameters including 3x3, 4x4 and 5x5 grids with the sink and source at different locations. Larger networks became infeasible to run due to the large amount of memory required. The results in Figure 5.1 show a 5x5 network with the source in the top left corner and the attacker starting at the sink which was positioned in the centre. This configuration was chosen as it is similar to the SourceCorner configuration and allowed the state space of the model to be small enough for a solution to be found within a reasonable

time. Figures 5.1a to 5.1g show the pattern of broadcasts and Figure 5.1h shows how the distance of the attacker from the source changes over time. From the results four major trends were observed:

1. The routing path should go around the sink and approach from the opposite direction to the source.
2. Some routes should take the shortest path from the source to the sink.
3. Messages should be delayed so multiple messages are grouped together.
4. Messages should be delayed as late as possible with respect to the safety period.

The contributions of these trends are to achieve the following:

- To obscure the direction in which the source that is sending messages is (trends 1 and 2).
- To reduce the number of moves the attacker is able to make towards the source by grouping messages and delaying sending messages as late as possible (trends 3 and 4).

Some of these observations have been used in previous work, while others have not. For example, having the route approach the sink from a direction other than the one the source is in has been used in Phantom Routing [81] and Ring-based Routing [196]. There are also some algorithms whose routes occasionally take the shortest path [60]. However, delaying and grouping messages has not been previously used in providing SLP. By delaying and grouping messages the attacker should end up with less time to make moves towards the sink within the safety period as shown by Figure 5.1h, where the attacker moves mainly near the end of the simulation (towards the top of the graph). By grouping and allowing messages to be delivered out-of-order the attacker is also forced to require a larger memory to ensure previously received messages are ignored in the future.

The ILP modelling of SLP has assumed that links are reliable and bidirectional and that there are also no message collisions. As the routing protocol indicated takes a single path, message retransmissions will need to be performed in the implementation to increase the reliability of delivering a message to the sink. These are two issues this causes: (i) retransmissions will alter the way messages are delayed and grouped, and (ii) messages may need to take an alternate route if a link is unavailable for a period of time. These two issues will be addressed during in the next section which details the implementation of *ILPRouting*.

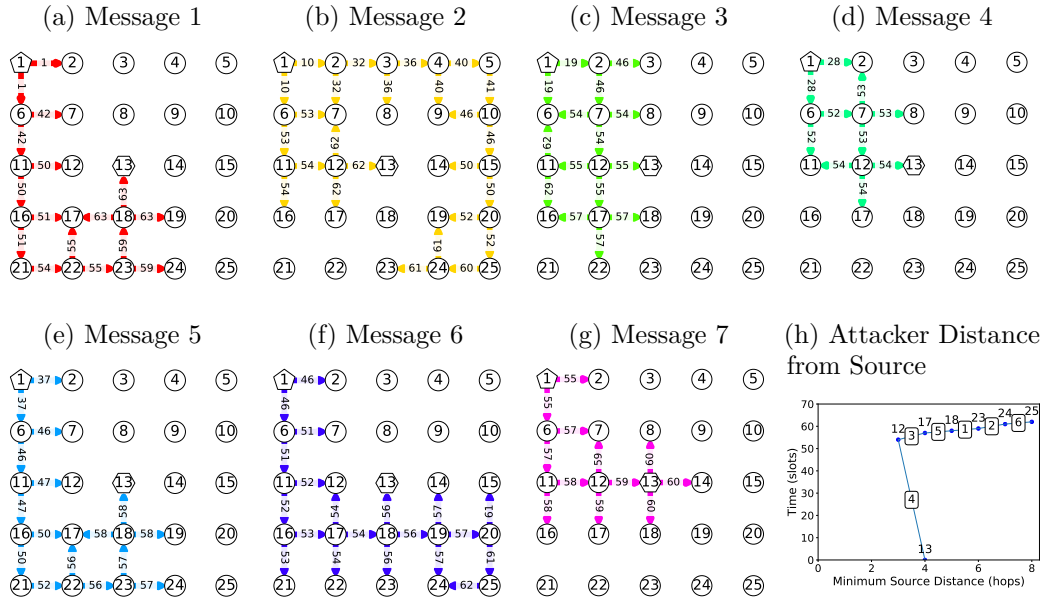


Figure 5.1: An output of the ILP model for a 5x5 network with node 1 sending 7 messages to the sink at node 13 with $P_{src} = 1$ and $\Omega = 9$. The attacker starts at node 13. Message broadcasts are represented by arrows from the sending node to the receiving nodes, the arrows are labelled with the time slot of the broadcast. Lines in (h) are labelled with the message number the attacker responded to and the location of the attacker is shown above the point.

5.3 Routing Algorithm

In this section the implementation of the *ILPRouting* algorithm is described. This algorithm is inspired from analysing the model output. It is not expected to behave optimally, but is intended to be more generic and not restricted to a specific topology. There are four stages:

1. **AvoidSink**: In this first stage, messages are routed around the sink.
2. **Backtrack**: Messages may end up attempting to go towards the sink and not having any valid routes, so they need to backtrack.
3. **ToSink**: Once a message has finished routing to avoid the sink it needs to be delivered to the sink.
4. **FromSink**: Finally, the message is sent in a starburst from the sink.

5.3.1 Stage 1: Avoid Sink

In order to provide SLP the most important task that needs to be achieved is to be able to route messages reliably around the sink and have them approach it from the opposite direction that the source is in. This requires every node to be aware

of its neighbours, as one will be chosen to be the next in the path. Every node n also needs to know its distance to the sink $\Delta_{sink}(n)$, its distance from the source $\Delta_{src}(n)$, and the distance between the sink and source Δ_{ss}. These values are found by the landmark nodes (sink/source) flooding the network in a setup phase. Every node should know these values for each node in its 1-hop neighbourhood.

There is a parameter called *message group size* which specifies how many messages to group together. By grouping together, the algorithm will delay the messages each hop so that the messages reach Δ_{ss} hops travelled at the same time. Messages sent earlier in the group will be delayed longer. The delay is specified in Equation 5.18, where i is the position in the message group, P_{src} is the source period and α is the time it takes a message to send from one node to another. This is calculated at the source when the message is sent.

$$delay(i) = \frac{i \cdot P_{src} + \alpha \cdot \Delta_{ss}}{\Delta_{ss}} \tag{5.18}$$

Retransmissions are used to ensure reliability along the route. If retransmissions to a target fail a fixed number of times, then that target is blacklisted and another node will attempt to be used. If further retransmissions fail, then a $\langle$poll$\rangle$ message is sent which asks neighbours to broadcast their information. This is performed to ensure that the most up-to-date information is being used to make the routing decisions. Retransmissions are stopped when an acknowledgement packet is received or the maximum number of retransmissions have been sent. Even if retransmissions cause extra delay or messages need to take a different path (due to an unavailable link), these delays for a single link are bounded by how nodes decide to stop sending along the unavailable link and try another. This means that even with performing these extra measure to provide reliability, messages will be in a similar area when they have travelled Δ_{ss} hops.

5.3.2 Stage 2: Backtrack

There is the possibility that a message may reach a node which has no further neighbours to choose from, for example node 20 in Figure 5.2. In this case the message backtracks to a node that is further from the sink that was not the previous hop in the route. The next hop is then chosen using the avoid sink logic. This should allow a message to avoid the area of nodes that should not be used in the routing path (highlighted in grey in Figure 5.2). Backtracking is only done on node n when $\Delta_{src}(n) < \Delta_{ss}$ as this node would otherwise lead the message closer to the sink.

$$c(n, m) = \Delta_{src}(m) > \Delta_{src}(n) \wedge (\Delta_{sink}(n) > {}^{\Delta ss}/_{2} \vee \Delta_{sink}(m) \geq \Delta_{sink}(n)) \tag{5.19}$$

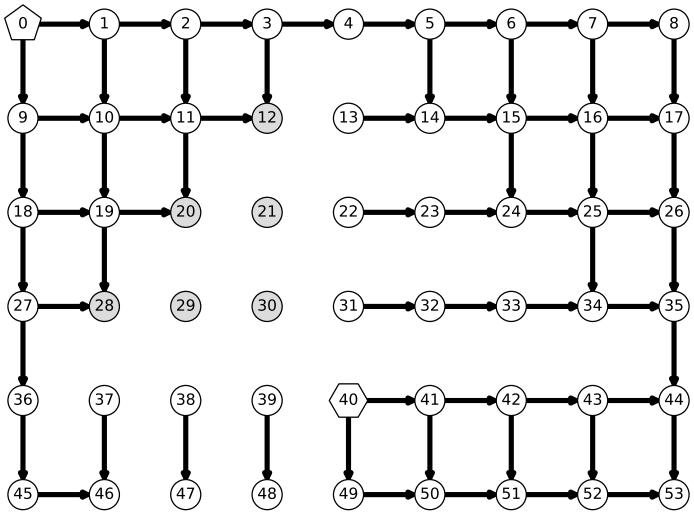


Figure 5.2: Showing the links a node n has between each m neighbour in its one hop neighbourhood iff $c(n, m)$ from Equation 5.19. Nodes that have no neighbours are marked in gray. The source is a pentagon and the sink is a hexagon.

5.3.3 Stage 3: To Sink

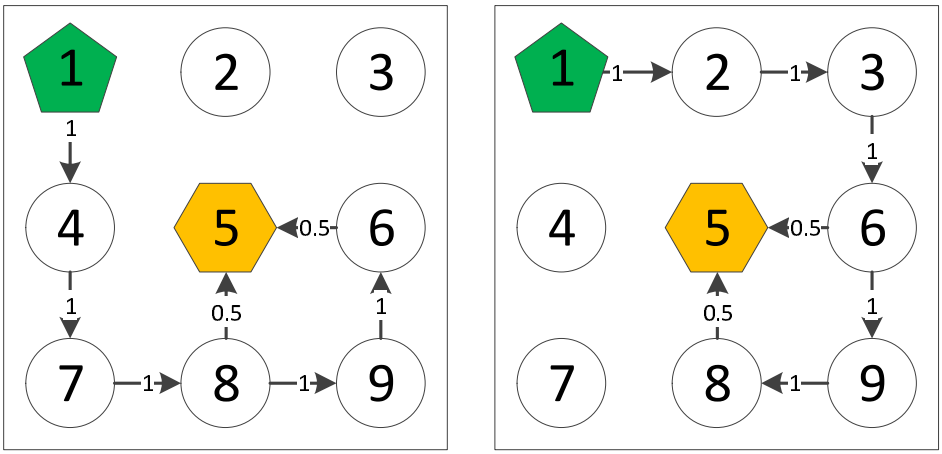
When a node's $\Delta_{src}(n) \geq \Delta_{ss}$ then the message has passed the area that would lead it towards the source. When this message has reached a node with no neighbours that are further from the sink, the message is at a local maxima for the sink distance, so the message is routed back towards the sink. When a message reaches a certain number of hops travelled it is routed back to the sink. This is to ensure bounded message latency and to prevent the message going too far on very large networks. As with routing to avoid the sink, retransmissions and blacklisting are used to provide a reliable message transmission to the sink.

5.3.4 Stage 4: From Sink

Finally, once the message reaches the sink, it is broadcasted in a starburst pattern away from the sink in all directions for a limited number of hops. Model results where the sink was the furthest point required this behaviour to ensure that the attacker is lured to the furthest location from the source. This behaviour was also observed in Figure 5.1g.

5.4 Competing Paths Analysis

In this section the *ILPRouting* protocol will be analysed in terms of competing paths and information loss. The structure of *ILPRouting* aims to have messages travel around the sink and approach from behind. In the grid network there is a chance



(a) *ILPRouting* route 1 (b) *ILPRouting* route 2

Figure 5.3: The different representations of *ILPRouting* routing paths.

for it to go one of two ways around the sink. As nodes 2 and 4 shown in Figure 5.3 both have the same sink and source distance the protocol would randomly choose between sending messages between the two nodes. Therefore, the protocol would be represented as an infinite sequence of routing matrices where each matrix is randomly chosen between the two examples.

There are three advantages *ILPRouting* has, the first is that by alternating between the two different routes, the attacker has the potential to miss receiving messages. The second, is that similar to the example given in Chapter 4, the route an attacker would follow has the potential to be longer than the safety period. The third, is that the two paths compete at a number of nodes behind the sink (5, 6, 8, and 9), which allows the attacker to be pulled back and forth in this area.

The information loss compared to *Protectionless* flooding can be calculated as follows. With $D_{\mathcal{N}}$ representing the transitions under *Protectionless* and the two *ILPRouting* matrix transitions by $D_{\mathcal{S}_1}$ and $D_{\mathcal{S}_2}$. A safety period of 4 is used as that is twice the distance it would take an attacker to capture the source under *Protectionless*. None of these sets of paths change with a safety period of 4. The information loss is then calculated at 75% for both path variants, however, the information loss will be greater if the attacker is at a location that will not eavesdrop a transition from the other path. This means that *ILPRouting* is effective in reducing the information leaked to an attacker.

$$D_{\mathcal{N}}^{P'_{safety}} = D_{\mathcal{N}} = \begin{array}{c}\{\,(1,2),(1,4),(2,3),(2,5),(3,6),(4,5),\\ (4,7),(5,6),(5,8),(6,9),(7,8),(8,9)\,\}\end{array} \tag{5.20}$$

$$D_{\mathcal{S}_1}^{P'_{safety}} = D_{\mathcal{S}_1} = \{\,(1,4),(4,7),(6,5),(7,8),(8,5),(8,9),(9,6)\,\} \tag{5.21}$$

$$D_{\mathcal{S}_2}^{P'_{safety}} = D_{\mathcal{S}_2} = \{\,(1,2),(2,3),(3,6),(6,5),(6,9),(8,5),(9,8)\,\} \tag{5.22}$$

$D_{\mathcal{N}}^{P'_{safety}}$	$(1,2)$	$(1,4)$	$(2,3)$	$(2,5)$	$(3,6)$	$(4,5)$	$(4,7)$	$(5,6)$	$(5,8)$	$(6,9)$	$(7,8)$	$(8,9)$
$D_{\mathcal{S}_1}^{P'_{safety}}$	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓
$D_{\mathcal{S}_2}^{P'_{safety}}$	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗

Table 5.2: Are the transitions from $D_{\mathcal{N}}$ in $D_{\mathcal{S}_1}^{P'_{safety}}$ and $D_{\mathcal{S}_2}^{P'_{safety}}$?

$$IL(D_{\mathcal{N}}, D_{\mathcal{S}_1}) = \frac{1+0+1+1+1+1+0+1+1+1+0+0}{12} = \frac{8}{12} = \frac{3}{4} \tag{5.23}$$

$$IL(D_{\mathcal{N}}, D_{\mathcal{S}_2}) = \frac{0+1+0+1+0+1+1+1+1+0+1+1}{12} = \frac{8}{12} = \frac{3}{4} \tag{5.24}$$

5.5 Results

The results presented in this section were generated by running experiments in TOSSIM as outlined by the experimental setup in Chapter 3. The algorithm being tested has four parameters: the maximum walk length, the buffer size, the number of messages to group and the probability the message is sent directly to the sink. The maximum walk length provides a finite bound of the length of the path ⟨normal⟩ messages take in large networks, it was set to 100 hops. The buffer size is the amount of space reserved to hold messages that are being delayed, it was set to 15 messages no more than 15 concurrent messages are being sent in the network at one time. The number of messages to group was varied between {1, 2, 3, 4}. Finally, the probability of sending a message directly to the sink was varied between {0, 0.1, 0.2, 0.3, 0.4, 0.5}. As part of these results six key metrics are analysed:

1. **Broadcast and Attacker Behaviour** — This is the patten in which messages are sent and the path the attacker takes in response.

2. **Received Ratio** — This is the percentage of messages that were sent by the source and received by the sink.

3. **Capture Ratio** — This is the percentage of runs in which the attacker reaches the location of the source, i.e., captures the source.

4. **Messages Sent Per Second** — This is the average number of messages sent by all nodes in the network per second.

5. **Latency** — This is the average time it takes a message sent by the source to be received at the sink.

6. **Attacker Distance** — This is the average attacker distance from the source recorded at the end of a run.

All results shown in this section are for when the probability that a message is sent directly to the sink is 20%. Lower probabilities gave similar, but slightly worse results and higher probabilities produced much worse results. Therefore, in this section only the results for when the probably of the message being sent directly to the sink was 20%. Graphs demonstrating results for different probabilities are be presented in Appendix D.

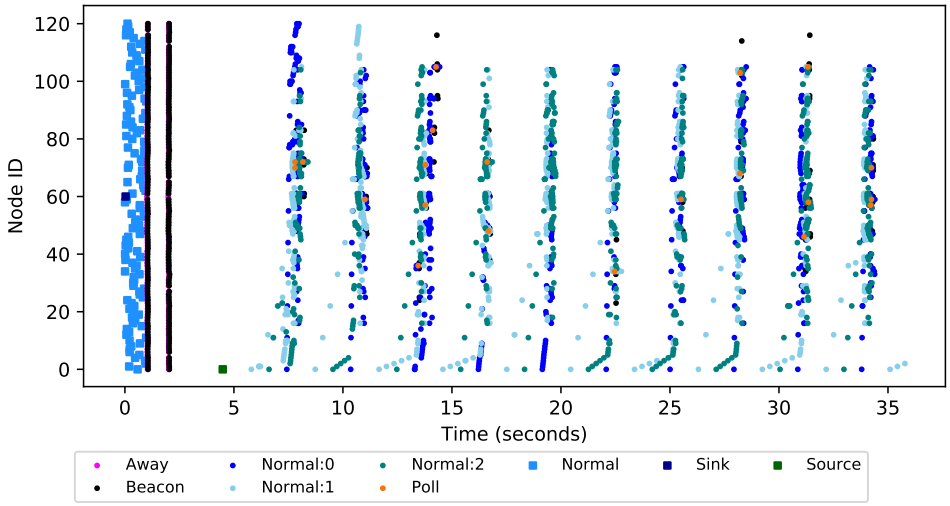
Testbed results are not presented for this algorithm as the FlockLab topology does not meet the assumptions made about the layout of the network assumed. The issue is that there is insufficient space for messages to travel around the sink to approach from behind. In a network deployed for asset monitoring a large number of nodes are expected to be deployed over a large area. These nodes are expected to be relatively uniformly distributed. The FlockLab topology is more linear than grid shaped, which leads to the lack of space around the sink. As no other testbed was suitable, no testbed results are presented.

5.5.1 Behaviour

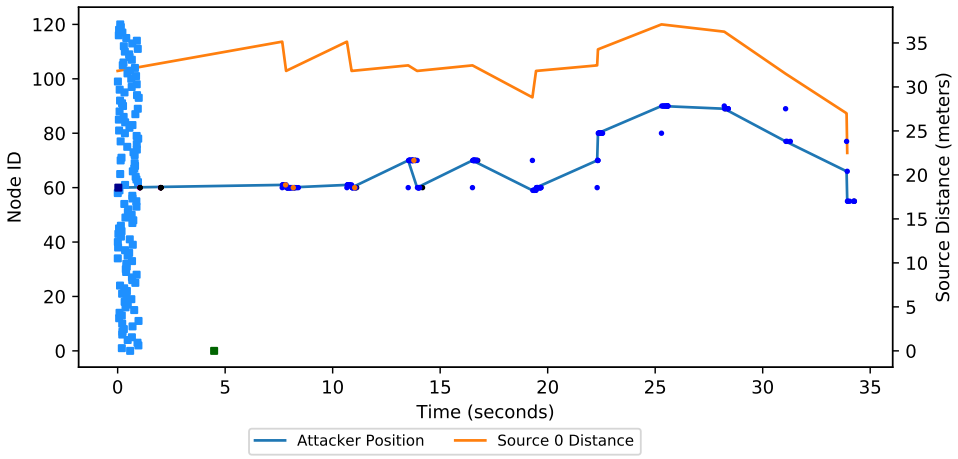
In Figure 5.4 the events that occur when *ILPRouting* is running is shown. The message group size is set to 3 and the ⟨normal⟩ messages are coloured to indicate what group they are in. Figure 5.4a shows how the messages group together. By looking around the source at node 0, messages can be seen to be delayed and join up with the main line of message transmissions. In Figure 5.4b rather than moving in response to a message every second (the source period), the attacker only moves once every three seconds. So delaying messages to group them together is successful in reducing the number of moves an attacker will make.

5.5.2 Receive Ratio

A high receive ratio between 75% and 95% is observed in Figure 5.5. Fewer messages were delivered with larger networks and larger messages group sizes. This suggests



(a) Messages Sent

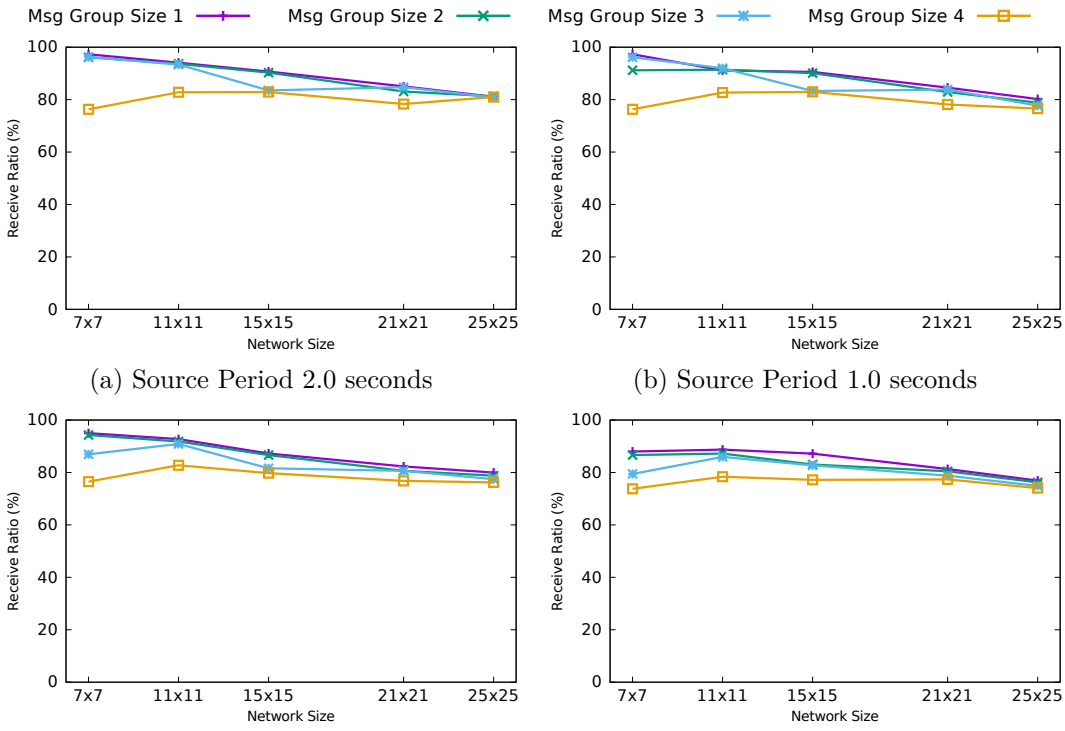


(b) Attacker messages received with attacker position and source distance

Figure 5.4: A demonstration of the behaviour of the *ILPRouting* algorithm and how the attacker moves in response to the messages sent.

that the attacker had the opportunity to eavesdrop most of the ⟨normal⟩ messages, meaning that the privacy level imparted by the algorithm is due to the efficiency of the protocol and not due to the unreliability of the network.

Because of the way the simulations terminate (when the safety period is reached or the attacker captures the source), it is possible that some messages may still be in the process of being routed to the sink. As these messages have not yet reached the sink they are not counted as received, even if they would be delivered successfully in the future. The delivery ratio of messages for *ILPRouting* will therefore be under-reported, especially when the group size is large as this means a larger wait time is used. The *Dynamic* and *DynamicSPR* techniques that will be



(a) Source Period 2.0 seconds

(b) Source Period 1.0 seconds

(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 5.5: *ILPRouting* results showing the receive ratio.

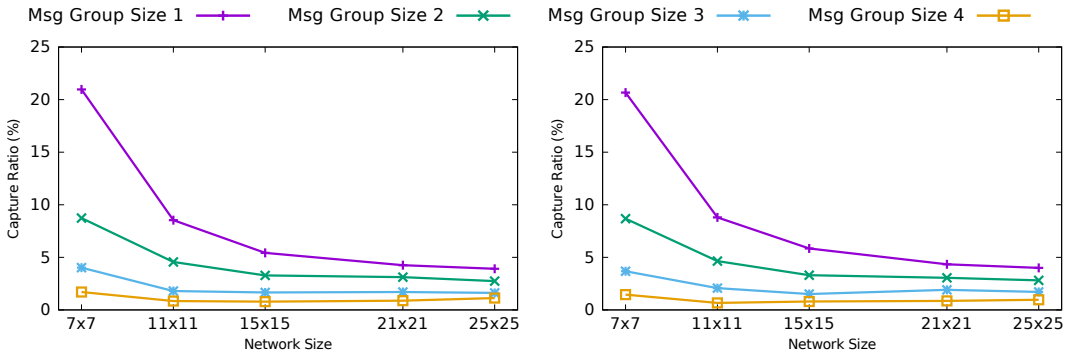
presented in Chapter 6 are less susceptible to this problem as ⟨normal⟩ messages have a low latency in those strategies. However, (as will be explained) *ILPRouting* uses a high latency to impart SLP.

5.5.3 Capture Ratio

There is a less than 10% probability of the attacker capturing the source within the safety period for these parameter combinations shown in Figure 5.6. However, as more messages are grouped together, the capture ratio falls lower to 1%. This matches with the intuition that a greater number of messages grouped together would give the attacker fewer chances to respond to messages within the safety period. On the other hand, with messages being grouped together, it suggests that the message delivery latency may be increased. When the size of the message group is high, the protocol delivers near optimal privacy level.

5.5.4 Messages Sent per Second

As the different network sizes being varied each have a different safety period, the number of messages sent has been normalised with respect to the simulation length



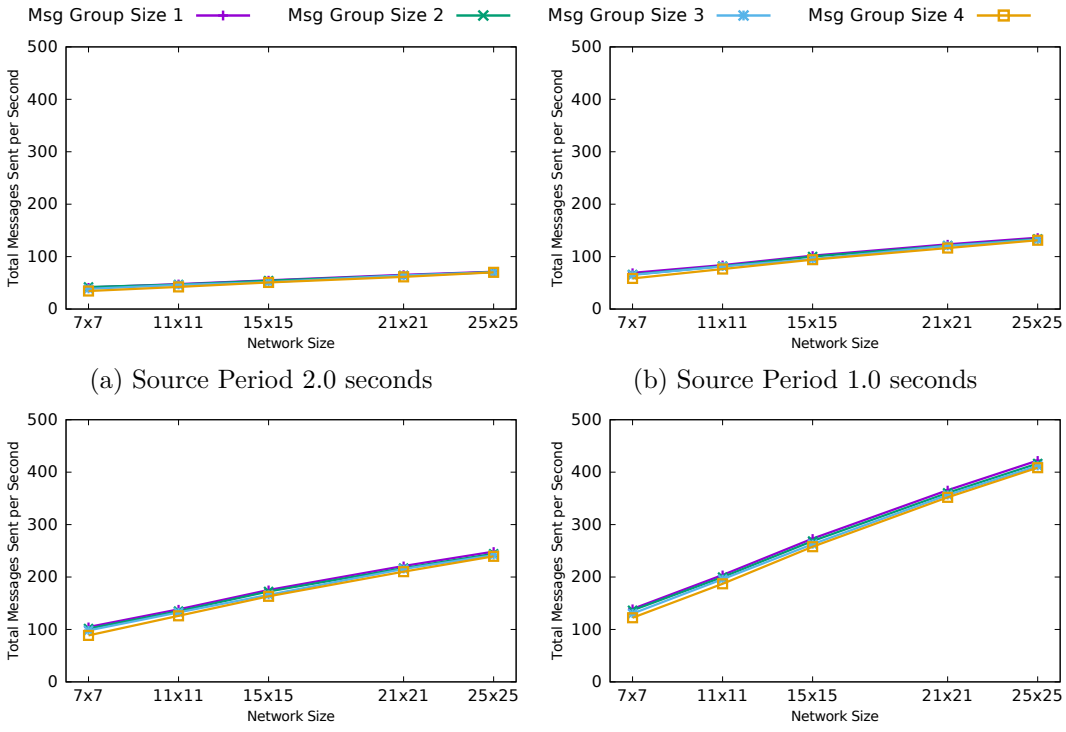
(a) Source Period 2.0 seconds

(b) Source Period 1.0 seconds

(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 5.6: *ILPRouting* results showing the capture ratio.



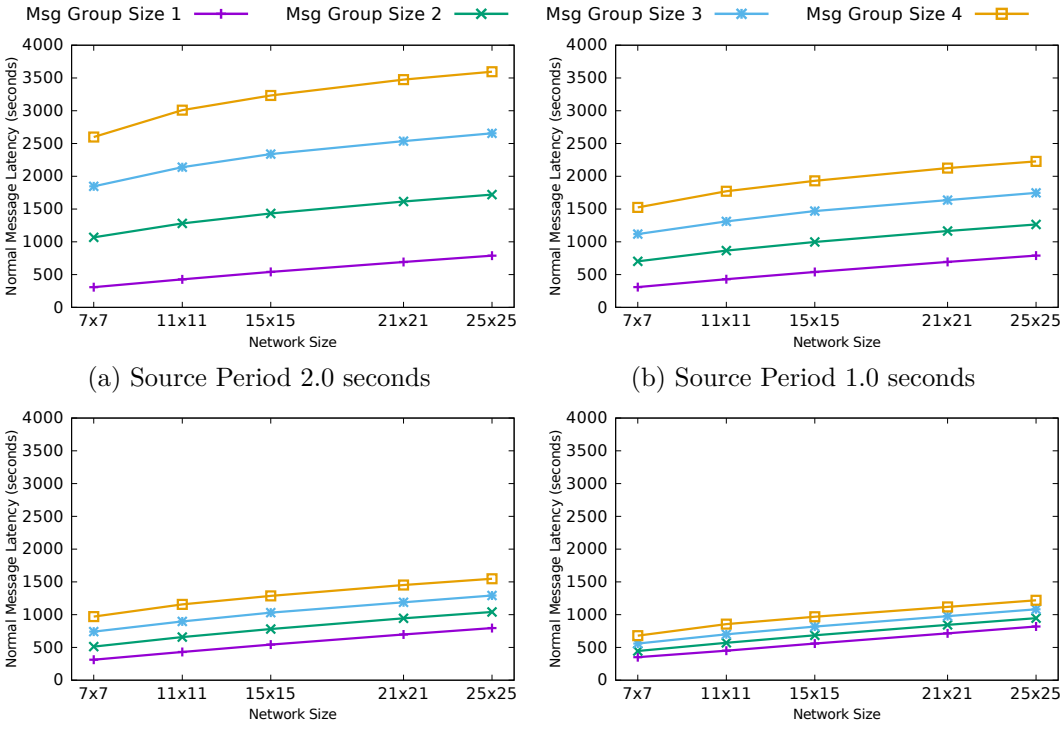
(a) Source Period 2.0 seconds

(b) Source Period 1.0 seconds

(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 5.7: *ILPRouting* results showing the messages sent per second.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 seconds

(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

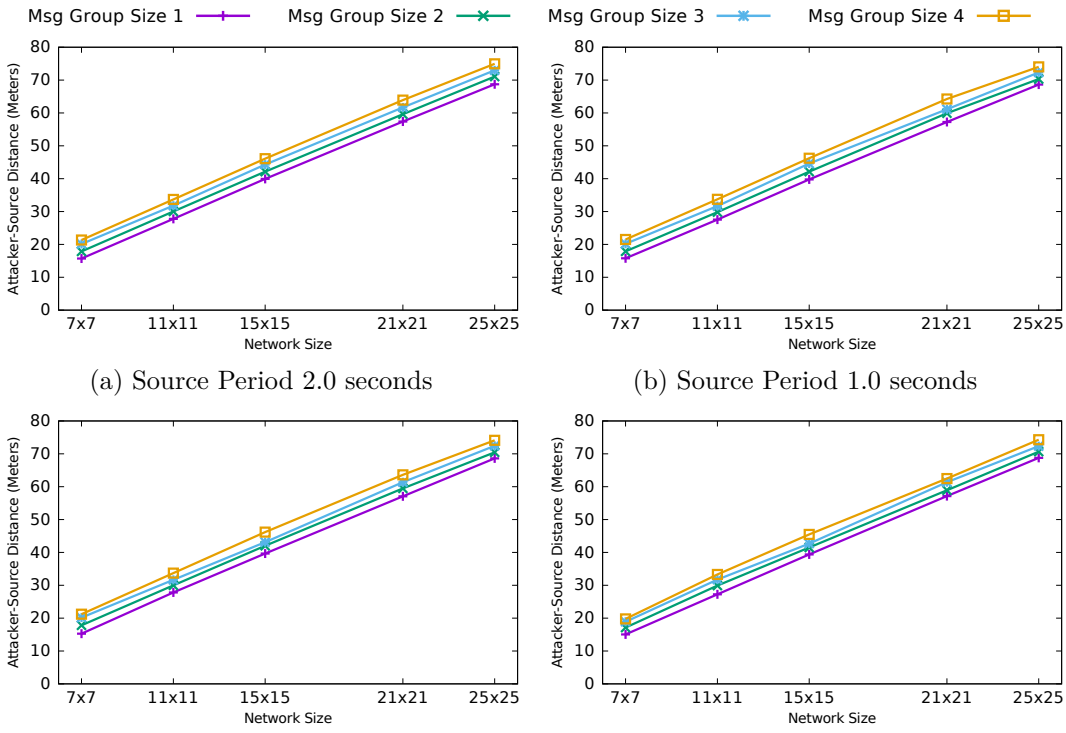
Figure 5.8: *ILPRouting* results showing the message latency.

to allow the results to be compared across different safety periods. Note that it would not be suitable to also normalise with respect to the number of nodes in the network as *ILPRouting* does not expect each node to send a similar number of messages.

The main observation of the results in Figure 5.7 is that lower source periods (faster message rates) send more messages per second. The results show that the number of messages sent per second does not vary greatly for different message grouped sizes. Larger message group sizes appear to require slightly fewer messages sent per second. This is because larger group sizes have a lower delivery ratio which causes more messages to still be in transit when the safety period expires. So not all these messages will have finished reaching the source. This means that larger group sizes will have a slightly higher number of messages sent than reported in these results, which is expected to be similar to the results for lower messages group sizes.

5.5.5 Latency

As shown in Figure 5.8, the longer the time between messages, the larger the latency between a source sending the message and the sink receiving the message. This is because messages are delayed to make sure that a certain number requested (as specified by the message group size) reach the sink-source distance at a similar time.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 seconds

(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 5.9: *ILPRouting* results showing the attacker distance.

When there are multiple messages the per hop delay needs to be longer. These results indicate that there is a trade-off between latency and capture ratio, to obtain a better capture ratio using *ILPRouting* a larger latency will need to be incurred. For many applications a latency of this magnitude will be acceptable. For example, tracking the location of a slow moving panda will not be adversely affected. But in scenarios where very low latency is important, such as on a battlefield, this technique may be less suitable.

5.5.6 Attacker Distance

As the objective function for *ILPRouting* was to maximise the attacker's distance from the source, it is important to analyse if the aim was achieved. The results are presented in Figure 5.9, As the network size increases, the attacker's average distance from the source also increases. This is because there is more space for the attacker to be pulled further away with larger networks. Larger message group sizes also improve the distance the attacker is from the source, but only by a small amount.

5.6 Discussion

In this section some issues and alternative approaches to using integer linear programming to develop solutions to SLP are discussed.

5.6.1 Integer Linear Programming Complexity

Computing an optimal solution to the ILP formulation in this chapter is an expensive procedure, because obtaining an optimal solution to an Integer Linear Programming problem is NP-complete [55]. Due to this complexity and the large state space that needed to be explored it was only feasible to run the IBM ILOG CPLEX solver for 5×5 grid networks over a short time period. To obtain a solution the solver consumed a large quantity of both CPU and RAM resources (2 days of compute time and over 100 GB of memory). This meant that the solver could not be run on larger network sizes, the consequence of this is that certain kinds of techniques might not have been able to be computed (e.g., Fog Routing [44]) due to the inability to explore the spatial redundancy those kinds of techniques require. This may be why this modelling produced a technique that used temporal redundancy to provide SLP.

5.6.2 Other Objective Functions

The objective function chosen aimed to ensure that the attacker ended up as far from the source nodes as possible. However, this is stricter than the actual requirement of SLP which is that the attacker must not capture (i.e., be co-located) with a source within the safety period. Attempting to optimise this model for other objective functions such as minimising latency or the total number of messages sent might produce different output indicating other kinds of routing that would provide SLP.

5.6.3 Impact of Grouping Messages

One of the downsides of delaying messages to group them at similar distances from the source is an increase in collisions. The likelihood of these collisions is lower when there are multiple paths around the sink. Another downside is that the way messages are being routed around the sink means *ILPRouting* is more susceptible to areas of poor reliability and is less able to reroute the path, like a protocol such as CTP [57] would be able to do so. One potential solution to the increase in collisions would be to merge packets if they are buffered on one node. This would reduce the number of transmissions being performed in the area of the network where messages meet.

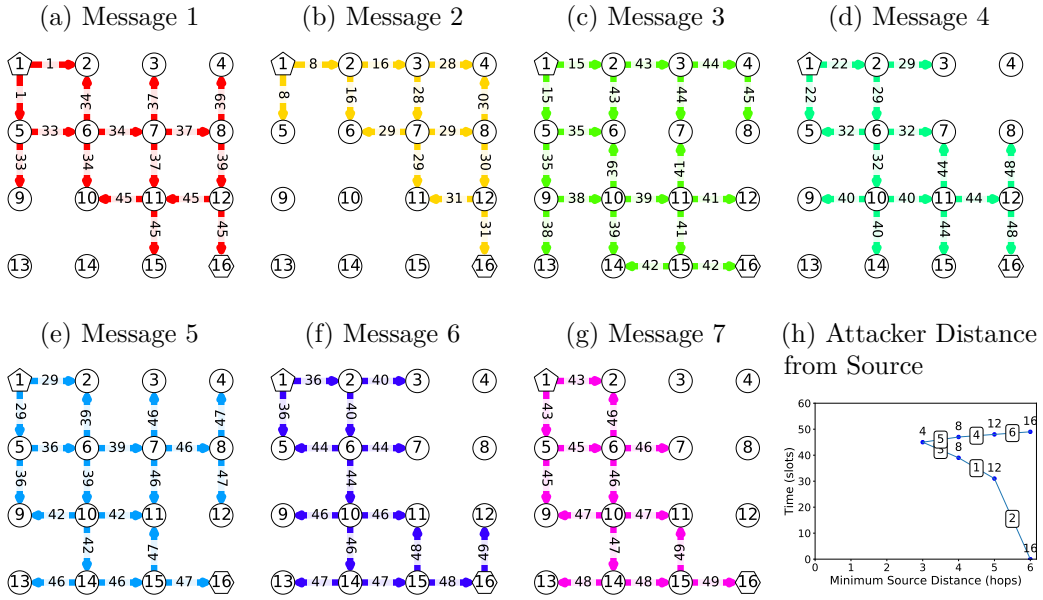


Figure 5.10: An output of the ILP model for a 4x4 network with node 1 sending 7 messages to the sink at node 16 where the attacker starts at node 16 with $P_{src} = 1$ and $\Omega = 7$. Message broadcasts are represented by arrows from the sending node to the receiving nodes, the arrows are labelled with the time of the broadcast. Lines in (h) show the message number the attacker responded to, the location of the attacker is above the point.

5.6.4 Alternate Configurations

In this chapter only one configuration that matches the SourceCorner configuration was focused on. There are other configurations of the sink and source location that might need to be considered. For example, Figure 5.10 shows results for a 4x4 network with the source in the bottom left corner and the attacker and sink in the top right corner. An important difference between this 4x4 configuration and the 5x5 configuration used earlier is that the attacker starts at the furthest point so there is no room for it to move further away. Because the attacker starts at the furthest node in the 4x4 configuration it is interesting to see how the attacker is allowed to come closer before being pulled back.

When the attacker and sink were the node furthest from the source the following observations were made:

1. The messages should lure the attacker closer to the source initially.
2. Messages should be routed around that position.
3. The attacker should finally be drawn back to the sink, this includes the sink sending messages.

These observations are different to the pattern previously seen. This is mostly because there is no space behind the sink, so it would be impossible to route messages to approach the sink from a direction further from the source. Instead messages must first lead the attacker to a different area of the network, so that they can subsequently use another area of the network. This would have caused the attacker to miss messages for a small time, but because the model is trying to maximise attacker distance the attacker needs to be pulled back to the sink. This is done by messages deliberately being routed in such a way that the attacker would receive them from the direction in which the sink is in.

While this works well in the ILP model result it is unlikely that it would work well in a real world scenario. This is because it would be hard to estimate the attackers exact location such that messages would lure it towards the sink and not towards the source. However, luring the attacker to a certain area of the network and then avoiding that area is a valid strategy to delay the attacker by causing it to miss receiving messages. This shows that modelling the SLP problem using constraint programming can provide some solutions that are impractical to deploy, but can provide useful insight into ways to provide SLP.

5.7 Conclusion

In this chapter a formalisation of SLP-aware routing as an ILP constraint satisfaction problem has been presented. This model produced optimal routes computed based on global network knowledge for small networks with a specific topology, making these routes unamenable to deployment. Using the concepts provided by the model's output, a distributed routing protocol inspired by the optimal solution called *ILPRouting* was developed. This technique uses delay to group messages together to reduce the progress an attacker can make towards the source by causing it to miss messages. Large scale simulations were performed to judge the performance. The results show that low capture ratios (high levels of SLP) can be obtained with the trade-off being a higher delivery latency.

Chapter 6

Dynamic Determination of Fake Source Parameters

A consequence of using *ILPRouting* is that source location privacy is provided by delaying messages. For time-sensitive scenarios the latency needs to be as minimal as possible (e.g., real-time tracking), which indicates that ⟨normal⟩ messages should travel along the shortest path from the source to the sink. One class of techniques in which shortest path routing is used for routing ⟨normal⟩ message are fake source techniques that have been previously shown to be effective in providing source location privacy [75, 78]. The idea behind them is to allocate fake sources at certain locations in the network which then broadcast ⟨fake⟩ messages. These ⟨fake⟩ messages are encrypted and padded so an attacker is unable to distinguish between ⟨normal⟩ and ⟨fake⟩ messages. This means the attacker responds when receiving either of the message types. Fake sources use these ⟨fake⟩ messages to lure the attacker to their location instead of the location of the real source.

As the Fake Source Selection Problem (FSSP) has been proved to be NP-complete [78] it is hard to decide when, where, and for how long nodes should generate ⟨fake⟩ messages. Previous work has assumed parameters that are fixed when a protocol is deployed on the network [78]. This assumption prevents the protocol from being able to adapt to changing network conditions.

In this chapter, two heuristics that adapt to changing network conditions will be presented, called *Dynamic* and *DynamicSPR*. These heuristics will determine the values of parameters online during execution using equations derived from a timing analysis of events. As these two heuristics use parameters identified as important by [78] a comparison will be performed against this heuristic called *Static*. Results show that these two techniques are capable of providing high levels of SLP.

6.1 Problem Statement

Given a network $G = (V, E)$, a distributed eavesdropping attacker $\mathbb{A}$ that is initially located at the sink q, a source $s \in \mathcal{S}$, a safety period P_{safety}, a routing algorithm $\mathcal{R}$, the problem is to select a set of fake sources $\mathcal{F} \subseteq V$ such that $\forall f \in \mathcal{F}$, assign a tuple (n, p, d) to f, where n is either a temporary or permanent fake source, p is the fake message period and d is the duration over which f sends fake messages such that $\mathbb{A}$ does not reach s within P_{safety} when $\mathbb{A}$ is following the movement rules defined in Section 3.5. The fake source selection problem (assigning the tuple (n, p, d)) is an NP-complete problem [78], so there is a need for heuristics to calculate good values that provide high levels of SLP whilst using as little energy as possible.

As *Protectionless* flooding provides no SLP, it will be used as the routing protocol $\mathcal{R}$ for $\langle$normal$\rangle$ messages. Flooding is the worst case protocol as it provides the most information to the attacker, as the attacker should never miss receiving a message. *Dynamic* and *DynamicSPR* will be shown to provide SLP by having low capture ratios even when flooding is used to route $\langle$normal$\rangle$ messages.

6.2 Static Overview

A heuristic called *Static* was initially proposed in [78] that the on-line heuristics *Dynamic* and *DynamicSPR* will improve upon. *Static* is first previewed, before *Dynamic* and *DynamicSPR* are presented. *Static* is based on three main parameters, which capture the trade-offs involved between SLP and energy usage:

- the temporary fake source (TFS) duration (D_{TFS})
- the temporary fake source period (P_{TFS})
- the permanent fake source (PFS) period (P_{PFS})

A number of additional parameters were investigated in *Static* that proved not to be interesting in terms of algorithm performance:

- the probability of a node becoming a temporary fake source
- the probability of a node becoming a permanent fake source

These two parameters will not be dynamically determined and will be fixed at 100%. So when a node has the option to become a fake source it will always become a fake source.

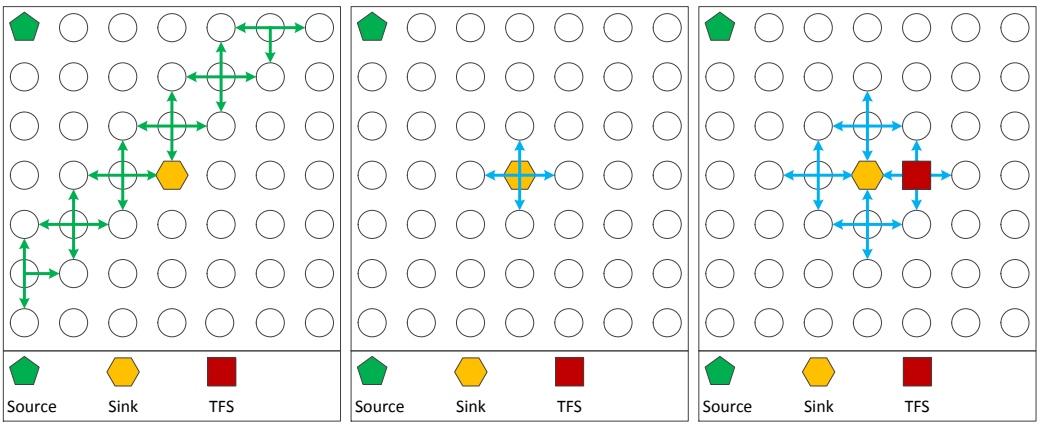
The *Static* algorithm works as follows:

1. The sink initially floods the network with ⟨away⟩ messages ($\mathcal{A}$) to inform nodes of their sink distance.
2. When the source detects an asset it sends a ⟨notify⟩ message ($\mathcal{NO}$) to the sink.
3. The source node repeatedly sends a ⟨normal⟩ message $\mathcal{N}_i$ with a time period between messages of P_{src}, beginning with $\mathcal{N}_1$.
4. When the sink receives $\mathcal{NO}$ it then broadcasts a ⟨choose⟩ message $\mathcal{C}$ to its 1-hop neighbours.
5. When a neighbour of the sink receives $\mathcal{C}$ it becomes a TFS.
6. A TFS broadcasts a ⟨fake⟩ message $\mathcal{F}_i$ with period P_{TFS} for a duration of D_{TFS}, before becoming a normal node and broadcasting a ⟨choose⟩ message $\mathcal{C}$.
7. When a normal node receives $\mathcal{C}$ it becomes a PFS if the node believes itself to be the furthest node in the network from the sink, otherwise it will become a TFS. A PFS broadcasts a ⟨fake⟩ message $\mathcal{F}_i$ with period P_{PFS}.

- When a node receives a previously unencountered $\mathcal{A}$, $\mathcal{NO}$, $\mathcal{N}_i$ or $\mathcal{F}_i$ it updates its last seen sequence number for that message and rebroadcasts the message.
- When a node receives a previously unencountered $\mathcal{C}$ it updates its last seen sequence number for that message.

The authors of [78] performed a large-scale simulation to show the high levels of SLP achievable by *Static*, when varying the values of the main variables, i.e., D_{TFS}, P_{TFS} and P_{PFS}. The *Static* scheme has fake sources initially selected close to the sink, that slowly *move* away to positions further from the sink and the source. This allows the attacker to be slowly pulled away from the source and was been informed by one of the results of [78] which showed that a higher TFS duration improved SLP.

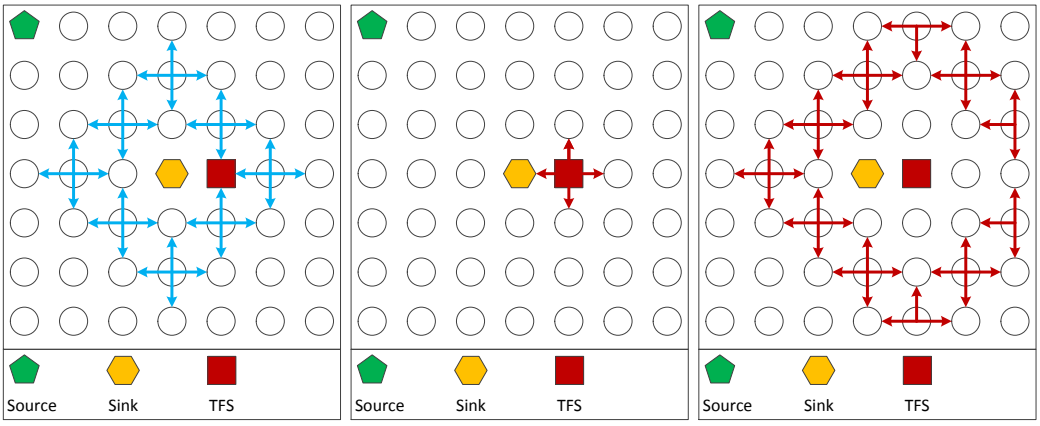
The problem is that if *Static* is deployed, D_{TFS}, P_{TFS}, and P_{PFS} must be fixed at compile time, making *Static* susceptible to poor performance under changing network conditions or incorrect parameters chosen at compile-time. Making these decisions correctly requires precise understanding of the environment where the network is being deployed. Therefore, in the next section a novel heuristic — called *Dynamic* — that determines these parameter values at *runtime* is presented. This heuristic uses a timing analysis of events that occur to calculate values needed for certain parameters. By doing so this analysis obviates the need to set parameters at compile time based on estimations of the network.



(a) The source floods the network with ⟨normal⟩ messages, repeating every P_{src} seconds. Nodes record their Δ_{src}.

(b) After the sink receives the first ⟨normal⟩ message it waits ω seconds then floods an ⟨away⟩ message to start the fake source allocation.

(c) All nodes 1-hop from the source that receive the ⟨choose⟩ become TFSs. This diagram only shows 1 for simplicity.



(d) The ⟨away⟩ flood continues, allowing nodes to record Δ_{ss} and $\Delta_{sink}(j)$.

(e) The TFS starts sending ⟨fake⟩ messages, for the duration of the fake source.

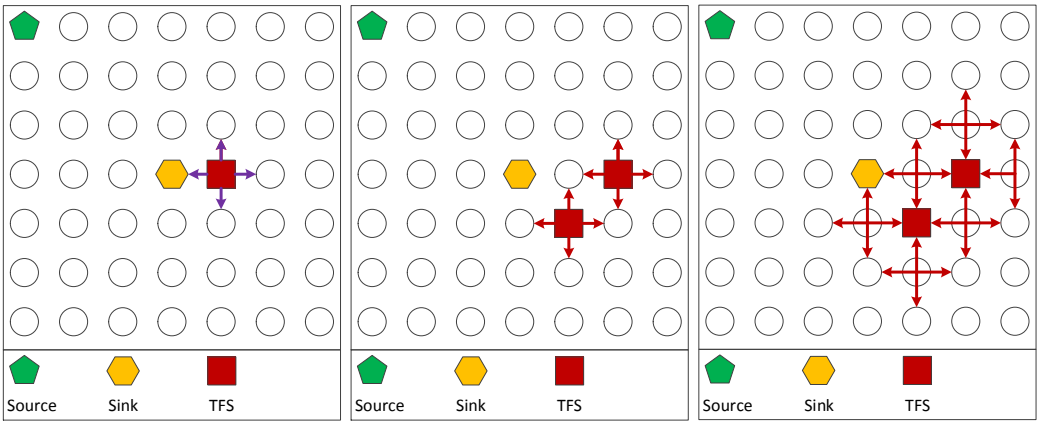
(f) The ⟨fake⟩ message flood should lure the attacker away from the source.

Figure 6.1: The common actions for the *Static* and *Dynamic* algorithms.

6.3 *Dynamic*: Estimating Parameters Online

Three types of information are required for the online evaluation of D_{TFS}, P_{TFS}, and P_{PFS}: (i) parameters fixed at compile time of the firmware that are known to all nodes, (ii) information needed to derive the parameters that is not required during network execution, and (iii) information that must be calculated during network execution and passed on to other nodes in the network.

The first piece of information required is the *source period* (P_{src}), which is fixed at compile time. The second required piece of information is the *delivery delay* (α), which is the time taken for a message sent by one node to be received at a

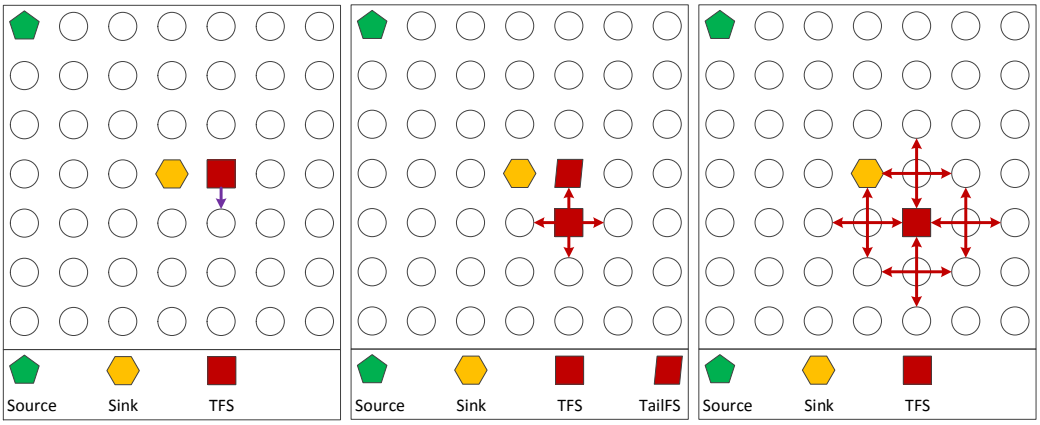


(a) Choose next fake source by broadcasting a ⟨choose⟩ message.

(b) Nodes further from the source than the original TFS become fake sources.

(c) The fake sources continue sending ⟨fake⟩ messages.

Figure 6.2: Spread of fake sources under *Static* or *Dynamic* after a TFS duration expires.



(a) Choose next fake source by unicasting a ⟨choose⟩ message.

(b) Original TFS becomes a TailFS until it detects a further TFS.

(c) TailFS detects further TFS and becomes normal again.

Figure 6.3: Spread of fake sources under *DynamicSPR* after a TFS duration expires.

neighbour. This delay has been the subject of research as it is an important value to take into account during clock synchronisation [114, 190]. Typically, α will be very small compared to the source period P_{src}, which means that its impact may be negligible on the final values. Finally, three pieces of network information are computed during execution: (i) the sink-source distance (Δ_{ss}), (ii) the sink distance for node n ($\Delta_{sink}(n)$) and (iii) the source distance for node n ($\Delta_{src}(n)$). All distances are calculated in hops. With this information, the three important parameters of *Static* can be calculated.

To develop the framework for calculating the parameters, the fake source

strategy of luring an attacker away from a source is viewed through an analogy with tug-of-war. In tug-of-war, two teams are pulling on either end of a rope, the team that pulls a marker on the rope over a certain point wins. In SLP, sending messages can be thought of as pulling on the rope, the source is on one side and the fake sources are on the other. The attacker is the marker that will cause one team to lose, i.e., $\mathbb{A}$ captures the source because the *pull* from the source is greater than the pull from the fake sources. The *pull* of the source or of the fake sources will often be referred to during this chapter. This analogy is also used in determining the values of parameters of *Dynamic* at runtime. To provide SLP, the fake sources need to exert a bigger pull on the attacker than the source. A smaller pull means that the attacker would instead be led to the source.

For this analysis the following notation is used. The current time will be denoted by t, at $t = 0$ the source node sends the first $\langle$Normal$\rangle$ message. The 1-hop neighbourhood of a node j is denoted by $1\text{HopN}(j)$. $\perp$ is used to represent variables that have not been initialised, or whose value is unknown. For example, if a node does not know their distance to another node it will be set to $\perp$. A function $\max_{\perp}$ that finds the maximum of all arguments that are not $\perp$ is also defined.

6.3.1 Message Timings

This analysis assumes that the source routes messages to the sink via a shortest path. This is the optimal case for *Protectionless* flooding, but messages can occasionally take longer routes if links are unreliable, messages collide, or there are other reliability issues. Assuming that the shortest paths are used is best from the attacker's perspective (when *Protectionless* is used) because it is the scenario that allows them to find the source fastest. Alternative routing protocols, such as the Collection Tree Protocol [57], would also guarantee this. Thus,

- The i^{th} $\langle$normal$\rangle$ message is sent by the source at time $S_{src}(\mathcal{N}_i)$:

$$S_{src}(\mathcal{N}_i) = (i-1)P_{src} \tag{6.1}$$

- In the worst case (from the defender's perspective) when no SLP is provided, an attacker will receive $\mathcal{N}_1$ at $t = \alpha\Delta_{ss}$ and will have moved to be at $\Delta_{ss} - 1$ hops from the source. For the i^{th} $\langle$normal$\rangle$ message the attacker receives, the distance between the attacker and the source (Δ_{as}) will be:

$$\Delta_{as}(\mathcal{N}_i) = \max(0, \Delta_{ss} - i) \tag{6.2}$$

- An attacker will receive the i^{th} $\langle$normal$\rangle$ message at time $R_{\mathbb{A}}(\mathcal{N}_i)$:

$$R_{\mathbb{A}}(\mathcal{N}_i) = S_{src}(\mathcal{N}_i) + \alpha\Delta_{as}(\mathcal{N}_{i-1}) \tag{6.3}$$

The rest of the analysis will detail how fake sources are allocated and how $\langle$fake$\rangle$ messages are sent. As flooding will be used to route the $\langle$fake$\rangle$ messages (because flooding is used for routing the $\langle$normal$\rangle$ messages), the impact of collisions and unreliable links is small. This is because there is a large amount of redundancy as many other links are involved with the routing of $\langle$fake$\rangle$ messages.

To allocate fake sources, $\langle$Choose$\rangle$ messages will be sent and received at these earliest times:

- The $\langle$choose$\rangle$ message $\mathcal{C}$ is sent by the sink at time $S_{sink}(\mathcal{C})$. Once the sink receives a $\langle$notify$\rangle$ message ($\mathcal{NO}$) it will send a $\langle$choose$\rangle$ message to start the fake source allocation process.

$$S_{sink}(\mathcal{C}) = S_{src}(\mathcal{NO}) + \alpha\Delta_{ss} \tag{6.4}$$

- A node j where $j \in 1\text{HopN}(sink)$, will receive $\mathcal{C}$ from the sink at the earliest at the time $R_j(\mathcal{C})$.

$$R_j(\mathcal{C}) = S_{sink}(\mathcal{C}) + \alpha \tag{6.5}$$

- Subsequent $\langle$choose$\rangle$ messages will be received at times described in the next section.

When node j receives a $\langle$choose$\rangle$ message $\mathcal{C}$, the node becomes a TFS and starts broadcasting $\langle$fake$\rangle$ messages. The time at which j becomes a TFS is $\tau_{TFS}(j)$, where:

$$\tau_{TFS}(j) = \begin{cases} R_j(\mathcal{C}) & \text{if } j \in 1\text{HopN}(sink) \\ \tau_{TFS}(k) + D_{TFS}(k) + \alpha & \text{if } k \in 1\text{HopN}(j) \wedge \Delta_{sink}(k) < \Delta_{sink}(j) \end{cases} \tag{6.6}$$

The number of $\mathcal{N}$ messages sent between $t = 0$ and $t = \tau_{TFS}(j)$ is $\Sigma_j(\mathcal{N})$:

$$\Sigma_j(\mathcal{N}) = \left\lceil \frac{\tau_{TFS}(j)}{P_{src}} \right\rceil \tag{6.7}$$

6.3.2 Calculating TFS Duration

Intuition

When a node is selected as a TFS, the period at which it generates ⟨fake⟩ messages defines the pull it exerts on the attacker. Since the node knows the pull exerted by the source (defined by P_{src}), it can estimate the duration over which the fake source needs to apply its greater pull to drag the attacker away from the source. The derivation will consider the time cost of sending messages through the network.

Derivation

To calculate D_{TFS}, the duration is set to be the difference in time between when the TFS is created and when the attacker receives the next ⟨normal⟩ message, less the time it takes to send the next ⟨choose⟩ message:

$$D_{TFS}(j) = R_{\mathbb{A}}(\mathcal{N}_{\Sigma_j(\mathcal{N})+1}) - \tau_{TFS}(j) - \alpha \tag{6.8}$$

The next step is to calculate the duration for nodes that are n-hops away from the sink. In this case the attacker has now received $\mathcal{N}_n$ and the duration of this TFS is to last until $\mathcal{N}_{n+1}$ is received. The knowledge about a node k that is $(n-1)$-hops from the sink, can be used to calculate when the node j that is n-hops from the sink becomes a TFS at $\tau_{TFS}(j)$.

$$\begin{aligned}
\tau_{TFS}(j) &= \tau_{TFS}(k) + D_{TFS}(k) + \alpha \\
&= \tau_{TFS}(k) + (R_{\mathbb{A}}(\mathcal{N}_n) - \tau_{TFS}(k) - \alpha) + \alpha \\
&= R_{\mathbb{A}}(\mathcal{N}_n) \\
&= (n-1)P_{src} + \alpha(\Delta_{ss} - (n-1))
\end{aligned} \tag{6.9}$$

$$\begin{aligned}
R_{\mathbb{A}}(\mathcal{N}_{n+1}) &= S_{src}(\mathcal{N}_{n+1}) + \alpha\Delta_{as}(\mathcal{N}_{n+1-1}) \\
&= nP_{src} + \alpha\Delta_{as}(\mathcal{N}_{n+1-1}) \\
&= nP_{src} + \alpha(\Delta_{ss} - n)
\end{aligned} \tag{6.10}$$

Therefore the duration is given by:

$$D_{TFS}(j) = R_{\mathbb{A}}(\mathcal{N}_{n+1}) - \tau_{TFS}(j) - \alpha = P_{src} - 2\alpha \tag{6.11}$$

As α is not available to the nodes during runtime and because α is expected to be

very small relative to P_{src}, α is ignored in the final result.

$$D_{TFS}(j) = P_{src} \tag{6.12}$$

Once the duration expires a ⟨choose⟩ message will be sent. So for the TFS j it will send a ⟨choose⟩ message at $S_j(\mathcal{C})$ and a neighbour k with $\Delta_{sink}(j) < \Delta_{sink}(k)$ will receive it at $R_k(\mathcal{C})$.

$$S_j(\mathcal{C}) = \tau_{TFS}(k) + D_{TFS}(k) \tag{6.13}$$

$$R_k(\mathcal{C}) = S_j(\mathcal{C}) + \alpha \tag{6.14}$$

6.3.3 Calculating Number of Fake Messages to Send

Intuition

There exists a relation between the TFS duration, the TFS period, and the number of messages that are sent in that period. Either the number of messages to be sent can be defined in terms of the duration and period, or the period can be defined in terms of the duration and number of messages. The latter is chose, as shown in Equation 6.16, as this allows the algorithm to make its decisions based on the number of ⟨normal⟩ messages sent during the TFS duration.

$$\#_{\mathcal{F}}(j) = \frac{D_{TFS}(j)}{P_{TFS}(j)} \tag{6.15}$$

$$P_{TFS}(j) = \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)} \tag{6.16}$$

Derivation

Three approaches are provided to calculate the number of ⟨fake⟩ messages to send.[1]

Pull Fixed 1 (`Fixed1`):

$$\#_{\mathcal{F}}(j) = 1 \tag{6.17}$$

This approach aims to keep the attacker where it is. By sending 1 ⟨fake⟩ message each period the attacker should be pulled towards a fake source, then pulled towards a real source.

[1] In previously published works [24, 25] different strategies were used to calculate $\#_{\mathcal{F}}(j)$. These strategies sent a much larger number of messages. The implementation of *Dynamic* has since been modified to improve error handling that would otherwise have led to lost messages. These old strategies are presented in Appendix C as they have significantly higher energy usage than the approaches presented here.

Pull Fixed 2 (`Fixed2`):

$$\#_{\mathcal{F}}(j) = 2 \tag{6.18}$$

If ⟨fake⟩ messages are lost due to noise or collisions then it is possible that an attacker may miss being lured by a ⟨fake⟩ message under the previous strategy. So rather than zigzagging back and forth, the attacker will be led closer to the source. By sending multiple messages in a TFS period the chance of this happening is reduced.

Pull Random (`Rnd`):

$$\#_{\mathcal{F}}(j) = \mathbf{rand}(1, 2) \tag{6.19}$$

However, sending two messages each period should be twice as expensive as sending one message each period. By randomly sending one or two messages each TFS period the energy cost of `Fixed2` should be reduced. There will also be a chance to send an extra message to assist in pulling back the attacker in the case of lost messages.

6.3.4 Calculating TFS Period

Intuition

The TFS period of a fake source defines how fast ⟨fake⟩ messages are sent by the node, i.e., it captures the strength of the pull of the fake source. A TFS must send at least 1 ⟨fake⟩ message to keep parity with the number of ⟨normal⟩ messages sent. $c > 1$ messages need to be sent to ensure that at least one ⟨fake⟩ message reaches the attacker when collisions occur. To pull an attacker back h hops, $h \times c$ different ⟨fake⟩ messages need to be sent.

Derivation

Using the TFS duration $D_{TFS}(j)$ and the number of ⟨fake⟩ messages to send $\#_{\mathcal{F}}(j)$ the TFS period is obtained by dividing them. The P_{TFS} can not be allowed to go below 3α as collisions would then occur between the current and the previously broadcast ⟨fake⟩ message. Again as α is unavailable, $P_{TFS}(j)$ is finally defined without it. This requires $\#_{\mathcal{F}}(j)$ to be defined in such a way that it does not lead to $P_{TFS}(j)$ being set to 3α or less.

$$P_{TFS}(j) = \max\left(3\alpha, \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)}\right) \tag{6.20}$$

$$P_{TFS}(j) = \frac{D_{TFS}(j)}{\#_{\mathcal{F}}(j)} \tag{6.21}$$

6.3.5 Calculating PFS Period

Intuition

By the time a PFS has been created, many TFSs should have been pulling the attacker away from the source. This means the PFS should not need to send as many ⟨fake⟩ messages as a TFS. It will need to send at least 1 ⟨fake⟩ message for each ⟨normal⟩ message sent by the source, plus some extra to consider collisions.

Derivation

If an attacker can be guaranteed to have been moved far enough from the source by TFSs, then having $P_{PFS}(j) = P_{src}$ would be preferable. However, the attacker's position should not be relied upon to be far enough away from the source's, meaning the algorithm requires $P_{PFS}(j) < P_{src}$, such that any PFSs retain the ability to pull back the attacker and cope with collisions of ⟨fake⟩ messages.

A lower bound on the period $P_{PFS}(j) \geq \alpha$ exists, as the PFS cannot physically send messages more often than that. There also exists an upper bound of $P_{PFS}(j) < P_{src}$ as the PFS should not broadcast slower than the source.

The technique used here is to set the PFS period to the source period multiplied by the receive ratio of ⟨fake⟩ messages at the source ($\psi_{src}(\mathcal{F})$). This is justified because it means for every ⟨normal⟩ message sent the PFS should send enough ⟨fake⟩ messages for the attacker to receive at least one ⟨fake⟩ message.

$$P_{PFS}(j) = \max\left(P_{src} \times \psi_{src}(\mathcal{F}), 3\alpha\right) \quad (6.22) \qquad P_{PFS}(j) = P_{src} \times \psi_{src}(\mathcal{F}) \quad (6.23)$$

In order to calculate this receive ratio, the source node needs to keep a record of the number of ⟨fake⟩ messages sent and received. The sequence number records the overall number sent and an additional counter records the number of times the sequence number was updated as the number received. This information must be transmitted back to the PFS, where it is needed, using ⟨normal⟩ messages. This is best effort as a PFS may not receive every ⟨normal⟩ message sent.

$$\psi_{src}(\mathcal{F}) = \frac{\text{fake\_messages\_received} + 1}{\text{fake\_sequence\_number} + 1} \quad (6.24)$$

6.3.6 Summary

These derivations allow the *Dynamic* algorithm to determine the three main parameters online and adapt to changes in the network. The diagram shown in Figure 6.4a depicts how node types can change during the execution of *Dynamic* and an ideal spread of fake sources for *Dynamic* is shown in Figure 6.5a. However, as the number

of messages sent is high for this technique there is a desire to optimise it.

6.4 *DynamicSPR*: Fake Source Allocation Strategy

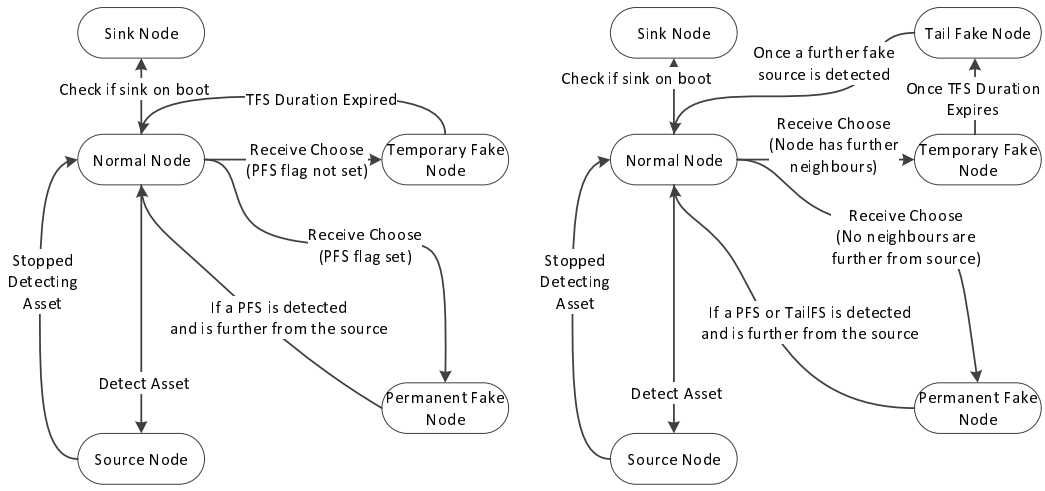
The *Dynamic* heuristic (and thus *Static*) works in such a way that fake sources spread out across part of the network. Nodes close to the source are prevented from becoming fake sources and nodes that are believed to be the furthest from the source become a PFS (see Figure 6.5a). The downside is that the many fake sources cause a high message overhead (i.e., use a large amount of energy) and may often undo the work of other fake sources or induce a high proportion of collisions. To circumvent this problem, an alternative technique for fake source selection — called *DynamicSPR* (dynamic single path routing algorithm) — will now be presented that makes use of the *directed random walk* technique from *phantom routing*. The routing protocol for ⟨normal⟩ messages remains unchanged for *DynamicSPR*, instead the directed random walk is used in fake source allocation. The benefit of the directed random walk is that it allocates a node far from the source, which is a good location for a PFS to be. Intuitively, the use of a directed random walk should prevent multiple competing fake sources from being created, allowing high levels of SLP to be provided whilst using fewer messages.

6.4.1 *DynamicSPR* Overview

Initially a wave of ⟨away⟩ messages are sent to inform nodes of their sink distance. When the sink receives a ⟨notify⟩ from the source, a ⟨choose⟩ message will be sent to start the fake source allocation. At the end of a fake node's duration the node then chooses a neighbour that has the furthest source distance and sends them a ⟨choose⟩ message. If there are multiple candidates, the next fake node will be chosen randomly from them. This is the main difference between *DynamicSPR* and *Dynamic*, because in *Dynamic* all neighbours would become fake sources. Along the walk TFSs will be created to pull the attacker away hop-by-hop. The walk will end when no neighbours are further from the source than the current node, this node will be a PFS. The state machine for *DynamicSPR* is depicted in Figure 6.4b. A typical execution of *DynamicSPR* will result in a fake source selection shown in Figure 6.5b.

6.4.2 Ensuring Reliability

It is unrealistic to assume that links between sensor nodes are reliable or that the links remain bi-directional [11]. Unreliable links become problematic when considering delivery of messages along single path routes, because if one message is lost it can prevent the single-path route from reaching its target. If a ⟨choose⟩ message fails to



(a) *Static* and *Dynamic* (b) *DynamicSPR*

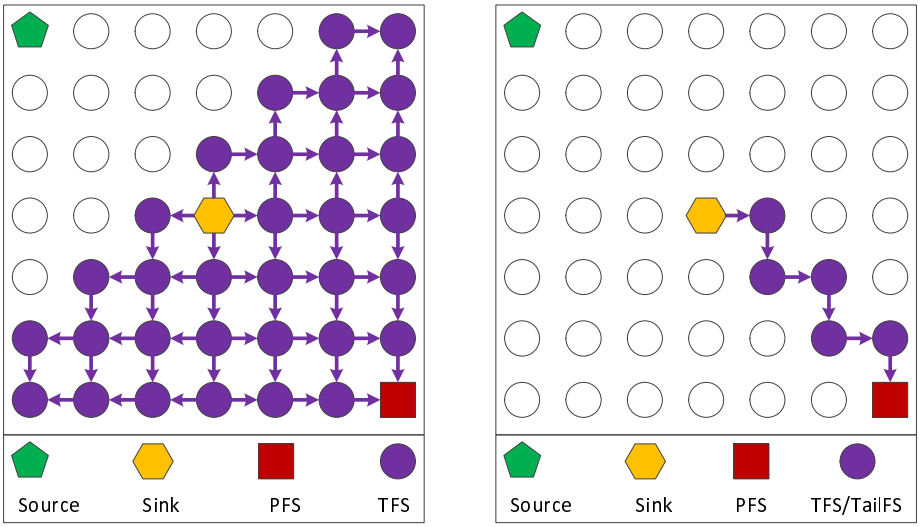
Figure 6.4: The conditions under which nodes transition from one type to another.

be delivered then the directed walk could terminate prematurely and no more SLP protection would be allocated. To ensure the walk is continued, a ⟨choose⟩ message needs to be retransmitted until it is acknowledged. However, if the fake source only sent ⟨choose⟩ messages until they are acknowledged, SLP will not be provided for a time as no ⟨fake⟩ messages are flooded through the network. To resolve this problem a new type of fake source — *tail* fake sources (TailFS) — will be created along the directed walk to continue providing SLP.

A TFS becomes a TailFS after its duration expires. While it is a TailFS, both ⟨choose⟩ and ⟨fake⟩ messages are sent periodically. As soon as a ⟨fake⟩ message is received from a fake source that is further from the real source than the TailFS is, the TailFS reverts to being a normal node and ceases sending ⟨choose⟩ and ⟨fake⟩ messages. In this case the ⟨fake⟩ message acts as an acknowledgement packet. Using this technique pairs of fake sources will be created along a walk, a TFS and a TailFS that is one-hop close to the source than the TFS is. As a TailFS needs to pull the attacker to its position while no TFSs exist, the fake broadcast period is set to be those used for TFSs[2]. The ⟨choose⟩ message is repeated every TFS duration until the TailFS becomes a Normal node.

$$P_{TailFS}(j) = P_{TFS}(j) \qquad (6.25) \qquad D_{TailFS}(j) = D_{TFS}(j) \qquad (6.26)$$

[2]Note that in previously published work [25] the period at which TailFSs broadcasted ⟨Fake⟩ messages was set to the PFS period $P_{PFS}(j)$. This was changed to the TFS period for performance reasons and also timing reasons which will be elaborated on in Chapter 7.



(a) *Static* and *Dynamic* (b) *DynamicSPR*

Figure 6.5: Best-case spread of fake sources for three different fake source algorithms.

6.4.3 Choosing The Next Fake Source

In order to choose the next fake node from the 1-hop neighbourhood, nodes need to keep their neighbour's updated. To do this every node periodically broadcasts a message informing neighbours of their id as well as important distance information that is used to make a decision on the next fake source. The next fake node in the directed random walk is chosen randomly from the set defined in Equation 6.27 and is allowed to have the same source distance as the current fake source. By reducing the strictness of the distance decision, the directed walk is allowed to go along a path of same-distance nodes which could potentially help the walk reach further from the source.

$$\textsc{CanBeFake}(j) = \{\, n \mid n \in \textsc{1HopN}(j) \wedge \Delta_{src}(n) \geq \Delta_{src}(j) \wedge n \neq sink \,\} \quad (6.27)$$

6.4.4 Unchanged Settings

The way D_{TFS}, P_{TFS}, P_{PFS} and $\#_{\mathcal{F}}(j)$ are calculated in *DynamicSPR* has not been changed from the *Dynamic* algorithm. In one sense, *DynamicSPR* can be considered a special instance of *Dynamic* where only a single path in the flood of fake nodes is active.

$$\mathcal{R}_{\mathcal{N}} = \begin{array}{c} \\ \text{Sending Node} \end{array} \begin{array}{c} \text{Receiving Nodes} \\ \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \end{array} \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.28}$$

$$\mathcal{R}_{\mathcal{F}_1}^{Dynamic} = \begin{array}{c} \\ \text{Sending Node} \end{array} \begin{array}{c} \text{Receiving Nodes} \\ \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.29}$$

$$\mathcal{R}_{\mathcal{F}_1}^{DynamicSPR} = \begin{array}{c} \\ \text{Sending Node} \end{array} \begin{array}{c} \text{Receiving Nodes} \\ \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 1 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \end{pmatrix} \tag{6.30}$$

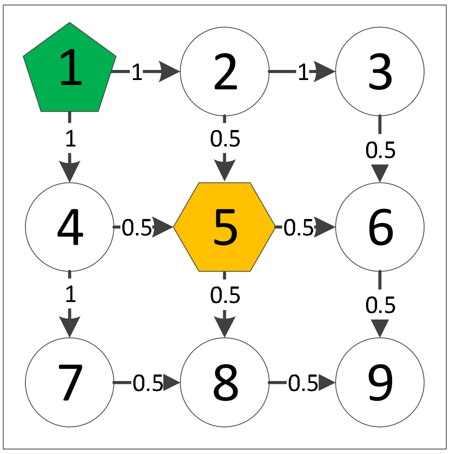
$$\mathcal{R}_{\mathcal{F}_2} = \begin{array}{c} \\ \text{Sending Node} \end{array} \begin{array}{c} \text{Receiving Nodes} \\ \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{6.31}$$

Figure 6.6: Routing matrices of *Protectionless* flooding and *Dynamic*'s two fake source positions

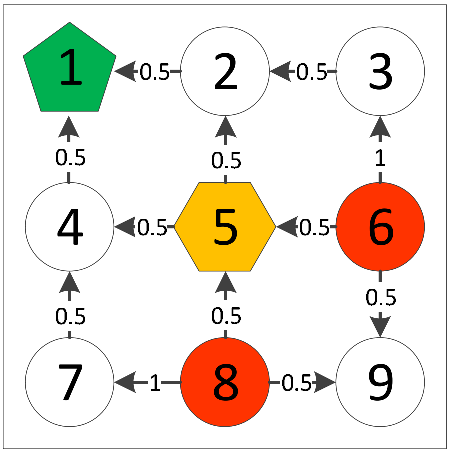
6.5 Competing Paths Analysis

Dynamic and *DynamicSPR* have an interesting structure when considered in terms of competing paths because rather than perturbing the original routing matrix additional routing matrices are added. So a sequence of routing matrices is used to model the routing protocol for *Dynamic* and *DynamicSPR*. The first routing matrix will represent flooding ⟨normal⟩ messages from the source node. Depending on the approach there will either be one, two or randomly one or two entries in the sequence of routing matrices that represent the flood of ⟨fake⟩ messages. This pattern will repeat with the routing matrices for ⟨fake⟩ messages changing as the fake sources change location.

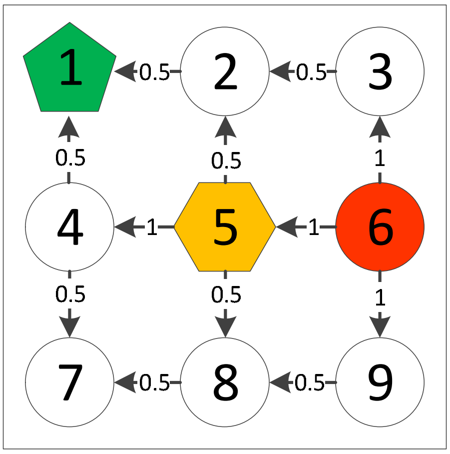
An example of the sequence of routing matrices with the approach set to `Fixed1` where 1 ⟨fake⟩ message is sent per source period is shown in Equation 6.32. Another example when the approach is `Fixed2` where 2 ⟨fake⟩ messages are sent per source period is shown in Equation 6.33. While these routing matrices are infinite in length only the finite number of entries before the safety period would be considered in the analysis. This would be the first $2P'_{safety}$ entries for `Fixed1` and $3P'_{safety}$ for `Fixed2`. The reason for this is that 2 routing matrices (one normal and one fake) would occur within the source period for `Fixed1` and 3 would occur for `Fixed2` (one



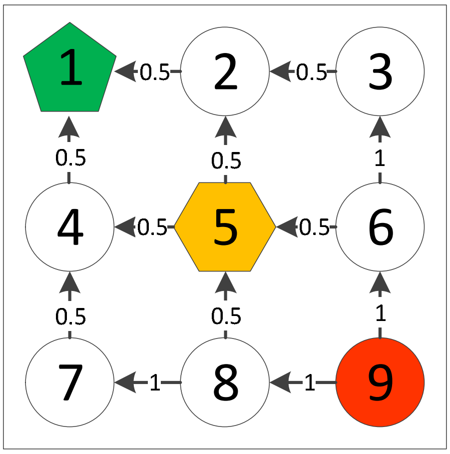
(a) Flooding: Equation 6.28



(b) *Dynamic* 1: Equation 6.29



(c) *DynamicSPR* 1: Equation 6.30



(d) *Dynamic* 2: Equation 6.31

Figure 6.7: Graphical representation of the *Protectionless* and two *Dynamic* and *DynamicSPR* routing matrices. With fake sources shown in red.

normal and two fake).

$$\mathcal{R}_1 = [\, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_1}, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_2}, \ldots, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_2} \,] \tag{6.32}$$

$$\mathcal{R}_2 = [\, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_1}, \mathcal{R}_{\mathcal{F}_1}, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_2}, \mathcal{R}_{\mathcal{F}_2}, \ldots, \mathcal{R}_{\mathcal{N}}, \mathcal{R}_{\mathcal{F}_2}, \mathcal{R}_{\mathcal{F}_2} \,] \tag{6.33}$$

In terms of competing paths, an advantage of *Dynamic* and *DynamicSPR* is that the two fake routing matrices compete at all nodes that an attacker is likely to be located at when considering the normal routing protocol. This means that there is a large number of opportunities for the attacker to be pulled away from the real source.

The information loss of this sequence of routing protocols is calculated below. In this example the safety period is 4, for the same reason in Chapter 4, because the

attacker would capture the source within 2 moves under *Protectionless* flooding $\mathcal{R}_{\mathcal{N}}$ and the safety factor is set to 2. The set of transitions remains the same when a safety period of 4 is used. As flooding is used as the base routing protocol it leaks maximal information to the attacker when ⟨normal⟩ messages are sent. This is because this matrix is exactly the same as the *Protectionless* routing matrix. However, the first and second ⟨fake⟩ message routing matrices provide high information loss of 75% and 100% respectively, as very few paths are shared with *Protectionless* flooding.

$$D_{\mathcal{N}}^{P'_{safety}} = D_{\mathcal{N}} = \begin{array}{l}\{\,(1,2),(1,4),(2,3),(2,5),(3,6),(4,5),\\ \quad (4,7),(5,6),(5,8),(6,9),(7,8),(8,9)\,\}\end{array} \tag{6.34}$$

$$D_{\mathcal{F}_1}^{P'_{safety}} = D_{\mathcal{F}_1} = \begin{array}{l}\{\,(2,1),(3,2),(4,1),(4,7),(5,2),(5,4),\\ \quad (6,3),(6,5),(6,9),(8,5),(8,7),(8,9)\,\}\end{array} \tag{6.35}$$

$$D_{\mathcal{F}_2}^{P'_{safety}} = D_{\mathcal{F}_2} = \begin{array}{l}\{\,(2,1),(3,2),(4,1),(5,2),(5,4),(6,3),\\ \quad (6,5),(7,4),(8,5),(8,7),(9,6),(9,8)\,\}\end{array} \tag{6.36}$$

$D_{\mathcal{N}}^{P'_{safety}}$	$(1,2)$	$(1,4)$	$(2,3)$	$(2,5)$	$(3,6)$	$(4,5)$	$(4,7)$	$(5,6)$	$(5,8)$	$(6,9)$	$(7,8)$	$(8,9)$
$D_{\mathcal{N}}^{P'_{safety}}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$D_{\mathcal{F}_1}^{P'_{safety}}$	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓
$D_{\mathcal{F}_2}^{P'_{safety}}$	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 6.1: Are the transitions from $D_{\mathcal{N}}$ in $D_{\mathcal{N}}^{P'_{safety}}$, $D_{\mathcal{F}_1}^{P'_{safety}}$ and $D_{\mathcal{F}_2}^{P'_{safety}}$?

$$IL(D_{\mathcal{N}}, D_{\mathcal{N}}) = \frac{0+0+0+0+0+0+0+0+0+0+0+0}{12} = 0 \tag{6.37}$$

$$IL(D_{\mathcal{N}}, D_{\mathcal{F}_1}) = \frac{1+1+1+1+1+1+0+1+1+0+1+0}{12} = \frac{9}{12} = \frac{3}{4} \tag{6.38}$$

$$IL(D_{\mathcal{N}}, D_{\mathcal{F}_2}) = \frac{1+1+1+1+1+1+1+1+1+1+1+1}{12} = \frac{12}{12} = 1 \tag{6.39}$$

6.6 Results

In this section the results of the *DynamicSPR* protocol will be examined via a comparison between the *Dynamic* and *Static* SLP protocols plus the *Protectionless* protocol. The results presented were generated by experiments performed using the experimental setup outlined in Chapter 3. The *Dynamic* and *DynamicSPR* algorithms both have one parameter which is the approach it takes regarding the number of ⟨fake⟩ message sent per period, simulations for `Fixed1`, `Fixed2` and `Rnd`

are run. From the simulations various metrics about the performance of the *Dynamic* and *DynamicSPR* protocols were collected. The following metrics will be analysed in this section:

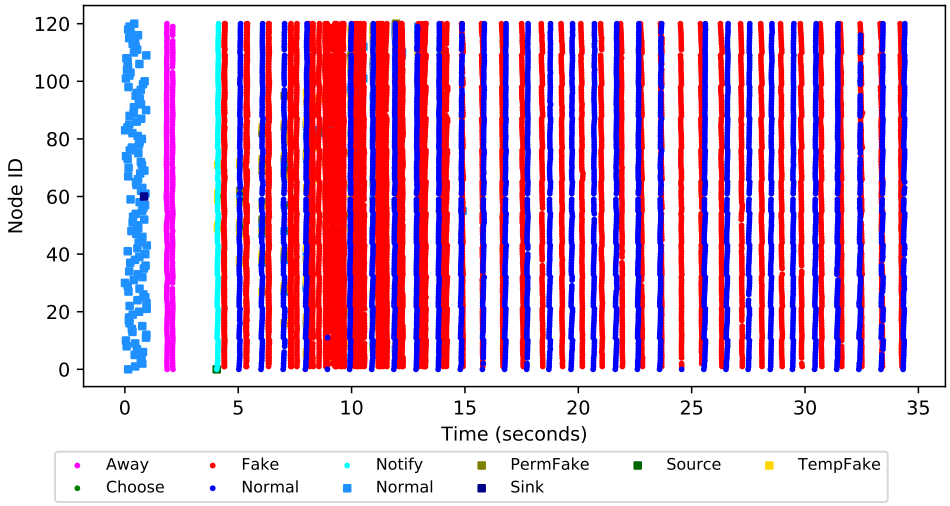
1. **Broadcast and Attacker Behaviour** — This is the patten in which messages are sent and the path the attacker takes in response.
2. **Received Ratio** — This is the percentage of messages that were sent by the source and received by the sink.
3. **Capture Ratio** — This is the percentage of runs in which the attacker reaches the location of the source, i.e., captures the source.
4. **Average Number of Fake Messages Sent** — This is the average number of ⟨fake⟩ messages sent across all nodes.
5. **Attacker Distance** — This is the average attacker distance from the source recorded at the end of a run.
6. **Latency** — This is the average time it takes a message sent by the source to be received at the sink.

Several graphs include values for *Baseline* results which are the results for *Protectionless* flooding under the same conditions as the results when SLP is provided. These results are included to put the overhead of running the fake source protocols into perspective.

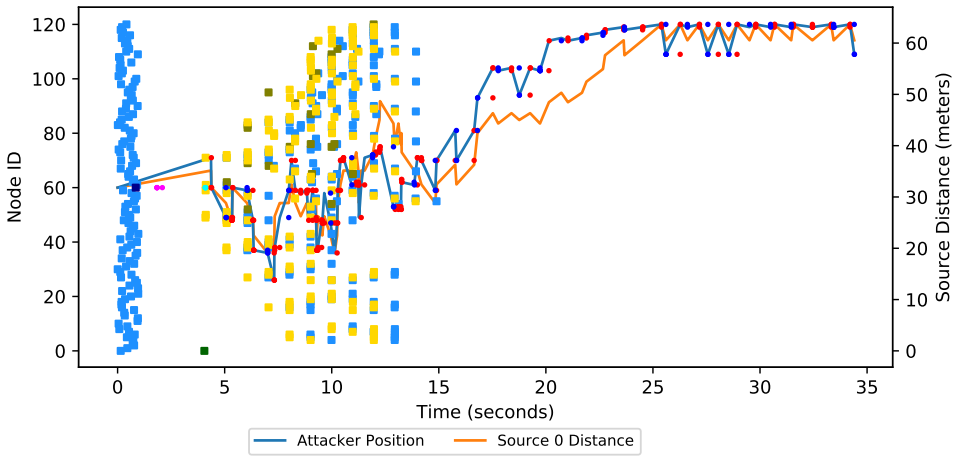
The results for *Static* are also included for some graphs to situate the performance of *Dynamic* and *DynamicSPR*. These simulation include all the parameter combinations that were used in [78]. These results are not an exhaustive search of the parameter space, but include values likely to be good. Conversely, these parameters also produced results that demonstrated poor performance.

6.6.1 Behaviour

A demonstration for one execution of the *Dynamic* and *DynamicSPR* protocols are shown in Figure 6.8 and Figure 6.9 respectively. Both are run using the `Fixed1` approach, where 1 ⟨fake⟩ message is sent over the TFS's duration. Both executions manage to pull the attacker away to a far location. In Figure 6.8a there is a large number of ⟨fake⟩ messages sent by the initial wave of fake sources in *Dynamic*. As the simulation progresses the ⟨fake⟩ messages sent reduces as permanent fake sources are allocated. This large block of ⟨fake⟩ messages is not presented in Figure 6.9a for *DynamicSPR* as fewer fake sources are present. The reduction in fake sources can be



(a) Messages Sent



(b) Attacker messages received with attacker position and source distance

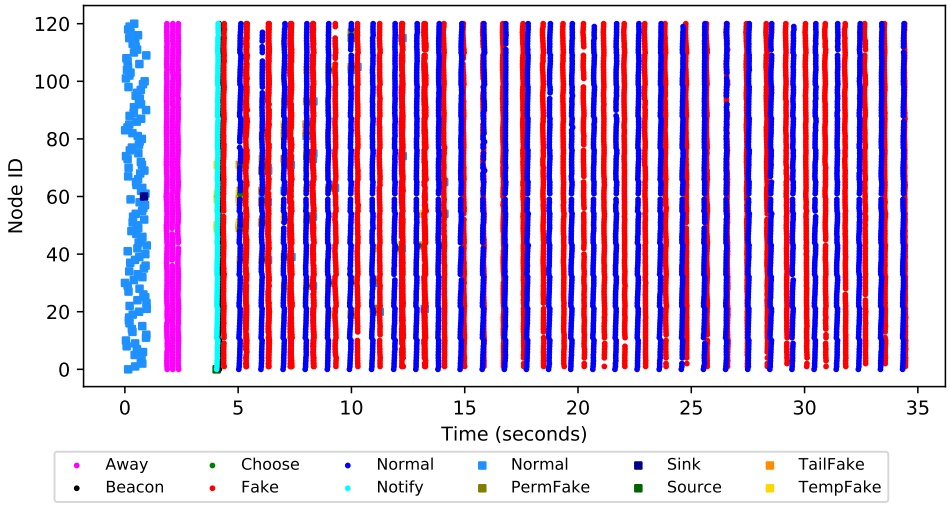
Figure 6.8: A demonstration of the behaviour of the *Dynamic* algorithm and how the attacker moves in response to the messages sent.

observed by comparing Figure 6.8b and Figure 6.9b, where *DynamicSPR* allocates fewer TFSs compared to *Dynamic*. This means that *DynamicSPR* has achieved its aim of allocating fewer fake sources compared to *Dynamic* in these two instances.

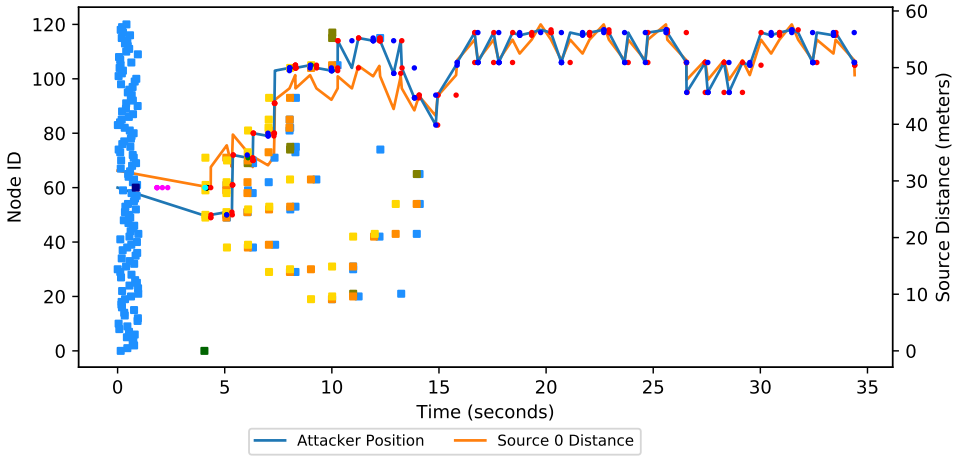
6.6.2 Received Ratio

The graphs in Figure 6.10 show that for slower broadcast rates (Figures 6.10a and 6.10b) the received ratio is at or near 100% for *Dynamic* and *DynamicSPR*. *Static* performs poorly in the worst cases due to a bad combination of parameters.

As the period between messages decreases (the ⟨normal⟩ message broadcast rate increases) the delivery ratio drops. *Protectionless* tends to outperform *Dynam-*



(a) Messages Sent

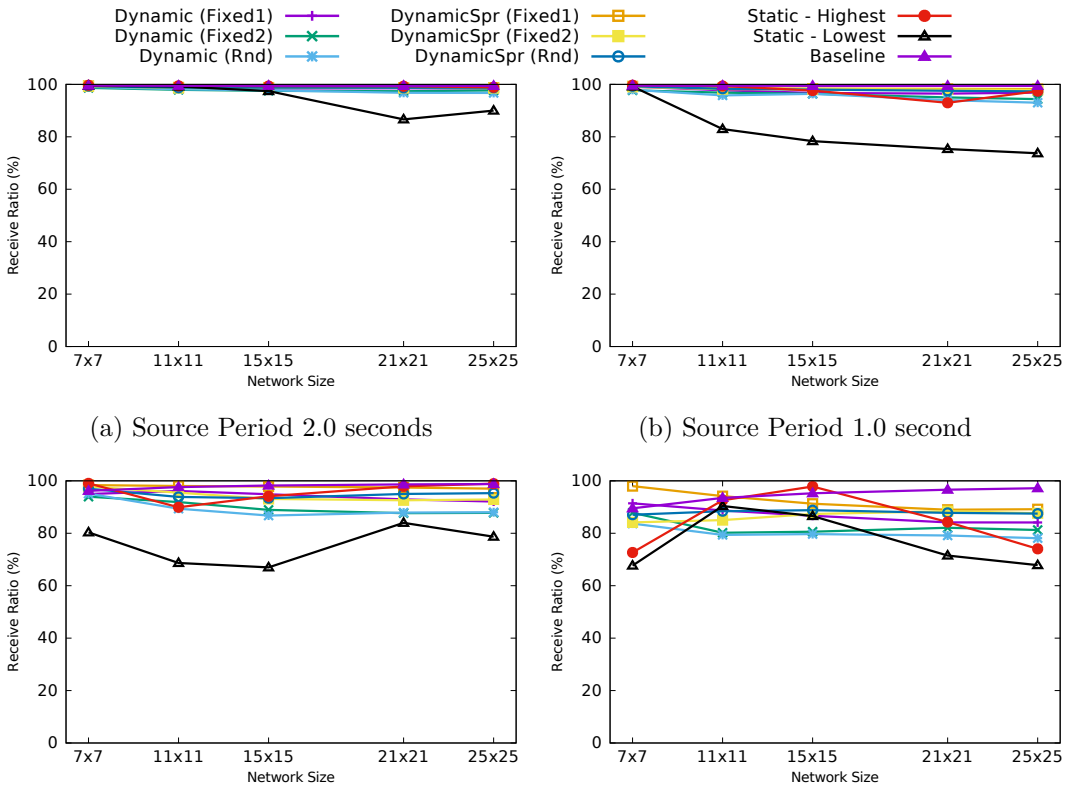


(b) Attacker messages received with attacker position and source distance

Figure 6.9: A demonstration of the behaviour of the *DynamicSPR* algorithm and how the attacker moves in response to the messages sent.

icSPR and *DynamicSPR* which tends to outperform *Dynamic*. This is due to an increase in collisions, as more ⟨normal⟩ messages are being sent for lower source periods which requires more ⟨fake⟩ messages to be sent. The increase in network traffic leads to more message collisions.

As the network size increases two trends can be observed. For *Protectionless* the delivery ratio tends to increase. This is due to the increase in links available to carry the message to the sink. However, for *Dynamic* and *DynamicSPR*, delivery ratio tends to decrease for larger network sizes. In these situations the availability of more links is not sufficient to account for the increased unreliability of transmission over the longer distance between source and sink.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

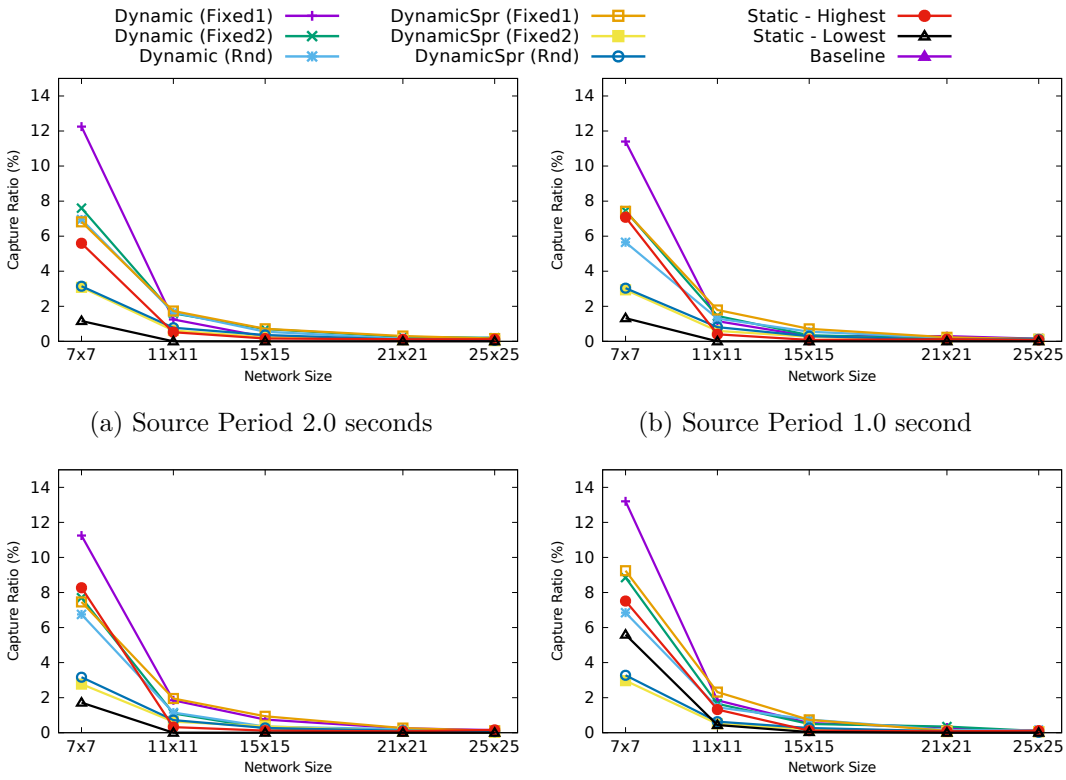
(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 6.10: Results showing the percentage of ⟨normal⟩ messages received at sink.

6.6.3 Capture Ratio

Figure 6.11 shows the capture ratio for the three configurations of the *Dynamic* and *DynamicSPR* protocols, as well as the best and worst *Static* protocol results. The capture ratio for the baseline case is not visible because it is 100% for all network sizes, and has been hidden by the scale of the graph. *DynamicSPR* provides a lower capture ratio compared to *Dynamic* across the four different source periods. On larger networks the capture ratio tends towards the optimal of 0%. This is a behaviour that has been previously observed [78] because larger networks have more space between the sink and source which gives the fake source algorithms chance to pull the attacker back if it makes headway towards the source. The results for *Static* show that good parameterisation can lead to very low capture ratios. But this is traded off with very high energy usage (in terms of the number of ⟨fake⟩ messages sent). *DynamicSPR* manages to achieve low capture ratios that approach *Static* for larger networks whilst requiring fewer ⟨fake⟩ messages to be sent as will now be shown.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

(c) Source Period 0.5 seconds

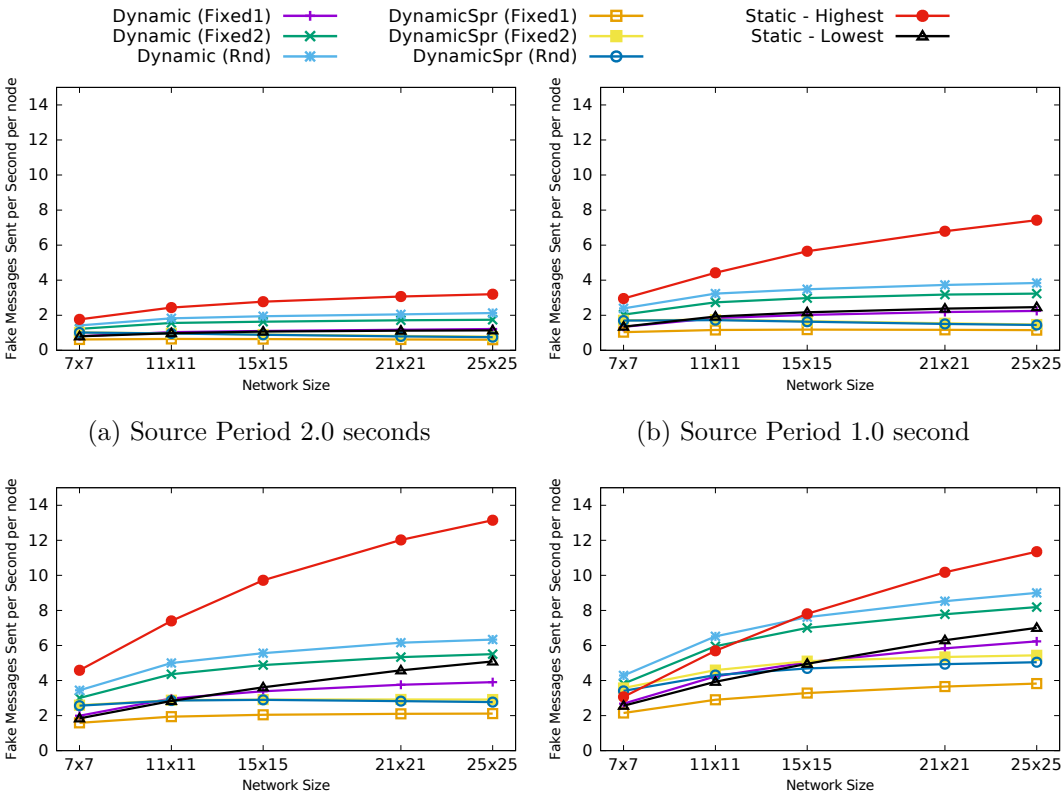
(d) Source Period 0.25 seconds

Figure 6.11: Results showing the capture ratio.

6.6.4 Average Number of Fake Messages Sent

The purpose of *DynamicSPR* was to reduce the message overhead of *Dynamic*. As sending and receiving messages tend to be among the most energy expensive activities performed by a WSN [118], the number of messages sent will be used as a proxy for energy consumption. The number of messages sent per node per second is used because these techniques tend to cause each node to send a similar number of messages over the safety period. However, different network sizes and different source periods have different safety periods, so the results are better compared by the number of messages sent per node per second.

The graphs in Figures 6.12 and 6.13 show that the *DynamicSPR* protocol tends to perform better than the *Dynamic* and *Static* protocols with respect to the number of messages sent per node per second. This reduction is due to *DynamicSPR* using a reliable directed random walk to allocate fewer fake sources compared to the *Dynamic* protocol that uses a controlled flooding approach. For smaller networks and faster source periods *Static* and *Dynamic* can perform better. But because of lower safety periods in these cases, *DynamicSPR*'s fake source allocation strategy is



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

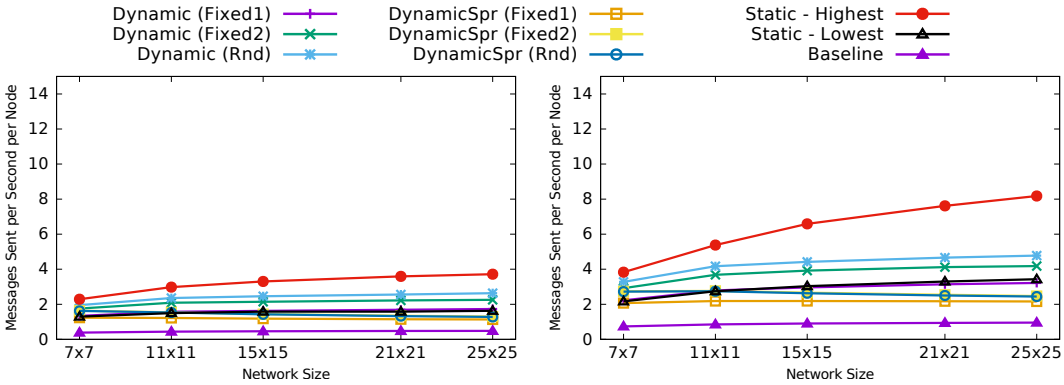
(c) Source Period 0.5 seconds

(d) Source Period 0.25 seconds

Figure 6.12: Results showing the average number of ⟨fake⟩ messages sent per node per second.

performed for less time, so the benefit of it is reduced.

There are some differences between the three *Dynamic* and *DynamicSPR* configurations. In both `Fixed2` causes more messages to be sent than `Fixed1`, however, `Rnd` sends nearly as many messages as `Fixed2` under *DynamicSPR* and actually sends more under *Dynamic*. This outcome for `Rnd` was unexpected, as it was expected to be in the middle of `Fixed1` and `Fixed2`. The reason for this behaviour is that different random numbers of ⟨fake⟩ messages to send causes different periods between the messages. This prevents the messages from interfering with each other, leading to fewer collisions and increasing the number of messages that propagate further. There is a higher impact for *Dynamic* as a large number of fake sources may be allocated. This greater number of messages sent is also reflected in the capture ratio which is lower for `Rnd` then `Fixed2` for *Dynamic*.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

(c) Source Period 0.5 seconds

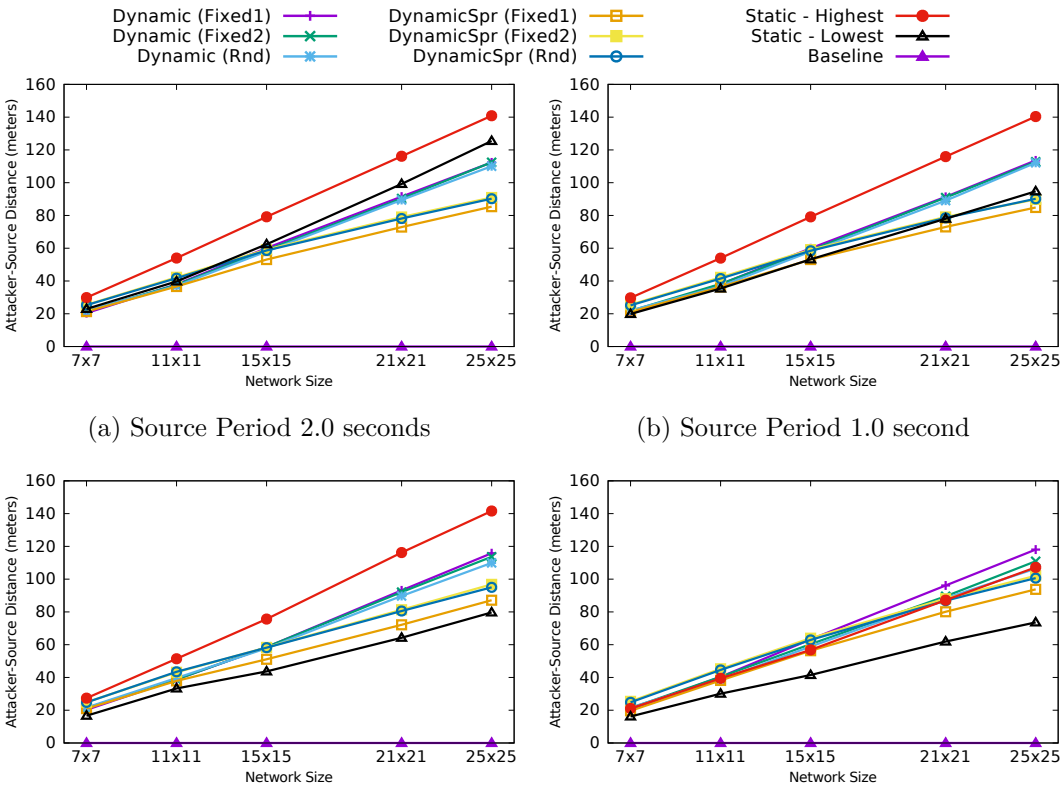
(d) Source Period 0.25 seconds

Figure 6.13: Results showing the average number of messages sent per node per second.

6.6.5 Attacker Distance

The attacker distance metric quantifies how good the SLP protocol is at pulling an attacker away from the source. Figure 6.14 shows that the *Dynamic* protocol is better able to pull the attacker further from the source compared to *DynamicSPR*. There is little difference between the different techniques used in the protocols. Although *DynamicSPR*'s `Fixed1` performs worse than the other two approaches. By only sending 1 ⟨fake⟩ message per ⟨normal⟩ message *Dynamic* is not capable of pulling the attacker away from the source, but can only keep the attacker at its current distance.

Both protocols perform better than baseline protectionless which failed to prevent the attacker from finding the source. The cause of the difference is that *Dynamic* is capable of allocating a PFS at a node further from the source than *DynamicSPR*. This is because *Dynamic* uses a controlled flooding of the network with TFSs which allows it to reach all nodes. In comparison, *DynamicSPR* uses a directed walk along the source distance gradient, which allows a local source distance



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

(c) Source Period 0.5 seconds

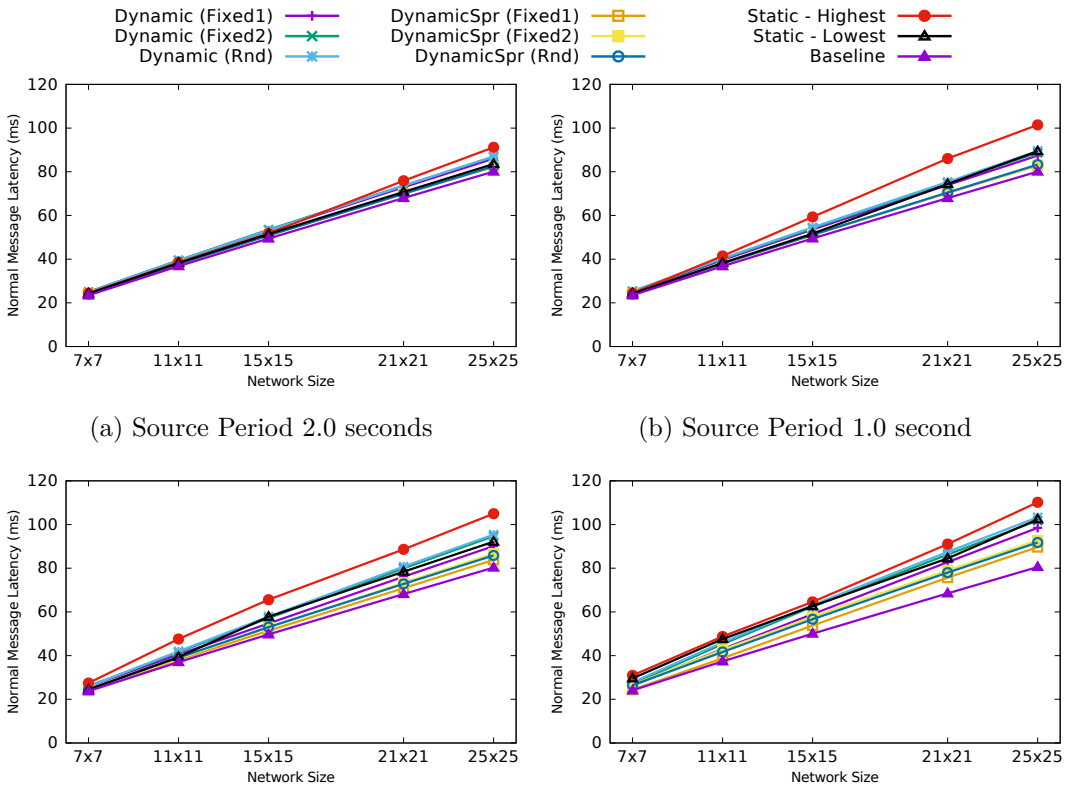
(d) Source Period 0.25 seconds

Figure 6.14: Results showing the attacker's distance from the source.

maxima to be found, but not necessarily the global maxima.

6.6.6 Normal Latency

The latency of ⟨normal⟩ messages (the time it takes to reach the sink after being broadcasted by the source) is shown in Figure 6.15. The latency is low, but increases as network size increases because the sink and source become further from each other. This is because the ⟨normal⟩ messages are sent via flooding and roughly follow the shortest path to the sink. Messages do not always follow the shortest path, as collisions or noise could mean a node does not receive a message. This is demonstrated by *Protectionless* having the lowest latency and the different fake source based SLP algorithms having higher latency. As the ⟨fake⟩ messages collide with ⟨normal⟩ messages the flood of ⟨normal⟩ messages cannot always continue along the shortest path. However, overall the latency is very low.



(a) Source Period 2.0 seconds

(b) Source Period 1.0 second

(c) Source Period 0.5 seconds

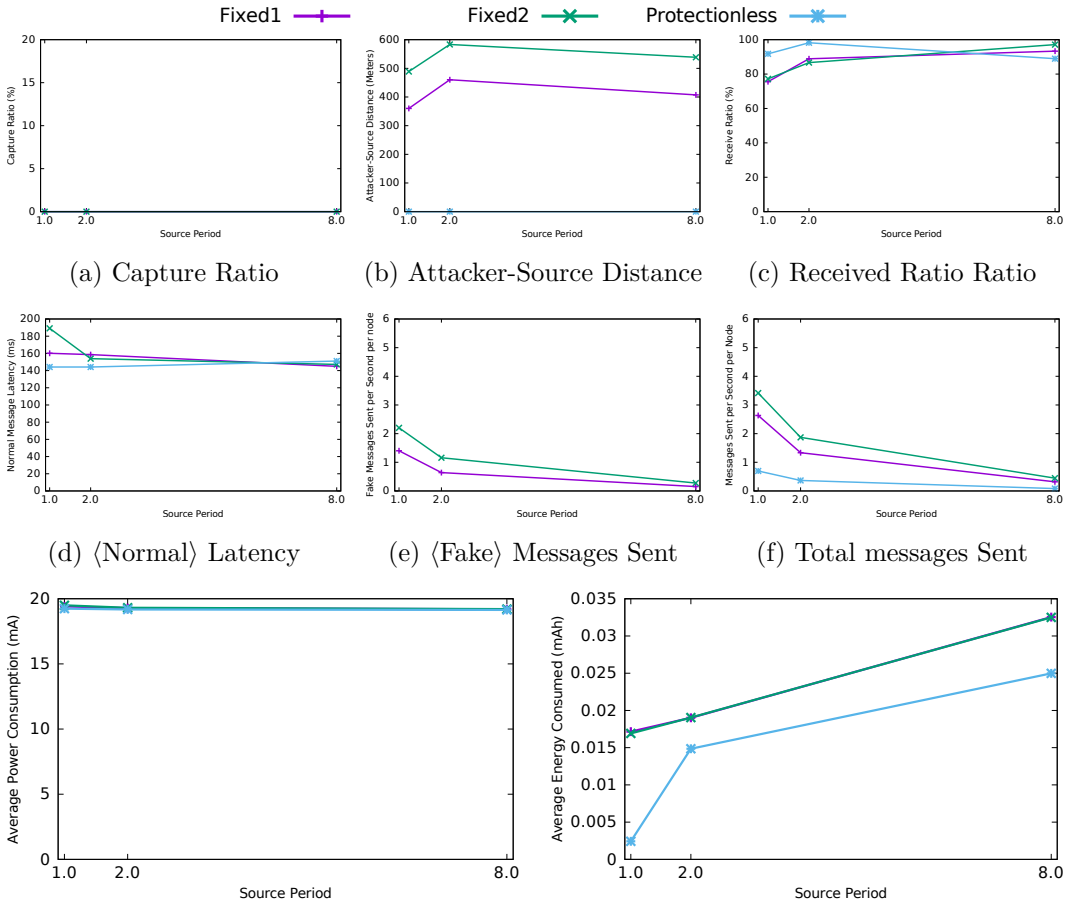
(d) Source Period 0.25 seconds

Figure 6.15: Results showing the time it took a normal message to reach the sink.

6.7 Testbed Results

This section will present the results gathered for *DynamicSPR* while running on the FlockLab testbed. Results are also included for *Protectionless* flooding. Only 8 repeats were performed for each parameter combination for *DynamicSPR* and 12 for *Protectionless* due to the time cost of obtaining the results. This means that the resolution of the capture ratio in Figure 6.16a is very low ($^1/_8 = 12.5\%$) compared to the sub-1% obtained by the larger number of repeats performed for the simulations. The implication of this is that the capture ratio may be higher than 0%, but will not be higher than 12.5%.

The received ratio in Figure 6.16c increases with longer source periods due to fewer collisions. Both `Fixed1` and `Fixed2` demonstrate similar received ratios. The different approaches were capable of pulling the attacker away from the source as shown in Figure 6.16b. `Fixed2` performed better in all cases as expected, due to the larger number of ⟨fake⟩ messages sent. In terms of latency, Figure 6.16d shows `Fixed1` performed better than `Fixed2` for fast message rates. The latency converged for smaller message rates. As shown in Figures 6.16e and 6.16f `Fixed1`



(a) Capture Ratio (b) Attacker-Source Distance (c) Received Ratio Ratio

(d) ⟨Normal⟩ Latency (e) ⟨Fake⟩ Messages Sent (f) Total messages Sent

(g) Average Power Consumption (h) Average Energy Consumed

Figure 6.16: FlockLab testbed results for *DynamicSPR*.

sends fewer messages than `Fixed2` and this number is also lower for longer source periods. Power and energy consumption are obtained from sensors on the testbed. A mostly flat power consumption is observed in Figure 6.16h. The energy consumption in Figure 6.16h increases with larger source periods as the experiment lasted longer and consumed more energy. These energy measurements will be analysed in more detail in Chapter 7. So overall the testbed results show very similar results to those obtained via simulations. This validates the performance results obtained from simulations of *DynamicSPR*, which showed that *DynamicSPR* is capable of providing near optimal levels of SLP.

6.8 Discussion

In this section implications of some of the assumptions made and their impact will be discussed.

6.8.1 Assumption of Link Directionality and Reliability

In the timing analysis presented in this chapter it is assumed that messages travel along the shortest path. This assumption will not always hold as links may become unidirectional, collisions may occur, or other link failures may occur which require either retransmission or route redirection. However, depending on the network topology, the difference in the time it takes a message to travel from source to sink is likely to be very small when link failures occur (as will be demonstrated in Figure 7.1). As the source period will be orders of magnitude greater than this travel time difference, its impact on how often to send messages will be small. In this timing analysis, the time taken to send a message α is disregarded due to its small relative size, so the small time variance caused by link failures can also be disregarded.

6.8.2 Application to Other Fake Source Techniques

This chapter focused on comparing *Dynamic* and *DynamicSPR* against *Static*, due to them using similar fake source allocation strategies to *Static*. However, performing a timing analysis is applicable to other fake source techniques to determine parameters because the three key parameters (permanent fake source period, temporary fake source period, fake source duration) investigated here are parameters that other techniques (such as [44, 115, 117]) will want to set. By determining what values they should use based on an analysis similar to the one presented in this chapter, these protocols can be made adaptive to changes in the network.

6.8.3 Application to Alternate Routing Protocols

One of the key assumptions in this chapter is the usage of a hop-based routing protocol. This is in contrast to other routing protocols based on the expected number of transmission [57], routing for maximum network lifetime [34], or geographical routing [29]. The reason for not routing based on the energy cost is because the physical location through which messages are transmitted is important for SLP-aware routing. As the transmission range is assumed to be similar, hop-based routing provides an approximation of geographical routing without needing GNSS hardware.

The equations that calculate parameters could use distance to derive the values, but as it ends up being eliminated, the dynamically determined parameters are applicable to energy or link-based routing protocols. However, the issue of where to allocate the fake sources and in which direction they should spread remains. For geographic routing, the location knowledge is available to allocate the fake sources in the correct direction. For energy and link-based routing, an option might be to

have nodes record if they have been involved with sending messages recently and then having fake sources avoid going near these paths. Such an allocation is unlikely to be as good as when considering distance or geographical location. Future work should investigate using such routing protocols.

6.9 Alternate Fake Source Movement Techniques

In this chapter both fake source techniques aim to provide SLP by luring the attacker to another part of the network that is away from the real source. To do this the fake sources either spread out in a wave or as part of a directed random walk away from the sink. These fake sources may wish to move in an alternate pattern. For example, the fake source may wish to mimic the paths that assets take. In [123] Source Simulation was proposed in which multiple *fake* traces of asset movement were created in the network. The movement performed aimed to simulate that of a real asset in order to provide SLP against an attacker with global visibility. The SLP provided is suboptimal for a global attacker on purpose as a way to trade-off lower SLP provision for lower energy usage.

If *DynamicSPR*'s fake source allocation strategy was changed from moving away from the sink to following a path that simulates the movement of the asset, then it could be used as a way to dynamically determine the parameters for Source Simulation. If multiple traces can be created then, *DynamicSPR* would also be able to provide some level of SLP against a global attacker as well as a local attacker. This means that Source Simulation can be viewed as an instantiation of *DynamicSPR* augmented with a fake source movement strategy. The advantage of this is that the timing analysis performed for *DynamicSPR*, can also be used for Source Simulation.

6.10 Conclusion

In this chapter two online fake source-based protocols have been presented, namely *Dynamic* and *DynamicSPR*, that provide SLP. These protocols are based on the *Static* heuristic, which was proposed to address the intractable nature of SLP. A timing analysis framework for estimating SLP-relevant parameters has been developed for fake-source based techniques. *Dynamic* was shown to provide high levels of SLP at the expense of a high message overhead. To circumvent this problem, *DynamicSPR* was developed to allocate fewer fakes sources by using a directed random walk. The results show that, in general, the SLP levels provided by *DynamicSPR* are low like those of *Static* and *Dynamic*, whilst being more energy efficient. This makes *Dynamic* and *DynamicSPR* scalable and suitable for WSN deployments that require SLP.

Chapter 7

Duty Cycling for Fake Source Routing

Throughout this thesis there has been much emphasis on reducing the number of messages sent as a proxy for reducing the energy usage of the protocol. This is useful as sending and receiving messages tend to be the most expensive tasks performed by a sensor node [118]. However, a fixed energy cost per send or receive action does not take into account the cost of leaving the radio on all the time (and the CPU along with the radio) in order to listen to messages sent by neighbouring nodes. To save energy sensor nodes tend to spend the majority of their lifetime asleep [65] and only periodically wakeup, a technique known as *duty cycling*. This technique is important enough to justify preventing attacks that prevent nodes from entering sleep [26]. An issue with duty cycling is that while a node is asleep it cannot receive messages, this means that there is a need for a wakeup strategy to ensure it is ready to receive messages. When providing SLP using specific messages that are depended upon to lure the attacker away from the source, special care needs to be made to ensure these message are not missed due to the node being asleep.

This chapter will present the first investigation into duty cycling the radio to save energy, whilst ensuring a high level of SLP being provided by the previously developed protocols. To achieve this the timing analysis performed in Chapter 6 for *DynamicSPR* will be extended to apply to when messages are expected to be sent and received. This information will then be used on a per node basis to predict when future messages will be sent by neighbours. To achieve reliability the nodes will wake up early and go to sleep late around the time a message is expected to be receive. ⟨Fake⟩ messages are prioritised by having a larger wakeup period. Simulation results show that a wakeup strategy based on this timing analysis can lead to large reductions in the time network nodes spend awake and performing idle listening.

A duty cycle of 45% is achievable for the fast message rate of 1 message every 0.5 seconds. Longer source periods produce lower duty cycles, with 10% achievable when 1 message is sent every 8 seconds. This lower duty cycle leads to a reduction in the energy cost of the *DynamicSPR* routing protocol, making it more practical for deployment.

7.1 Duty Cycle Timing Analysis

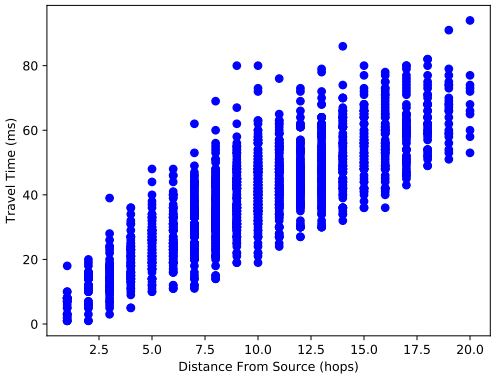
A problem with asynchronous duty cycle approaches (such as TinyOS's LPL) is that they require the routing protocols to retransmit messages if acknowledgements are not received. Because *DynamicSPR* does not use retransmissions to provide reliability (and instead relies on the extra paths provided via flooding) duty cycle techniques that use retransmissions are not suitable. So in order to reduce the energy cost of using *DynamicSPR* a synchronous duty cycle technique will be developed based on extending the timing analysis developed in Chapter 6. By allowing nodes to predict when their neighbours will send messages they can then schedule their wakeups accordingly. There are two components that need to be implemented, one that focuses on the ⟨Normal⟩ messages and another that focuses on the protection provided with ⟨Fake⟩ and ⟨Choose⟩ messages.

An important caveat is that all of these timings need to be deterministic. This means that some previous definitions of *DynamicSPR* that were left unspecified now need to be defined. Also, other parameters that involve a random component cannot be used. For example, the number of ⟨fake⟩ messages to send $\#_{\mathcal{F}}(j)$ under the `Rnd` strategy (1 or 2 messages per the TFS duration), cannot be used.

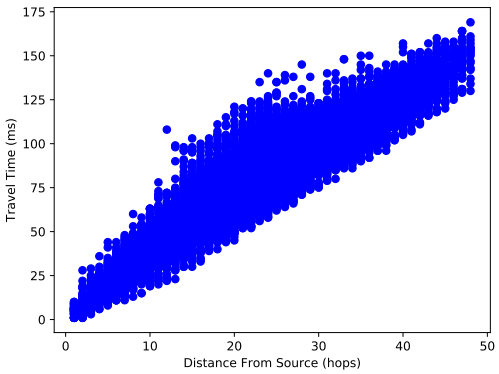
Another important aspect of this timing analysis is that it will occur locally to each node. This means that there is no need for time synchronisation across the network and no need for extra protocol messages to disseminate timing information. It is critical that this analysis does not add much computation overhead, otherwise the expected broadcast times will not match the actual broadcast times which will cause the duty cycle to go out of sync.

7.1.1 ⟨Normal⟩ Message Timings

This section details when a node should wakeup and to go sleep to ensure delivery of ⟨normal⟩ messages. This timing analysis is based on the one performed in Chapter 6, certain aspects of it will be repeated here for clarity.



(a) Size 11×11



(b) Size 25×25

Figure 7.1: The time it takes ⟨normal⟩ messages to travel from the source to various nodes under *Protectionless* flooding on TOSSIM in the SourceCorner configuration

- A node j will receive $\mathcal{N}_i$ at the time $R_j(\mathcal{N}_i)$.

$$R_j(\mathcal{N}_i) = S_{src}(\mathcal{N}_i) + \alpha\Delta_{src}(j) \tag{7.1}$$

- The time between a node j receiving $\mathcal{N}_i$ and $\mathcal{N}_{i+1}$ is $\hat{R}_j(\mathcal{N}_i)$:

$$\begin{aligned}
\hat{R}_j(\mathcal{N}_i) &= R_j(\mathcal{N}_{i+1}) - R_j(\mathcal{N}_i) \\
&= (S_{src}(\mathcal{N}_{i+1}) + \alpha\Delta_{src}(j)) - (S_{src}(\mathcal{N}_i) + \alpha\Delta_{src}(j)) \\
&= S_{src}(\mathcal{N}_{i+1}) - S_{src}(\mathcal{N}_i) \\
&= (i+1-1)P_{src} - (i-1)P_{src} \\
&= P_{src}
\end{aligned} \tag{7.2}$$

Due to unreliability in the network, messages will not always arrive exactly at this time. Figure 7.1 shows the spread in the amount of time it takes a ⟨normal⟩ message to travel from the source to an arbitrary node in the network a specific number of hops away from it. In some cases there can be up to 80 ms difference in the time it takes a node to receive a message. Therefore, it is important to wake up early $W_e(\mathcal{N})$ and sleep later $W_l(\mathcal{N})$ around the expected delivery time of the next ⟨normal⟩ message. The times that the node j is asleep and awake are defined by:

$$awake(\mathcal{N}_1) = \begin{bmatrix} \textbf{from} & 0 \\ \textbf{to} & R_j(\mathcal{N}_1) + W_l(\mathcal{N}) \end{bmatrix} \tag{7.3}$$

$$awake(\mathcal{N}_i) = \begin{bmatrix} \textbf{from} & \textbf{end}(asleep(\mathcal{N}_{i-1})) \\ \textbf{to} & \textbf{end}(asleep(\mathcal{N}_{i-1})) + W_e(\mathcal{N}) + W_l(\mathcal{N})) \end{bmatrix} \tag{7.4}$$

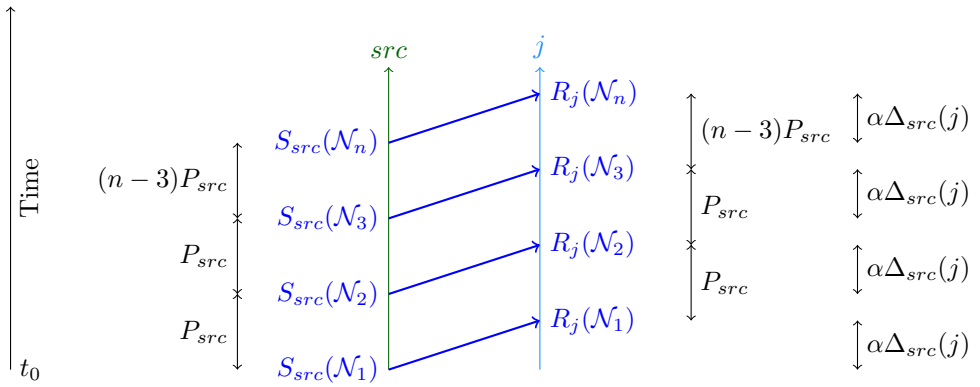


Figure 7.2: Timing of a source src sending ⟨Normal⟩ messages $\mathcal{N}_n$ to node j.

$$asleep(\mathcal{N}_i) = \begin{bmatrix} \textbf{from} & \textbf{end}(awake(\mathcal{N}_i)) \\ \textbf{to} & \textbf{end}(awake(\mathcal{N}_i)) + \hat{R}_j(\mathcal{N}_i) - W_e(\mathcal{N}) - W_l(\mathcal{N})) \end{bmatrix} \tag{7.5}$$

A diagram of the timing of events when sending ⟨normal⟩ messages is shown in Figure 7.2. The source node is seen to send ⟨normal⟩ messages to the normal node j every P_{src}. The blue arrows from the green source node to the light blue normal node represent the ⟨normal⟩ messages being sent. Node j will receive a ⟨normal⟩ message $\alpha\Delta(src, j)$ time units after it is sent, as this is the amount of time it takes messages to travel through the network. Once node j receives a ⟨normal⟩ message m at $R_j(\mathcal{N}_m)$ it knows that the next ⟨normal⟩ message $m+1$ will arrive at $R_j(\mathcal{N}_m) + P_{src}$. Between this time node j can sleep.

7.1.2 ⟨Fake⟩ and ⟨Choose⟩ Timings

For ⟨fake⟩ and ⟨choose⟩ messages, there is additional complexity to consider compared to ⟨normal⟩ messages that needs to be handled. After a node becomes a fake source there is a short delay before it starts to send messages $I_{TFS}(j)$ to help space the ⟨fake⟩ messages out. This value was previously undefined for *Dynamic* and *DynamicSPR*, but now needs to be be deterministically specified. To simplify the implementation it is assumed to be the same constant for all nodes.

$$I_{TFS}(j) = \frac{P_{TFS}(j)}{4} \tag{7.6}$$

- A TFS j will send its n^{th} ⟨fake⟩ message $\mathcal{F}_n^{TFS_j}$ during its duration at the following times:

$$S_{TFS_j}(\mathcal{F}_n^{TFS_j}) = \tau_{TFS}(j) + I_{TFS}(j) + (n-1)P_{TFS}(j) \tag{7.7}$$

- The set of all times ⟨fake⟩ messages will be sent by TFS j is:

$$S_{TFS_j}(\mathcal{F}^{TFS_j}) = \left\{ S_{TFS_j}(\mathcal{F}_n^{TFS_j}) \mid n \in 1..\#_{TFS}(j) \right\} \tag{7.8}$$

- Thus a node k is expected to receive ⟨fake⟩ messages from the TFS at node j at the following times:

$$R_k(\mathcal{F}^{TFS_j}) = \left\{ t + \alpha\Delta(k,j) \mid t \in S_{TFS_j}(\mathcal{F}^{TFS_j}) \right\} \tag{7.9}$$

- The time between two fake nodes j and k sending their first ⟨fake⟩ message, where j precedes k will be:

$$\begin{aligned} B(j,k) &= S_j(\mathcal{F}_1^{TFS_k}) - S_i(\mathcal{F}_1^{TFS_j}) \\ &= (\tau_{TFS}(k) + I_{TFS}(k) + (1-1)P_{TFS}(k)) - \\ &\quad (\tau_{TFS}(j) + I_{TFS}(j) + (1-1)P_{TFS}(j)) \\ &= \tau_{TFS}(k) + I_{TFS}(k) - \tau_{TFS}(j) - I_{TFS}(j) \\ &= (\tau_{TFS}(j) + D_{TFS}(j) + \alpha) + I_{TFS}(k) - \tau_{TFS}(j) - I_{TFS}(j) \\ &= D_{TFS}(j) + \alpha + I_{TFS}(k) - I_{TFS}(j) \end{aligned} \tag{7.10}$$

As a TFS at j moves to k its distance in hops with respect to another node i can change. This change will be mostly positive with respect to the source as the fake sources move away from the real source during the directed random walk, and there will be no change once a PFS has been allocated. However, the fake sources will get closer to other nodes in the network as they move away from the source. This means that the duty cycling needs to be able to handle messages arriving earlier and later than at the expected arrival time.

$$\Delta_i(TFS_k) - \Delta_i(TFS_j) = -1 \text{ or } 0 \text{ or } +1 \tag{7.11}$$

TailFS Timings

A TFS j becomes a TailFS after its duration expires to ensure a reliable progression of fake sources.

$$\tau_{TailFS}(j) = \tau_{TFS}(j) + D_{TFS}(j) \tag{7.12}$$

A TailFS will send a ⟨fake⟩ message with same period and duration as a TFS after the same initial start delay. Essentially a TailFS is a TFS, but without a fixed duration. Once the duration period expires it will send another ⟨choose⟩ message

to try to select the next TFS, but will remain a TailFS. Once a TailFS receives a message from a further TailFS or PFS, it will cease broadcasting ⟨fake⟩ messages and return to being a normal node.

$$D_{TailFS}(j) = D_{TFS}(j) \tag{7.13}$$

$$P_{TailFS}(j) = P_{TFS}(j) \tag{7.14}$$

$$I_{TailFS}(j) = I_{TFS}(j) \tag{7.15}$$

PFS Timings

- The time at which the TFS j hands off to a PFS k is shown in Equation 7.16. The α component represents the time cost of sending a ⟨choose⟩ message from TFS j to the node k that becomes the PFS.

$$\tau_{PFS}(k) = \tau_{TFS}(j) + D_{TFS}(j) + \alpha \tag{7.16}$$

- The PFS k also has an initial start delay $I_{PFS}(k)$. To simplify the implementation this is set to the same constant used by both TFSs and TailFSs:

$$I_{PFS}(k) = I_{TailFS}(k) = I_{TFS}(k) = \frac{P_{TFS}(k)}{4} \tag{7.17}$$

- A PFS k will send its n^{th} ⟨fake⟩ message $\mathcal{F}_n^{PFS_k}$ at the following times:

$$S_{PFS_k}(\mathcal{F}_n^{PFS_k}) = \tau_{PFS}(k) + I_{PFS}(k) + (n-1)P_{PFS}(k) \tag{7.18}$$

- As the duration of a PFS is potentially unbounded, the set of all ⟨fake⟩ messages that the PFS k could send is:

$$S_{PFS_k}(\mathcal{F}^{PFS_k}) = \left\{ S_{PFS_k}(\mathcal{F}_n^{PFS_k}) \mid n \in \mathbb{N}_1 \right\} \tag{7.19}$$

The time between each ⟨fake⟩ message from a PFS is its period $P_{TFS}(k)$. An issue is that the PFS period adjusts itself based on the ratio of ⟨fake⟩ messages the source has received ($\psi_{src}(\mathcal{F})$). The network is not expected to know when this change will occur or what the new period will be set to. Instead the change in the PFS period is required to occur slowly over time in order to allow the duty cycle to catch a ⟨fake⟩ message sent using the new period.

Sleep Schedule

Using these timings a sleep schedule can be developed for ⟨fake⟩ and ⟨choose⟩ messages. As with ⟨normal⟩ messages the nodes will wake up earlier and sleep later for both message types. This early and late wakeup will also help account for the changing PFS period. The early and late wakeups for ⟨choose⟩ messages will be represented as $W_e(\mathcal{C})$ and $W_l(\mathcal{C})$. The early and late wakeups for ⟨fake⟩ messages will be represented as $W_e(\mathcal{F})$ and $W_l(\mathcal{F})$. There are four timing aspects that need to be kept track of:

1. When the next ⟨fake⟩ message for the same TFS will be sent (considers the TFS period)
2. When the next TFS will send its first ⟨fake⟩ message (considers the TFS duration)
3. When a neighbouring TFS will send its ⟨choose⟩ message (considers the TFS duration)
4. When a PFS will send its next ⟨fake⟩ message (considers the PFS period)

After receiving the n^{th} ⟨fake⟩ message from j at node k the following knowledge about future message receives is known:

- The next ⟨fake⟩ message $n+1$ will be received at

$$R_k(\mathcal{F}_{n+1}^{TFS_j}) = R_k(\mathcal{F}_n^{TFS_j}) + P_{TFS}(j) \tag{7.20}$$

  and every subsequent ⟨fake⟩ message from this TFS each $P_{TFS}(j)$ after.

- If the node did not receive the first ⟨fake⟩ message from a TFS for some reason, the time it should have been received can be calculated by:

$$R_k(\mathcal{F}_1^{TFS_j}) = R_k(\mathcal{F}_n^{TFS_j}) - (n-1)P_{TFS}(j) \tag{7.21}$$

- The ⟨choose⟩ from TFS j at:

$$R_k(\mathcal{C}) = R_k(\mathcal{F}_1^{TFS_j}) + D_{TFS}(j) - I_{TFS}(j) \tag{7.22}$$

- The first ⟨fake⟩ message from the next TFS q is received at k at:

$$R_k(\mathcal{F}_1^{TFS_q}) = R_k(\mathcal{F}_1^{TFS_j}) + D_{TFS}(j) - I_{TFS}(q) + I_{TFS}(j) + \alpha \tag{7.23}$$

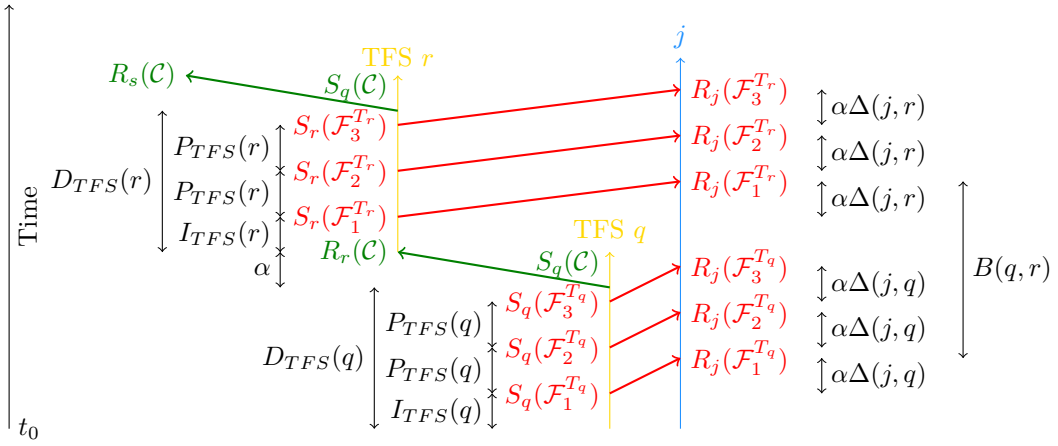


Figure 7.3: Timing of events for two TFS q and r when $\#_{TFS}(j) = \#_{TFS}(k) = 3$

As the initial delay is assumed to be a constant and $I_{TFS}(q) = I_{TFS}(j)$, this simplifies to:

$$R_k(\mathcal{F}_1^{TFS_q}) = R_k(\mathcal{F}_1^{TFS_j}) + D_{TFS}(j) + \alpha \tag{7.24}$$

7.1.3 Timing Demonstration

To aid understanding the times at which events can occur, this section will explain a number of timing diagrams. These diagrams aim to show the standard cases that can occur, plus error cases and how the protocol handles these.

In Figure 7.3 the timing of events for when a TFS q hands off to a TFS r is shown. The subsequent conversion of TFS q into TailFS q is omitted to keep the diagram simpler. The diagram shows how ⟨fake⟩ messages are sent periodically after some initial start delay ($I_{TFS}(q)$). The ⟨fake⟩ messages are represented by red arrows from the TFS q or TFS r (in yellow) to the normal node j (in light blue). The normal node j will receive messages from TFS q with the same period between the messages ($\alpha\Delta(j,q)$). However, there is a gap when transitioning from TFS q sending, to TFS r. This is down to the time cost of sending the ⟨choose⟩ message (α) plus the initial start delay ($I_{TFS}(r)$). Node j works out when to wake up for TFS r's ⟨fake⟩ messages by considering the TFS duration.

The diagram in Figure 7.4 is very similar to Figure 7.3 except that TFS q hands off to a PFS r via the ⟨choose⟩ message in green. Node j can still work out when to expect PFS r's first message using the duration of TFS q because $I_{TFS}(q) = I_{PFS}(r)$.

The timing diagram in Figure 7.5 includes TFS q becoming a TailFS and

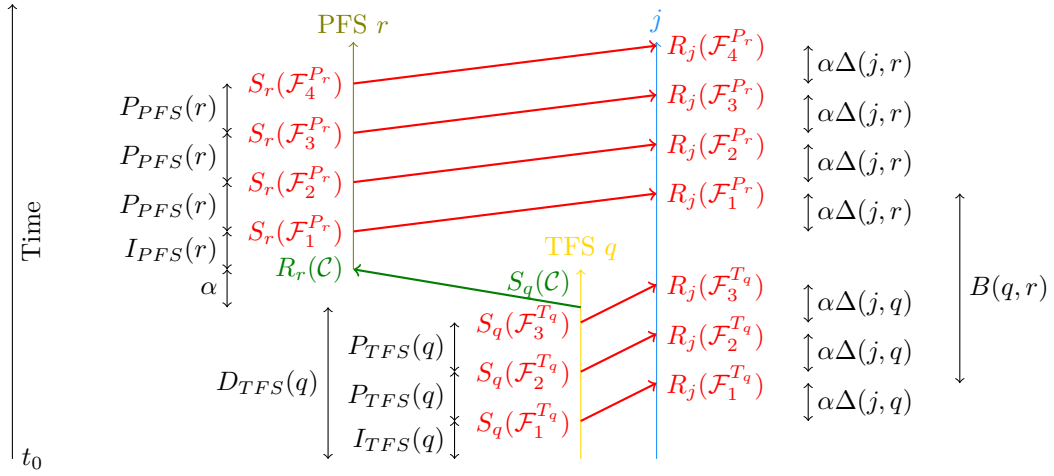


Figure 7.4: Timing of events for a TFS q and a PFS r when $\#_{TFS}(q) = 3$

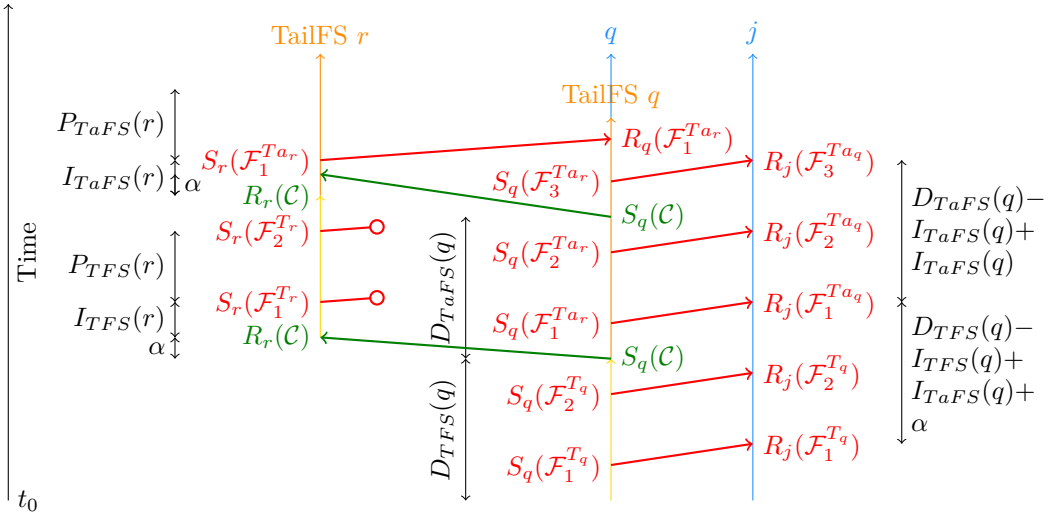
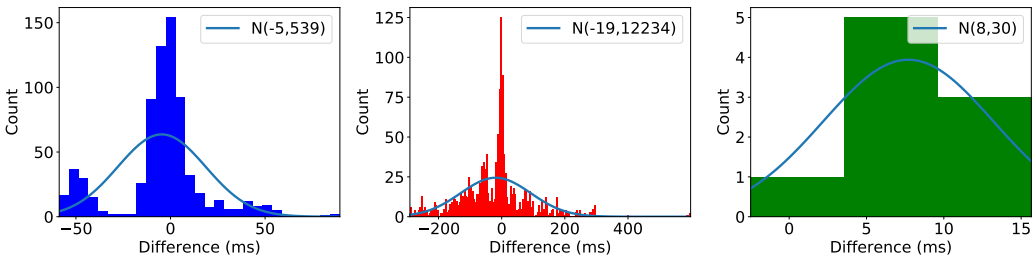


Figure 7.5: Timing of events for a TFS q that becomes a TailFS q when $\#_{TFS}(q) = 2$ and q fails to receive any ⟨fake⟩ messages from r until its second duration

shows how it remains a TailFS until it detects a ⟨fake⟩ message from a further node r. In this case there are 2 ⟨fake⟩ messages sent per period. In the first period of TFS r the two messages get lost, perhaps due to collisions, high noise, or some other error. These lost messages are represented by lines that terminate with a circle. When TFS r becomes a TailFS it continues broadcasting at the same period after the initial start delay because $P_{TFS}(q) = P_{TailFS}(q)$. TailFS q then receives this message and becomes a normal node as it has detected a further TailFS to take over its role. In this case the second round of ⟨fake⟩ messages are sent slightly earlier as there is no



(a) ⟨Normal⟩ (b) ⟨Fake⟩ (c) ⟨Choose⟩

Figure 7.6: Receive time difference from the expected receive time when the early and late wakeups are large.

delay from sending a ⟨choose⟩ message. As this time is expected to be small it will be handled by the early and late wakeups for ⟨fake⟩ messages. Note that in this scenario a second ⟨choose⟩ message was sent by TailFS q as at that point it did not believe that a further fake node had been created.

7.2 Algorithm

This section will describe how the *DynamicSPR* algorithm was adjusted to implement duty cycling. An important aspect of these changes are the early and late wakeups for ⟨normal⟩, ⟨fake⟩ and ⟨choose⟩ messages and what values should be used. So before the algorithm implementation is described, the values these wakeups should take are analysed. An investigation into the spread of when messages are received will now be performed.

7.2.1 Early and Late Wakeups

It is to be expected that there will be some difference with respect to when messages are expected to be received. For example, messages may not take the shortest path through the network due to collisions, or computation on one node may take longer than expected leading to a delay in broadcasting the message. This is why early wakeup ($W_e(\mathcal{M})$) and late wakeup ($W_l(\mathcal{M})$) times are used. Different wakeup times can be specified for ⟨normal⟩ message ($\mathcal{N}$), ⟨fake⟩ messages ($\mathcal{F}$), and ⟨choose⟩ messages ($\mathcal{C}$).

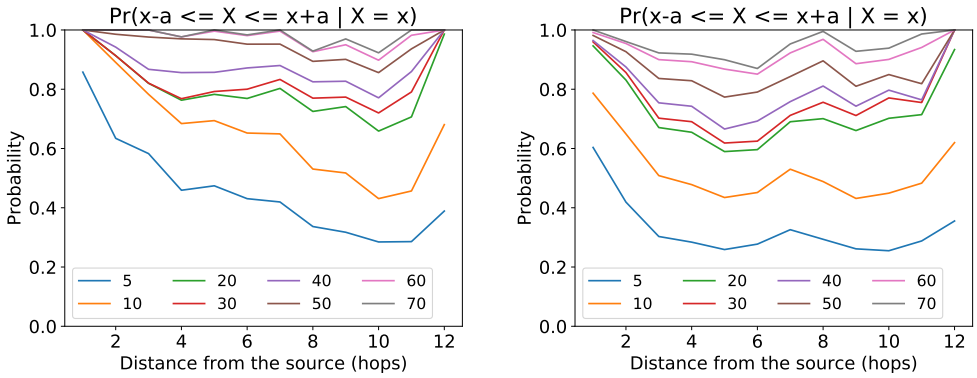
Figure 7.1 shows how long it takes a ⟨normal⟩ messages to travel to various nodes at specific hop distance from the source under *Protectionless* with no duty cycle. Each point represents the travel time of a single message. Note that at a given hop distance there is a arrival difference between 20 ms and 40 ms. This difference is difficult to precisely predict as the route the messages take is unspecified.

To gain a better understanding of what good early and late wakeup values will be for the *DynamicSPR* duty cycle, simulations were run using large values for the early and late wakeup times. A sample of these results is shown in Figure 7.6, with a normal distribution of mean and variance calculated from the data also shown. Ideally the majority of messages should be received around a difference of 0 ms as that means the message was received in the middle of the early and late wakeup periods. While there is a large peak around this point, many messages are also received both early and late. This means there will need to be a trade-off between smaller early and late wakeups (increased power savings) and the number of messages received and forwarded. So a way of determining these values is required.

In Figure 7.7 the probability that another message will be received within some time bound of another message is shown for different time bounds. These results were calculated as shown in Equation 7.25 by taking the weighted average of the ratio of messages received at a given latency x multiplied by the ratio of messages that would be received a ms before or after x. Where, X is the set of integer latencies that messages were received after. $C : X \to \mathbb{N}$ maps a latency in milliseconds to the number of messages that were received with that latency. $|R|$ is the total number of messages that was received. For simplicity the early and late wakeups were set to be the same.

$$\Pr\left(x - a \leq X \leq x + a \mid X = x\right) = \sum_{x \in X} \left(\frac{C(x)}{|R|} \sum_{t=x-a}^{x+a} \frac{C(t)}{|R|} \right) \tag{7.25}$$

These graphs are intended to provide an approximate insight into how changing the wakeup intervals affect the receive probability. It was expected that the delivery probability within a certain time bound would decrease as the distance from the source increases. However, this graph shows that the receive probability actually increases. The reason for this is that fewer messages are received at those higher source distances, making this probability value less accurate. A limitation of this graph is that it shows the probability for nodes at different distances from the source, but does not take into account how a lower probability for nodes closer to the source would effect the receive probability at those further from the source. Nevertheless, it is still a useful tool at estimating what early and late wakeup intervals to use. One of the main points to take away is that smaller wakeup intervals produce lower receive probabilities for ⟨fake⟩ messages compared to ⟨normal⟩ messages, so ⟨fake⟩ messages will need larger wakeup intervals compared to ⟨normal⟩ message wakeup intervals. The other point is that there is diminishing returns in terms of increasing receive probability when increasing the wakeup intervals.



(a) ⟨Normal⟩ (b) ⟨Fake⟩

Figure 7.7: Examples of the probability of receiving a message within a specific time bound from the previously received message at different distances from the source. When early and late wakeups were set to large values.

Algorithm 3 Duty Cycling Control

▷ Always start the radio when requested to
1: **event** StartRadio() →
2: **signal** StartRadioHardware()

▷ Only stop the radio if all components are not using it
▷ The sink node never duty cycles
▷ The source node ignores turn off rules for ⟨normal⟩ messages
3: **event** StopRadio() →
4: **if** ¬IsSinkNode()∧(IsSourceNode()∨NormalCanTurnOff())∧FakeCanTurnOff()∧¬sending **then**
5: **signal** StopRadioHardware()

7.2.2 Implementation

The implementation for *DynamicSPR* with duty cycling has one minor change to the way the algorithm works. In *DynamicSPR*, when a ⟨fake⟩ message is sent, the protocol will retry to send it until it is successful. This is not performed when duty cycling is enabled, as it causes ⟨fake⟩ messages to go out of sync with their expected time. If a node was to receive one of these delayed ⟨fake⟩ messages, it would expect to receive future ⟨fake⟩ messages at later times than they would actually be delivered.

The way the radio hardware is controlled is shown in Algorithm 3. Rather than maintain a large state machine that tracks all components and whether any is using the radio or not, when one component is finished using the radio it signals for the radio to be switched off. Only if no components are then using the radio is it actually switched off. When any component asks for the radio to be switched on it is always switched on. When a node is sending a message it requests the radio to turn on and leaves it on until sending is complete as shown in Algorithm 4.

Algorithm 4 Send Duty Cycling

```
    sending ← 0
1:  event SEND(msg) →                    5:  event SENDDONE() →
2:      signal STARTRADIO()              6:      sending ← 0
3:      SENDONRADIO(msg)                 7:      signal STOPRADIO()
4:      sending ← 1
```

Duty cycling for ⟨normal⟩ messages in Algorithm 5 follows the Equations 7.3, 7.4, and 7.5. The usage of Equation 7.3 can be seen in STARTOFFTIMERFROMMESSAGE(), Equation 7.4 can be seen in STARTOFFTIMER(), and Equation 7.5 in STARTONTIMER(). The duty cycling starts when a ⟨normal⟩ message is received. The time at which the *radio* received the message is recorded. It is important to use this time because the WSN needs the duty cycle to be anchored to when messages are actually received instead of when they are processed. If the current node time was used then the duty cycling may go out-of-sync with respect to the source node sending messages.

When a ⟨normal⟩ message is received the OffTimer is started which times until the node should turn the radio off. When the OffTimer fires, the OnTimer is started which will turn the radio back on when it fires. As one ⟨normal⟩ message is expected to be received each time the radio is on, once that message has been received the radio can be turned off. When the radio is turned off in OffTimer the offEarly flag is reset. The radio can only turn off when OnTimer is running and OffTimer is not running because the node is waiting to turn on and not waiting to turn off, or when the radio can be turned off early.

The duty cycling for ⟨fake⟩ and ⟨choose⟩ messages is shown in Algorithm 6. The algorithm is event triggered when the first new ⟨fake⟩ message is received. If the ⟨fake⟩ message arrives from a TFS or TailFS then the TempOff timer is started to turn the radio off, the DurationOn timer is started (if not already running) which turns the radio on to receive a ⟨fake⟩ message from the next fake node, and the ChooseOn timer is started (if the node is adjacent to the fake source) to wake the node up to receive the ⟨choose⟩ message from the fake source. If the ⟨fake⟩ message arrives from a PFS then the PermOff timer is started. The offEarly flag is also set to allow the radio to turn off immediately and the radio is signalled to stop. This can be done as no other ⟨fake⟩ message from a PFS is expected. Starting the PermOff timer will lead to the PermOn timer being fired after the PFS period.

When receiving a ⟨fake⟩ message if the message has not come from a TFS or TailFS then the counter TempNoReceive is incremented, if the message has come from a TFS or TailFS then it is reset. If three rounds of ⟨fake⟩ messages are missed from TFSs or TailFSs, then both the TempOff and TempOn timers are stopped. The

Algorithm 5 ⟨Normal⟩ Duty Cycling

```
    offEarly ← 0
    OnTimer, OffTimer ← ⊥, ⊥

 1: receive Normal⟨...⟩ →
 2:     now ← MessageReceiveTime()
 3:     if isNew() then
 4:         offEarly ← 1                          ▷ Turn off radio after receiving message
 5:         signal StopRadio()
 6:         StartOffTimerFromMessage(now)

 7: timeout (OnTimer) at now →
 8:     signal StartRadio()
 9:     StartOffTimer(now)

10: timeout (OffTimer) at now →
11:     offEarly ← 0
12:     signal StopRadio()
13:     StartOnTimer(now)

14: function StartOnTimer(now)
15:     if ¬IsRunning(OnTimer) then
16:         StartAt(OnTimer, now, P_src − W_e(𝒩) − W_l(𝒩))

17: function StartOffTimer(now)
18:     if ¬IsRunning(OffTimer) then
19:         StartAt(OffTimer, now, W_e(𝒩) + W_l(𝒩))

20: function StartOffTimerFromMessage(now)
21:     if ¬IsRunning(OffTimer) then
22:         StartAt(OffTimer, now, W_l(𝒩))

23: function NormalCanTurnOff
24:     return (IsRunning(OnTimer) ∧ ¬IsRunning(OffTimer)) ∨ offEarly
```

reason for this is that once a PFS has been allocated, TFSs and TailFSs should revert to being normal nodes. To save energy the awake periods reserved for ⟨fake⟩ messages from TFSs and TailFSs are no longer needed, so these timers (which perform the wakeups) can be stopped.

The radio can be turned off for this component when all the following are true: (i) it is not the receive window for a ⟨fake⟩ message from a TFS or TailFS or receiving a ⟨fake⟩ message from these node types is disabled, (ii) it is not the receive window for a ⟨fake⟩ message from a PFS or the radio can turn off early as this ⟨fake⟩ has been received, and (iii) it is not the receive window for a ⟨choose⟩ message.

Algorithm 6 ⟨Fake⟩ and ⟨Choose⟩ Duty Cycling

```
    offEarly ← 0
    tempDisabled, tempNoReceive ← 0, 0
    ▷ Timers are initialised to not fire at any time
    ChooseOn, ChooseOff ← ⊥, ⊥
    TempOn, TempOff ← ⊥, ⊥
    PermOn, PermOff ← ⊥, ⊥
    DurationOn, DurationOff ← ⊥, ⊥

 1: function ReceiveFromTempOrTail(now, ult_fake_count)
 2:     if isNew() then
 3:         StartTempOffTimerFromMessage(now)
 4:         nth_msg_delay ← P_TFS × ult_fake_count ▷ ult_fake_count is 0 for the first ⟨fake⟩ message
 5:         if ¬IsRunning(DurationOn) then
 6:             StartAt(DurationOn, now, D_TFS − nth_msg_delay − W_e(F))
 7:         if isAdjacent() ∧ ¬IsRunning(ChooseOn) then
 8:             StartAt(ChooseOn, now, D_TFS − I_TFS − nth_msg_delay − W_e(C))

 9: function ReceiveFromPerm(now)
10:     if isNew() then
11:         StartPermOffTimerFromMessage(now)
12:         offEarly ← 1                          ▷ Turn off radio after receiving message
13:         signal StopRadio()

14: receive Fake⟨src_type, ult_fake_count, ...⟩ →
15:     now ← MessageReceiveTime()
16:     if src_type ∈ { TempFake, TailFake } then
17:         ReceiveFromTempOrTail(now, ult_fake_count)
18:     else if src_type ∈ { PermFake } then
19:         ReceiveFromPerm(now)
20:     if ¬tempDisabled then
21:         if IsRunning(TempOff) ∧ ¬IsRunning(TempOn) then
22:             tempNoReceive ← 0
23:         else
24:             tempNoReceive ← tempNoReceive + 1
25:             if tempNoReceive ≥ 3 then
26:                 tempDisabled ← 1
27:                 Stop(TempOff)
28:                 Stop(TempOn)
```

7.3 Results

This section will detail the results for the *DynamicSPR* duty cycle algorithm. The COOJA simulator was used for all simulated experiments as TOSSIM cannot simulate the duty cycle or MAC layers. Only the `Fixed1` and `Fixed2` approaches were simulated and not the `Rnd` approach. This is because the *DynamicSPR* duty cycle algorithm only supports deterministic approaches.

Results are shown for different wakeup intervals in the form of this 6-tuple: $(W_e(\mathcal{N}), W_l(\mathcal{N}), W_e(\mathcal{F}), W_l(\mathcal{F}), W_e(\mathcal{C}), W_l(\mathcal{C}))$, which includes the early and late wakeup intervals for ⟨normal⟩ messages ($\mathcal{N}$), ⟨fake⟩ messages ($\mathcal{F}$), and ⟨choose⟩

Algorithm 6 ⟨Fake⟩ and ⟨Choose⟩ Duty Cycling: On and Off Timers

```
29: timeout (DurationOn) at now →
30:     signal StartRadio()
31:     StartAt(DurationOff, now, W_e(F) + W_l(F))

32: timeout (TempOn) at now →
33:     signal StartRadio()
34:     StartTempOffTimer(now)

35: timeout (PermOn) at now →
36:     signal StartRadio()
37:     StartPermOffTimer(now)

38: timeout (ChooseOn) at now →
39:     signal StartRadio()
40:     StartAt(ChooseOn, now, W_e(C) + W_l(C))

41: timeout (DurationOff) at now →
42:     signal StopRadio()
43:     Stop(TempOn)
44:     StartTempOn(now)

45: timeout (TempOff) at now →
46:     signal StopRadio()
47:     StartTempOn(now)

48: timeout (PermOff) at now →
49:     offEarly ← 0
50:     signal StopRadio()
51:     StartPermOn(now)

52: timeout (ChooseOff) →
53:     signal StopRadio()
```

	$W_e(\mathcal{N})$	$W_l(\mathcal{N})$	$W_e(\mathcal{F})$	$W_l(\mathcal{F})$	$W_e(\mathcal{C})$	$W_l(\mathcal{C})$
1	200	200	250	250	75	75
2	80	80	120	130	5	50
3	40	40	120	130	5	50
4	35	35	100	100	5	50
5	35	35	60	60	5	50
6	200	200	120	130	75	75

Table 7.1: Wakeup intervals for *DynamicSPR* in milliseconds

messages ($\mathcal{C}$). Five different sets of wakeup intervals are simulated (1–5 shown in Table 7.1), including one with large wakeup intervals, to test the effects of different duty cycles on the selected metrics. The graphs also include a *baseline* result, which is *DynamicSPR* running on COOJA without duty cycling turned on. This is included to situate any performance differences that may occur due to including duty cycling. The graphs also show a comparison with using TinyOS LPL with the parameters shown in Table 7.2. This is to demonstrate the performance difference between the developed sleep schedule and TinyOS's existing sleep schedule.

From the simulations various metrics about the performance of *DynamicSPR* were collected. The following metrics will be analysed in this section:

1. **Broadcast and Attacker Behaviour** — This is the patten in which messages are sent and the path the attacker takes in response.

2. **Duty Cycle** — This is the average percentage of time that the radio was on.

3. **Received Ratio** — This is the percentage of messages that were sent by the source and received by the sink.

Algorithm 6 ⟨Fake⟩ and ⟨Choose⟩ Duty Cycling

```
54: function StartTempOn(now)
55:     if ¬IsRunning(TempOn) then
56:         StartAt(TempOn, now, P_TFS − W_e(F) − W_l(F))

57: function StartTempOffTimer(now)
58:     if ¬IsRunning(TempOff) then
59:         StartAt(TempOff, now, W_e(F) + W_l(F))

60: function StartTempOffTimerFromMessage(now)
61:     if ¬IsRunning(TempOff) then
62:         StartAt(TempOff, now, W_l(F))

63: function StartPermOffTimer(now)
64:     if ¬IsRunning(PermOff) then
65:         StartAt(PermOff, now, W_e(F) + W_l(F))

66: function StartPermOffTimerFromMessage(now)
67:     if ¬IsRunning(PermOff) then
68:         StartAt(PermOff, now, W_l(F))

69: function FakeCanTurnOff
70:     return ((IsRunning(TempOn) ∧ ¬IsRunning(TempOff)) ∨ tempDisabled) ∧ (offEarly ∨
    ¬IsRunning(PermOff)) ∧ ¬IsRunning(ChooseOff)
```

	t_s (ms)	t_{tx} (ms)	t_d (ms)	*cca*
1	50	50	10	1150
2	50	50	100	400
3	75	75	10	1150
4	75	75	10	2300
5	75	75	100	400

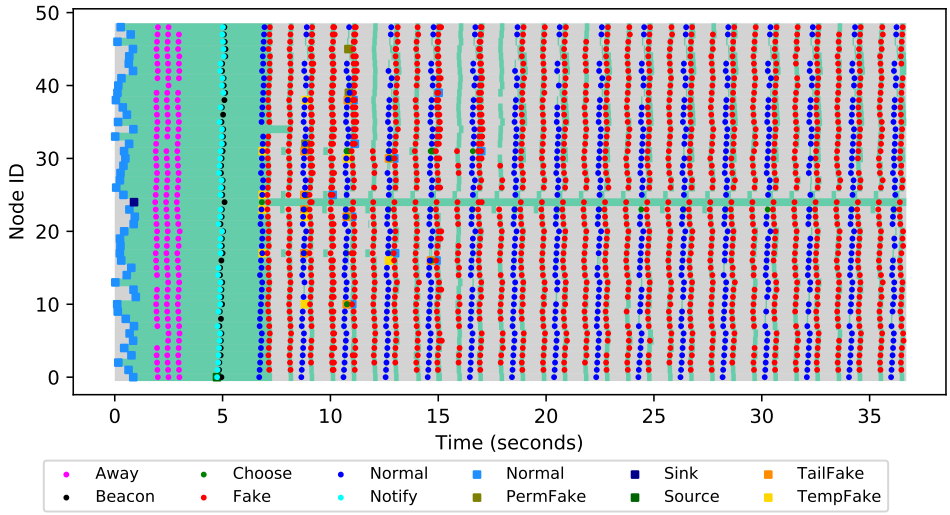
Table 7.2: Parameters for TinyOS LPL

4. **Average Number of Fake Messages Sent** — This is the average number of ⟨fake⟩ messages sent across all nodes.

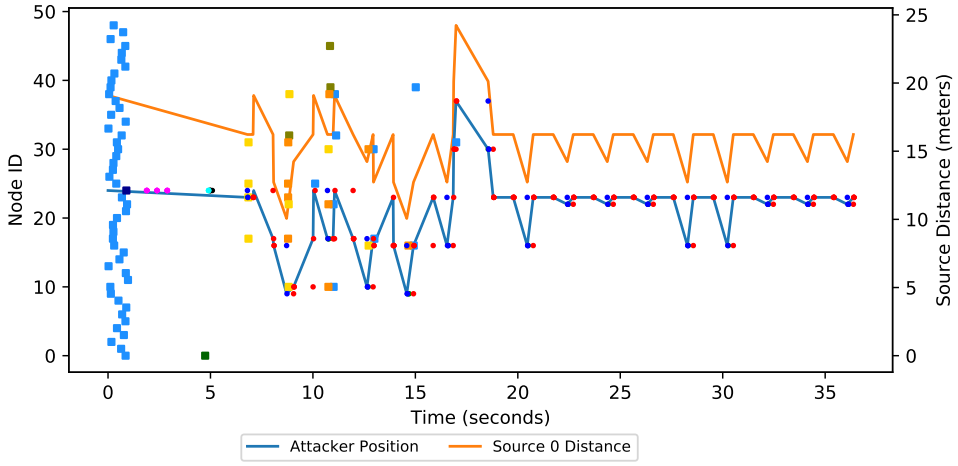
5. **Latency** — This is the average time it takes a message sent by the source to be received at the sink.

6. **Attacker Distance** — This is the average attacker distance from the source recorded at the end of a run.

7. **Capture Ratio** — This is the percentage of runs in which the attacker reaches the location of the source, i.e., captures the source.



(a) Messages Sent

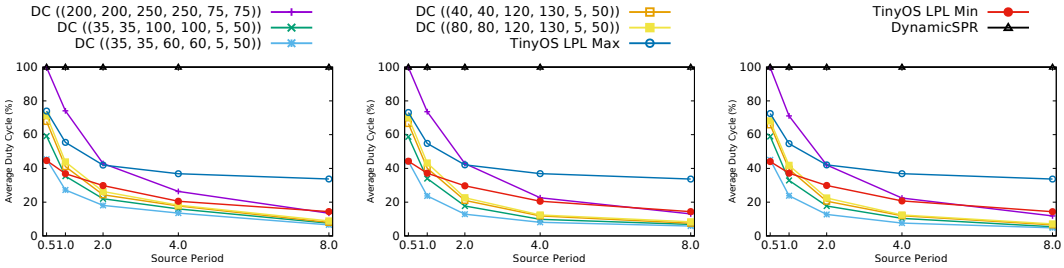


(b) Attacker messages received with attacker position and source distance

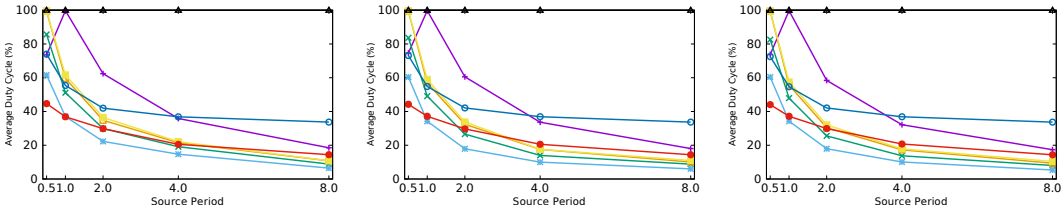
Figure 7.8: A demonstration of the behaviour of the *DynamicSPR* Duty Cycle algorithm and how the attacker moves in response to the messages sent.

7.3.1 Behaviour

The diagram shown in Figure 7.8 displays an example execution of *DynamicSPR* with duty cycling on a 7×7 network. ⟨Normal⟩ messages are sent with a P_{src} of 2.0 seconds and the `Fixed2` approach was used, so two ⟨fake⟩ messages are sent each P_{src}. This can be seen by the two columns of red ⟨fake⟩ messages between each column of blue ⟨normal⟩ messages. Some ⟨fake⟩ messages were delivered but not forwarded onwards early in the algorithm's execution, because the radio encountered an error. No retransmissions of these messages occurred to prevent ⟨fake⟩ messages



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.9: Results showing the Average Duty Cycle for different wakeup intervals.

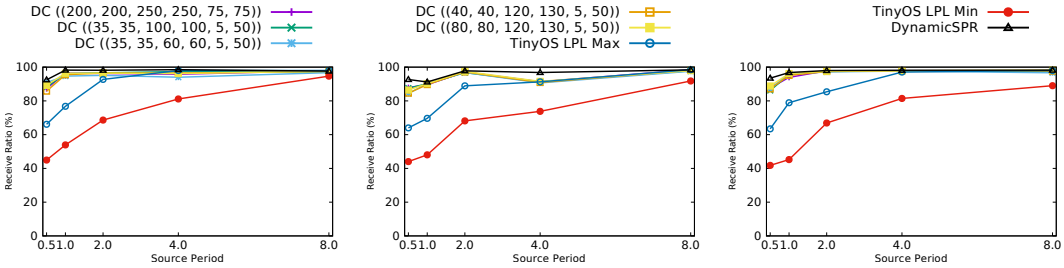
being sent at unexpected times.

When the background is green it means that the radio is on and when the background is grey the radio is off. The radios are all off until the nodes boot before the 1 second mark. Once booted the nodes will remain on until a ⟨normal⟩ and ⟨fake⟩ message have been received. The sink remains on for the entire simulation as it is assumed to be run at a base station connected to main power, so there is no need for duty cycling to save energy.

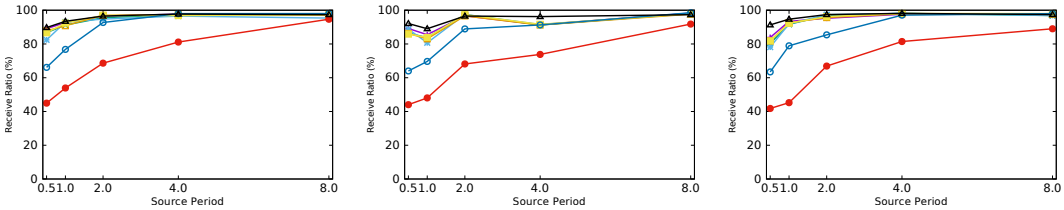
The attacker initially makes moves closer to the source before being pulled strongly away from it. Beyond 16 seconds only two fake nodes remain, the PFS at node 45 remains and a TailFS at node 23. Usually the TailFS would detect the PFS and become a normal node, however that has not happened in this case. This is caused by the PFS and TailFS using the same sequence numbers for their ⟨fake⟩ messages, which prevents the PFS's ⟨fake⟩ messages from reaching the TailFS and informing it to stop forwarding ⟨fake⟩ messages. This means that 2 messages continue to be sent from node 23 causing the attacker to cycle between moving closer to the source then back to node 23.

7.3.2 Duty Cycle

The graphs in Figure 7.9 show that the duty cycle technique for *DynamicSPR* does reduce the amount of time that the radio is left on for. Smaller wakeup intervals lead to a lower duty cycle. Decreasing the ⟨fake⟩ message wakeup intervals led to



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.10: Results showing the Received Ratio for different wakeup intervals.

a larger decrease as more ⟨fake⟩ messages are sent than ⟨normal⟩ messages. The baseline result shows a duty cycle of 100% because the radio is left on permanently as no duty cycling is performed.

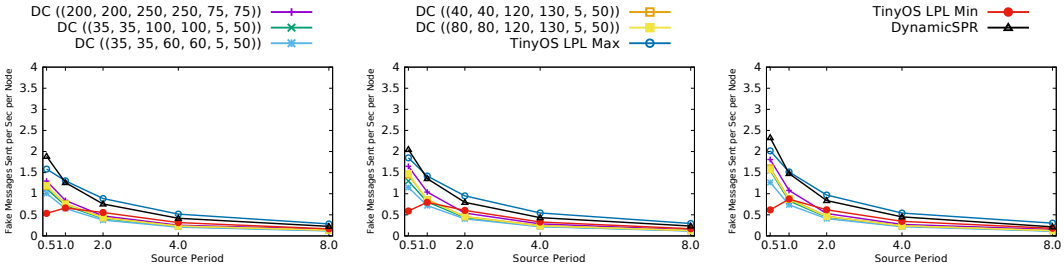
A larger source period led to smaller duty cycles because this duty cycle technique has a fixed wakeup for messages. When fewer messages are sent it means that there is a reduction in the number of radio wakeups that need to be performed. Because fewer absolute wakeups are performed over the same period of time, the ratio of awake time to asleep time decreases.

Using TinyOS LPL can produce a lower duty cycle than the one developed here under some parametrisations. When the source period is low (the message send rate is high) TinyOS LPL perform well. However, as the source period increases the *DynamicSPR* duty cycle tends to perform better than TinyOS LPL. This is because it takes advantage of knowledge of when to wake up, which TinyOS LPL does not.

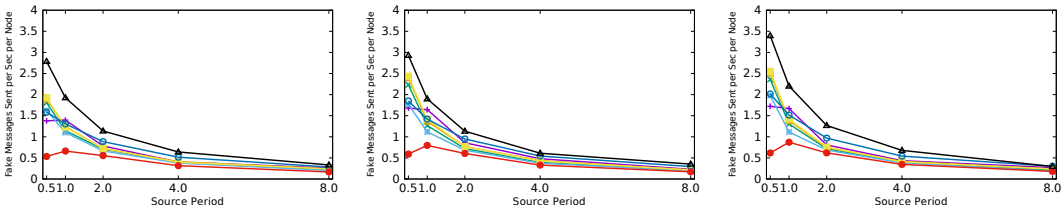
7.3.3 Received Ratio

The received ratio shown in Figure 7.10 is above 80%. The received ratio increases for slower source periods, as fewer messages are sent over the same period of time thus reducing the chance of message losses due to collisions. Larger network sizes also lead to a higher receive ratio. Because of the increase in the number of paths available for messages to travel, a collision has less effect in these larger networks.

The baseline results for *DynamicSPR* have a higher received ratio compared



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.11: Results showing the ⟨Fake⟩ messages sent per node per second for different wakeup intervals.

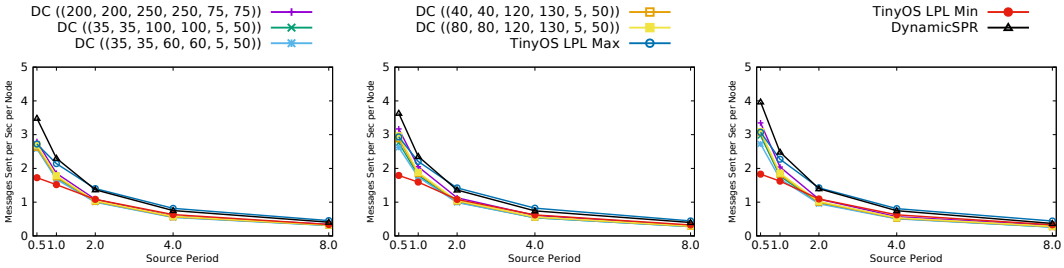
to when duty cycling is enabled. This is because there is no chance for a ⟨normal⟩ message to be lost because a node's radio is off. Smaller wakeup intervals have a lower received ratio for the opposite reason, as there is a greater chance for messages to be missed because the radio is off for longer.

The TinyOS LPL results show a lower receive ratio when the source period is low, but the receive ratio increases as the source period increases. This means that TinyOS LPL is really impractical for deploying with *DynamicSPR*, because when TinyOS LPL outperforms *DynamicSPR*'s duty cycle the receive ratio is low and when TinyOS LPL has a high receive ratio, *DynamicSPR*'s duty cycle outperforms it in terms of a lower duty cycle.

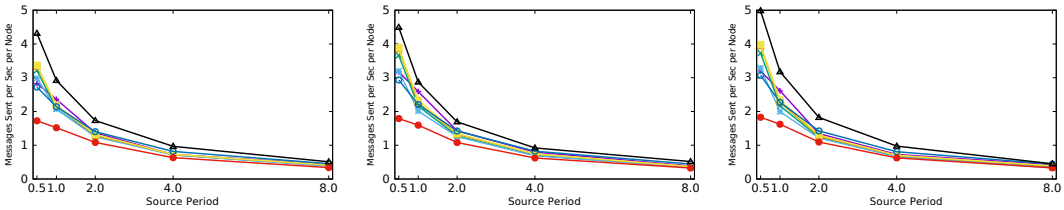
7.3.4 Messages Sent

In both Figure 7.11 and Figure 7.12 it can be seen that fewer messages are sent when duty cycling is enabled compared to when it is disabled. This is because when duty cycling is enabled there is the possibility that a message might be sent outside the of time the target node is awake. The reason that there appears to be a large difference between when duty cycling is enabled and when it is disabled, is because if a message is lost early in its path then it will not be received to be forwarded later.

Lower wakeup intervals lead to a lower number of messages sent. As there is less time when a node is awake, the probability of it receiving a message is lower. However, there only tends to be a small difference in the quantity of messages sent



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.12: Results showing the total messages sent per node per second for different wakeup intervals.

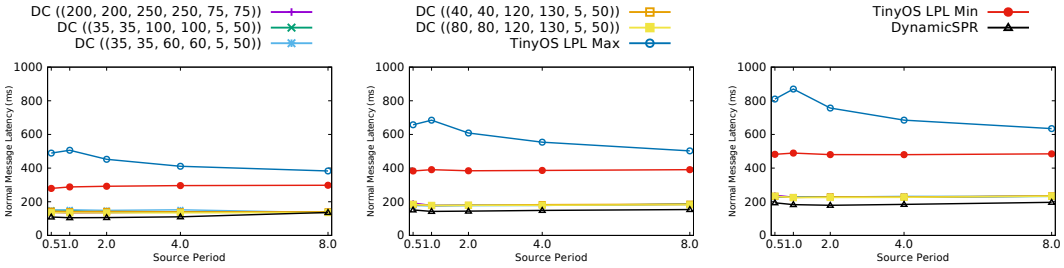
between the different wakeup interval parameters.

7.3.5 Normal Latency

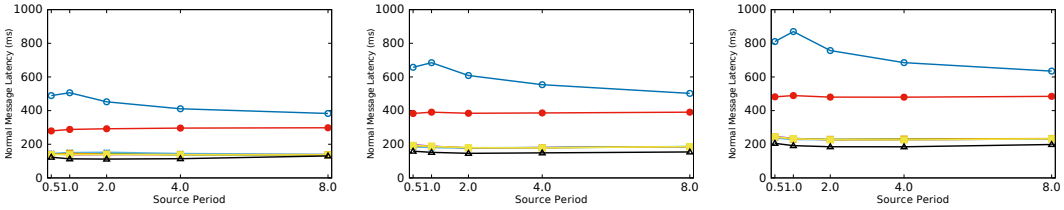
The latency of ⟨normal⟩ messages when duty cycling is enabled is increased compared to baseline *DynamicSPR* where duty cycling is not enabled, as shown in Figure 7.13. This occurs for all network sizes and source periods investigated. The increase in latency is typically about 40 ms to 50 ms.

An increase in this latency indicates that ⟨normal⟩ messages are not travelling along the shortest path from source to sink. It was expected that this latency would be similar to the case when no duty cycle technique is used, as the duty cycle only has nodes wakeup when a message should have arrived normally. So messages should have travelled along the shortest path. However, this result indicates that the wakeup intervals are causing ⟨normal⟩ messages to miss being forwarded to some nodes on the shortest path and are instead taking a slightly longer path from source to sink.

The latency is very similar for both `Fixed1` and `Fixed2`, meaning that the ⟨fake⟩ message rate does not have a large impact. This was expected as at maximum only one set of ⟨fake⟩ messages could be scheduled at a similar time to ⟨normal⟩ messages.



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.13: Results showing the Normal Latency for different wakeup intervals.

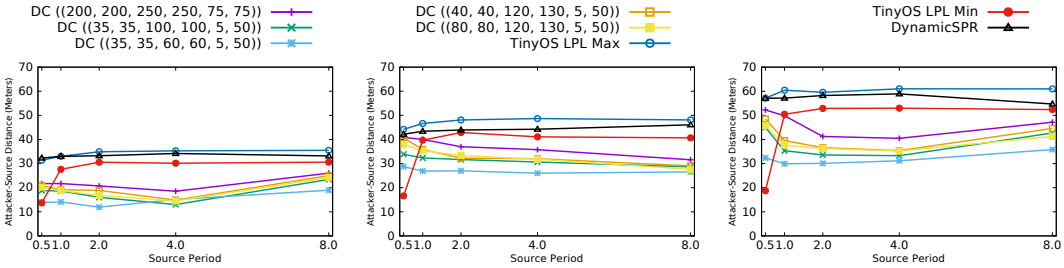
7.3.6 Attacker Distance

The `Fixed2` approach is better at pulling the attacker away compared to `Fixed1`. The issue with `Fixed1` is that it is only capable of keeping the attacker at the same distance it is from the source and does not send enough messages to pull the attacker further away. The `Fixed2` approach performs much better compared to `Fixed1` because it sends 2 messages each period, meaning it is much more tolerant to message losses.

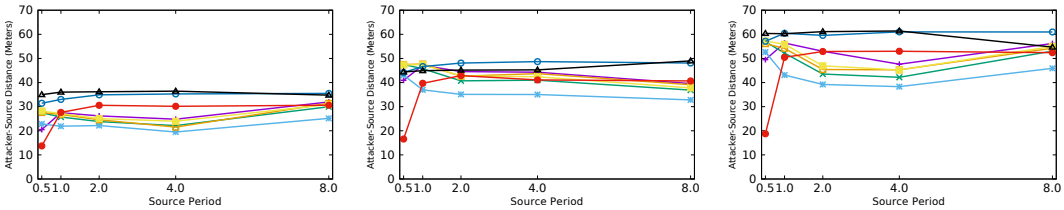
Enabling duty cycling leads to a reduction in the ability to pull an attacker further from the source, as shown in Figure 7.14. With fewer messages being sent and received it means that fewer ⟨fake⟩ messages are reaching the attacker to pull it further from the source.

7.3.7 Captured

The final metric that will be investigated is the capture ratio shown in Figure 7.15. When enabling duty cycling it will lead to a higher capture ratio. Different levels of increase are observed for different parametrisations. It was expected that smaller networks would experience a larger capture ratio compared to the larger networks, as this pattern was observed previously. However, the small size 7×7 network when using the `Fixed1` approach has a much higher capture ratio than expected. The problem is that under the `Fixed1` approach only 1 ⟨fake⟩ message is sent for each ⟨normal⟩ message. When there is the chance that some of these messages will not



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



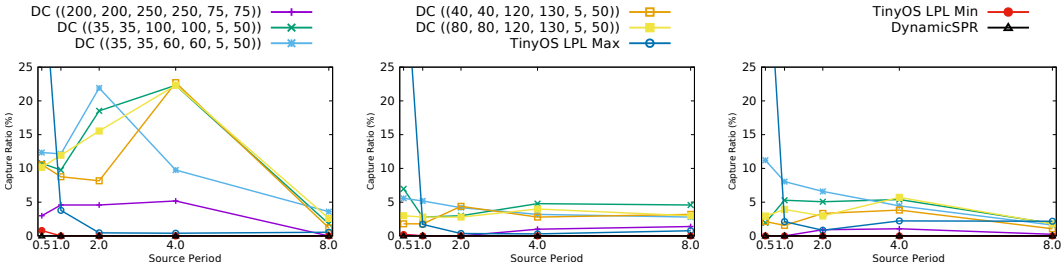
(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

Figure 7.14: Results showing the Attacker Distance for different wakeup intervals.

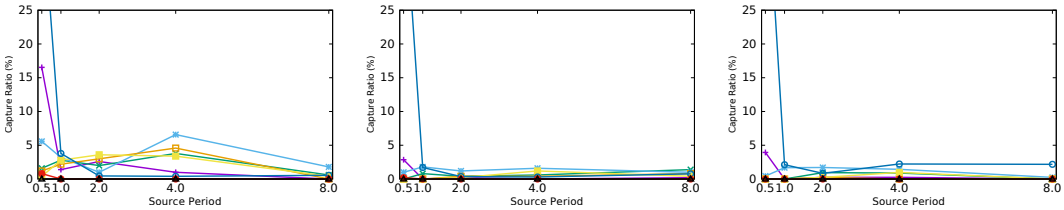
reach the attacker, `Fixed1` is no longer capable of keeping the attacker at a certain distance from the source. As has already been discussed, the smaller wakeup intervals lead to a lower chance of a message being forwarded. This can be seen in the graph for the 7×7 in Figure 7.15a, where smaller wakeup periods have larger than 5% capture ratio for most of the source periods. Larger network sizes perform better, because the attacker has more space to cover before it gets close to the source. This reduces the impact of lost ⟨fake⟩ messages, as `Fixed1` can hold the attacker at a certain distance that is closer to the source. When the distance between the sink and source is larger it provides extra space for *DynamicSPR* to recover and pull the attacker back if it makes moves towards the source.

The `Fixed2` approach performs better than `Fixed1` because rather than trying to hold the attacker at a certain distance, sending two messages each source period tried to pull the attacker to the fake source's location. By sending two messages in a period, *DynamicSPR* is better able to tolerate lost messages. `Fixed2` still has poor performance on 7×7 networks compared to the two larger networks for the same reason that `Fixed1` performed poorly on that network size.

In most cases, when the duty cycle had larger duty cycle intervals, the results show that the capture ratio is lower. This means that there is a trade-off to be made between energy cost and SLP. To provide better reliability when forwarding a message and thus a lower capture ratio, a node needs to wakeup for a longer period.



(a) Fixed1 Approach for 7×7, 9×9 and 11×11 sized networks



(b) Fixed2 Approach for 7×7, 9×9 and 11×11 sized networks

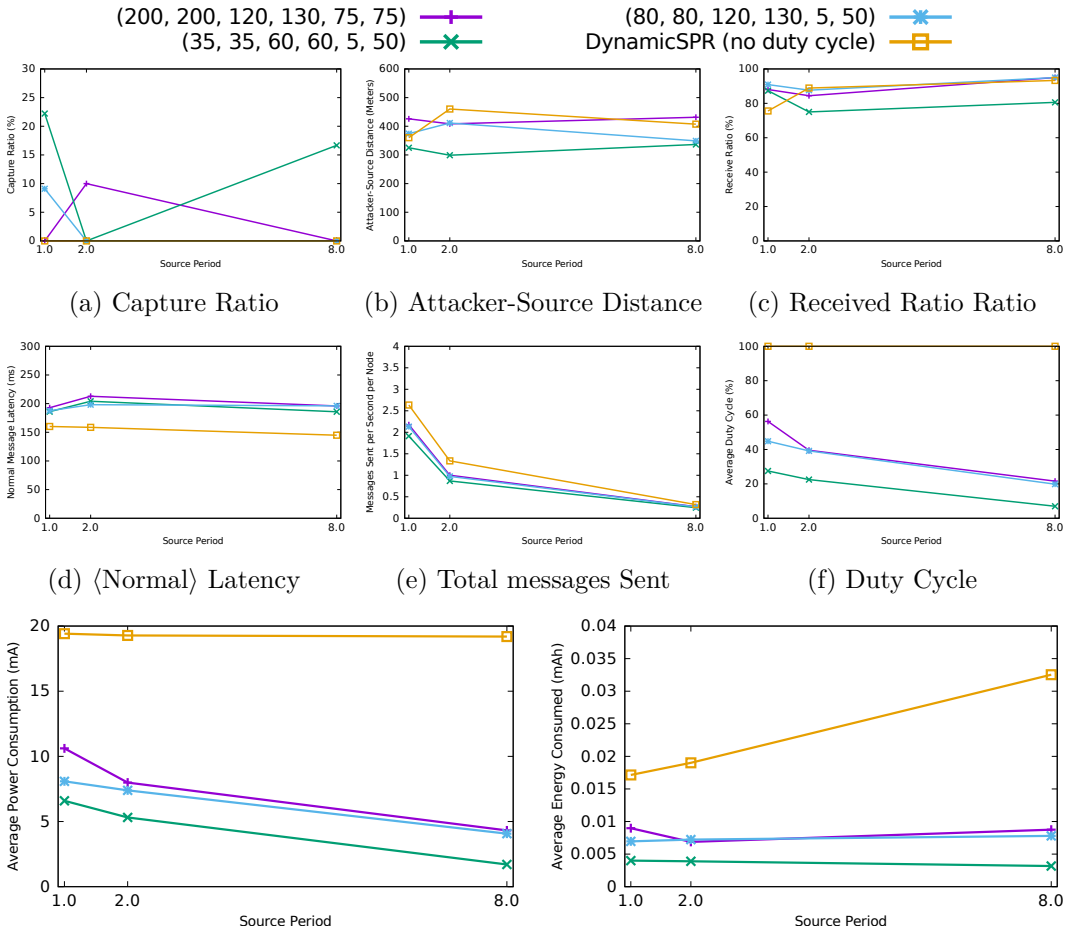
Figure 7.15: Results showing the Capture Ratio for different wakeup intervals.

7.4 Testbed Results

The results for *DynamicSPR* with duty cycling is shown in Figure 7.16 for the `Fixed1` approach and in Figure 7.17 for the `Fixed2` approach. The results for metrics previously investigated using the COOJA simulator show similar trends to the testbed results obtained. The key results introduced here are the actual energy and power measurements obtained from the testbed experiments. Figure 7.16g and Figure 7.17g show the average power consumption of the nodes in milliamps, and Figure 7.16h and Figure 7.17h shows the average energy consumed over the duration of testbed experiment, from the start to when either the source was captured or the safety period expired. These graphs show that using *DynamicSPR* with duty cycling can reduce the power consumption of the sensor nodes by more than 75% when a source period of 8 s is used. An interesting point is in Figure 7.17g where the average power consumption for a tuple of short wakeups is larger than results that use longer wakeups. The reason for this is that the duty cycle is actually higher (as shown in Figure 7.17f), which is caused by the algorithm lacking sufficient information to know when to sleep leading the radio to remain on.

7.5 Discussion

During the development of this duty cycle a major issue that was encountered was that under certain parametrisations the duty cycle went out of sync. This meant that



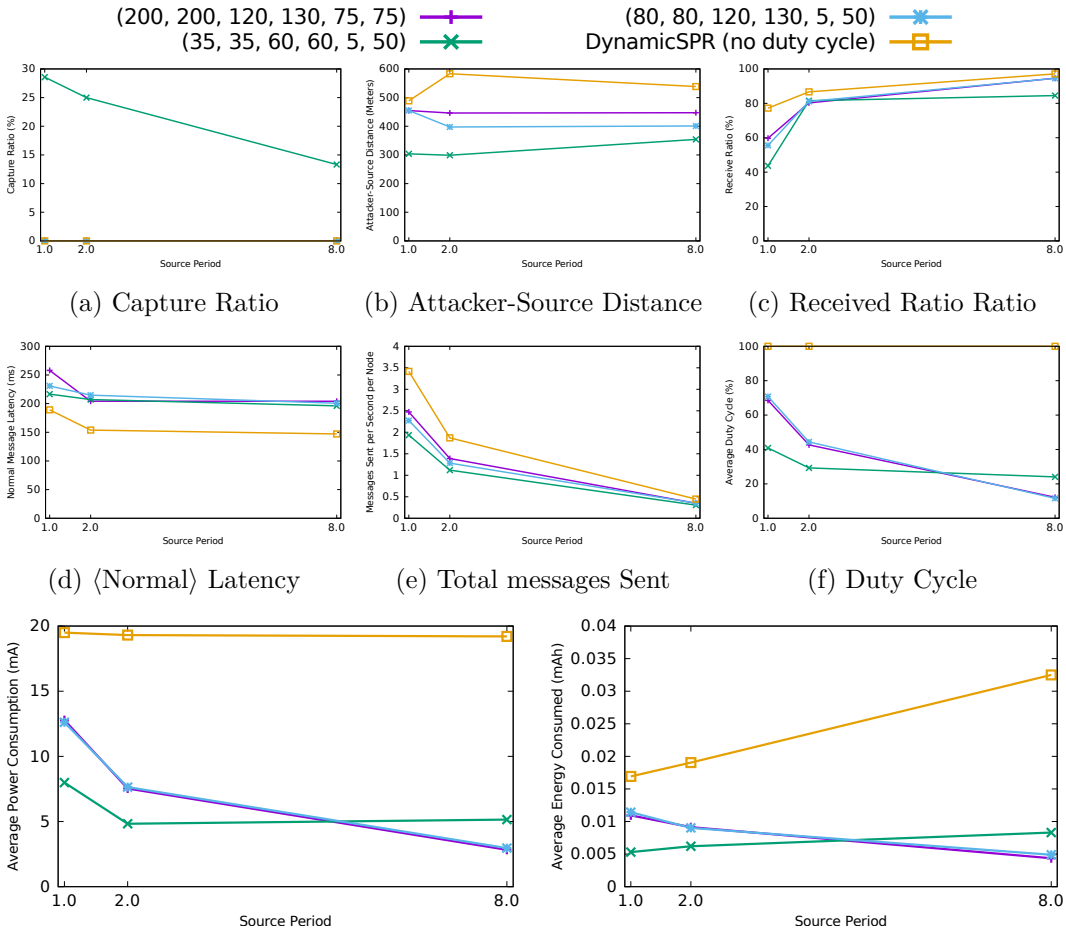
(a) Capture Ratio (b) Attacker-Source Distance (c) Received Ratio Ratio

(d) ⟨Normal⟩ Latency (e) Total messages Sent (f) Duty Cycle

(g) Average Power Consumption (h) Average Energy Consumed

Figure 7.16: FlockLab testbed results for *DynamicSPR* with duty cycling and approach `Fixed1`.

nodes were waking up when there were no messages to receive and sending messages when no nodes were awake to receive the messages. One of the main changes that was performed set a TailFS to use parameters from a TFS. In previous work [25] TailFSs used the parameters from a PFS because TailFSs have a potentially unbounded duration. However, this parametrisation did not work for this duty cycle because the PFS period changes depending on the percentage of ⟨fake⟩ messages received at the source. These TailFSs led the duty cycle to go out of sync. By changing TailFSs to use TFS parameters this issue was fixed. There are also a number of other issues that can lead the duty cycle to go out of sync which will now be described.



(a) Capture Ratio (b) Attacker-Source Distance (c) Received Ratio Ratio

(d) ⟨Normal⟩ Latency (e) Total messages Sent (f) Duty Cycle

(g) Average Power Consumption (h) Average Energy Consumed

Figure 7.17: FlockLab testbed results for *DynamicSPR* with duty cycling and approach `Fixed2`.

7.5.1 Handling Clock Drift

Over time a sensor node's clock will drift. This is due to the crystal oscillator (which is used to generate a series of clock pulses) not always oscillating at the desired frequency. Due to environmental effects the clock may run faster or slower. For a 32.768 kHz oscillator with a clock stability of ±20ppm, time could drift 52.7 ms a day. This drift is slow enough to not be an issue over a small number of messages as each node has a large wakeup period to allow it to handle differences in the time it takes a message to reach the node. However, over time this drift could cause issues to which there are two solutions. The first is to perform some form of clock synchronisation (such as FTSP [121]), however this would add an extra energy cost when running *DynamicSPR*.

An alternate approach would be to periodically reset the time from which

subsequent wakeups are computed relative to. In this implementation all wakeups are computed relative to the time the first ⟨normal⟩ and ⟨fake⟩ received. An alternative would be to periodically reset the wakeup calculation from a different message, for example, once every hour. Another way to mitigate the effect of clock drift might be to use an alternate way to calculate the time to sleep. For example, the early and late wakeups could be adjusted per message, with the aim to keep the next message's arrival time in the middle of the wakeup period. This would be able to adjust for latency changes as well as clock drift changes. The per message adjustment could be performed using an exponential weighted moving average of the latencies. The weights would need to be set to ensure that there are no wild swings in the early and late wakeups and that the majority of messages arrive at the end of the early wakeup.

7.5.2 Non-deterministic Parameters

The implementation of *DynamicSPR*'s duty cycle requires that parameters are deterministic. The reason is that a node cannot know when to wakeup if another node is determining when to send messages randomly. However, there are ways to support randomised parameters. One example, for the `Rnd` approach that sends 1 or 2 messages over the source period, is to have all nodes in the network wakeup for both cases. The cost here is the extra wakeups during which no messages will be received. This situation works because `Rnd` has a small state space, but would be unsuitable for randomly sending a much larger number of messages.

Randomising other parameters such as the initial start delay could be handled by increasing the early and late wakeup periods. These periods would need to be increased such that the range of values that could be used for the initial start delay are covered. Parameters such as the duration would need to remain fixed and deterministic. If the duration had a random range then the duty cycle would go out of sync after some time. This means that there is a trade-off between increasing randomness of events (which increases the attacker's uncertainty) and the time nodes are awake and listening for messages.

7.5.3 Other Issues

There are also a number of other issues that could lead this duty cycle to go out of sync. So in future work the duty cycle should be extended to detect and correct these kinds of issues. One way would be to increase the wakeup windows when messages are missed. Once a message is received the early and late wakeups could be reset to centre the expected arrival time around the arrival time of that message. This

duty cycle did not aim to be able to handle error conditions as the focus was on developing a working technique with minimum protocol overhead. But these kinds of issues will need to be handled for the duty cycle to be fully usable in practice.

7.5.4 When to use TinyOS LPL?

One of the assumptions made in the development of this duty cycle is that data will be periodically set at a predetermined rate. Using that knowledge and the understanding of how the fake source allocation occurs, a low duty cycle with good performance in other metrics can be achieved using the synchronous duty cycle presented in this chapter. However, if the transmissions do not occur periodically then this style of duty cycling would not work. Instead, an asynchronous [30] duty cycle such as TinyOS Low Power Listening which wakes the radio up periodically would be required. As has been shown with the results presented in this chapter, TinyOS's LPL does not perform well for low source periods as the delivery ratio is low. Alternate asynchronous duty cycles should be investigated to try to obtain better performance with *DynamicSPR*.

7.6 Conclusion

In this chapter the timing analysis of *DynamicSPR* has been extended to support calculating the times that nodes should wakeup to process a message. Different early and late wakeup intervals have been investigated for the three main messages involved with SLP provision: ⟨normal⟩, ⟨fake⟩, and ⟨choose⟩. Longer intervals were used for ⟨fake⟩ messages as there is greater uncertainty regarding when a node should expect to receive and forward these messages onwards. It is important to ensure high reliability of ⟨fake⟩ message delivery as they are crucial for SLP provision.

Capture ratios are up to 7 percentage points higher in the worst case when using a duty cycle with a source period of 1 second per message. A longer source period leads to a lower duty cycle, as the wakeup intervals are fixed periods of time, and a longer source period means fewer wakeups. With a source period of 8 seconds per message, a duty cycle of 10% is possible. This helps to offset the cost of broadcasting ⟨fake⟩ messages by allowing the radio and CPU to sleep for up to 90% of the time.

Chapter 8

Discussion

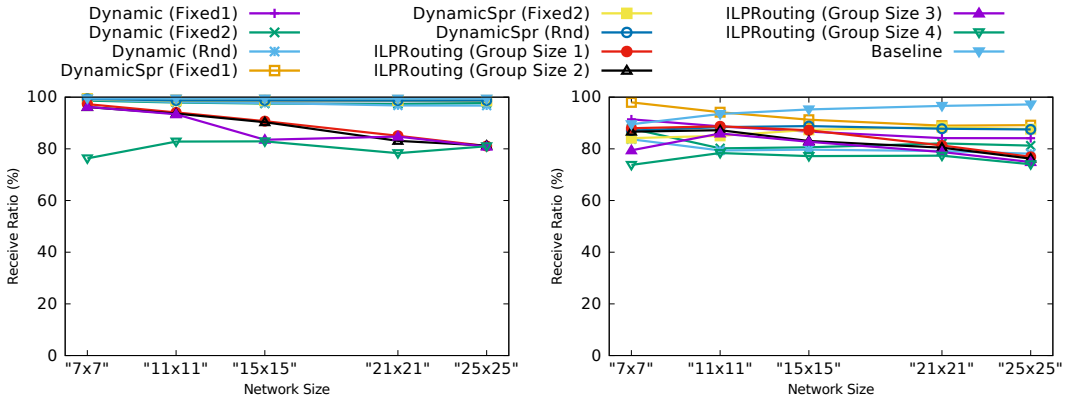
Throughout the work that has been performed a number of common patterns have been observed, even across very different techniques. These commonalities convey certain practices, issues, and ideas that can be applied to SLP-aware routing as a whole. This chapter will investigate these common patterns and explain their wider importance. Also in this chapter, a comparison between the three SLP-aware routing protocols *Dynamic*, *DynamicSPR*, and *ILPRouting* will be performed. This comparison is important because the different types of SLP-aware routing make different trade-offs, so it is important to situate the performance differences.

8.1 A Comparison of Techniques

In this thesis three different techniques have been presented, two using fake sources (*Dynamic* and *DynamicSPR*) and another that groups and delays messages (*ILPRouting*). These techniques perform very differently and will perform better in different scenarios. This section will compare the three techniques across the metrics previously analysed. Only results for the two source periods 2.0 seconds and 0.25 seconds will be presented for brevity.

Other techniques such as Phantom Routing [81, 192] were implemented and simulated, however, they performed very poorly. The reason behind this is that the TOSSIM simulator simulates noise and the possibility of links becoming unidirectional. However, the Phantom Routing techniques do not specify any retransmission strategy to handle this, causing all messages to fail to reach the sink. They are not included in this comparison because a 0% capture ratio was achieved by providing a near 0% delivery ratio.

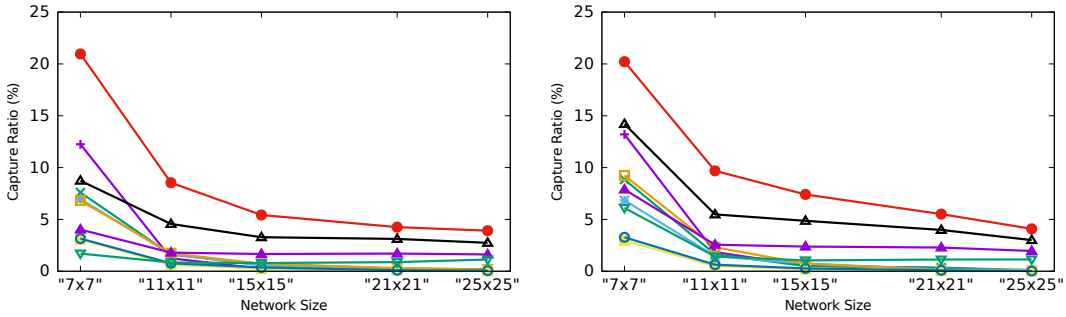
In terms of the receive ratio, *Dynamic* and *DynamicSPR* tend to outperform *ILPRouting*. This was unexpected as *Dynamic* and *DynamicSPR* rely on the multiple paths from flooding to provide reliability and *ILPRouting* provides reliability via



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure 8.1: Results showing the percentage of ⟨normal⟩ messages received at sink.



(a) Source Period 2.0 seconds

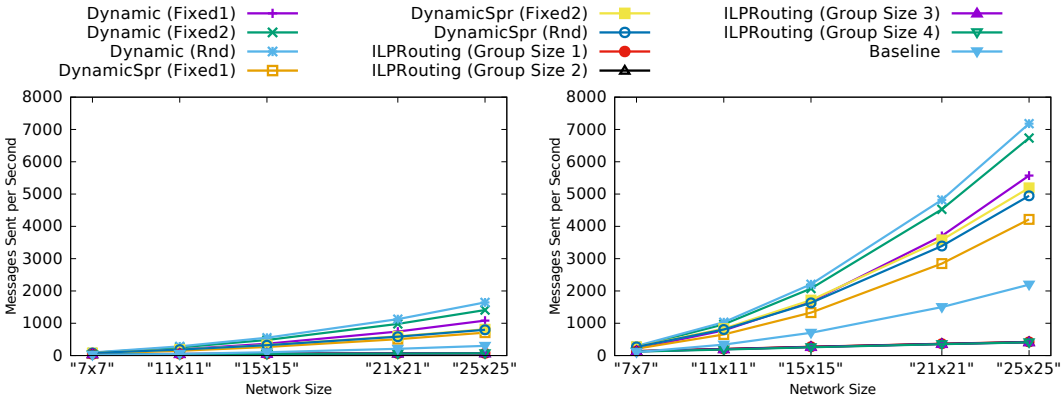
(b) Source Period 0.25 seconds

Figure 8.2: Results showing the capture ratio.

retransmissions. As was explained in Chapter 5 *ILPRouting* has a lower receive ratio due to some delayed messages not reaching the sink before the safety period expires and thus being counted as lost. So overall, the two protocols are expected to have a similar message delivery reliability.

The results for the capture ratio show that depending on the parametrisation of the technique *ILPRouting*, *Dynamic*, and *DynamicSPR* are all capable of achieving near optimal levels of SLP. However, achieving these levels of SLP require different trade-offs. For *Dynamic* and *DynamicSPR* this trade-off is in terms of the number of messages sent and received. This cost is caused by using flooding for both sending ⟨normal⟩ messages as well as sending the ⟨fake⟩ messages that provide SLP. While this cost can be offset by using the duty cycle schedule to ensure that the radio is turned off when not needed, there will still be a high cost in terms of energy usage to send and receive these messages.

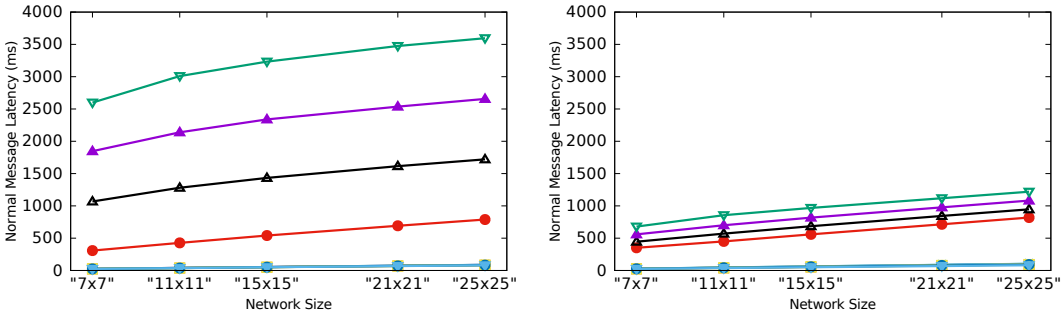
On the other hand, *ILPRouting* sends a small number of messages. The amount sent is lower than *Protectionless* flooding as single path routing is used,



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure 8.3: Results showing the average number of messages sent per second.



(a) Source Period 2.0 seconds

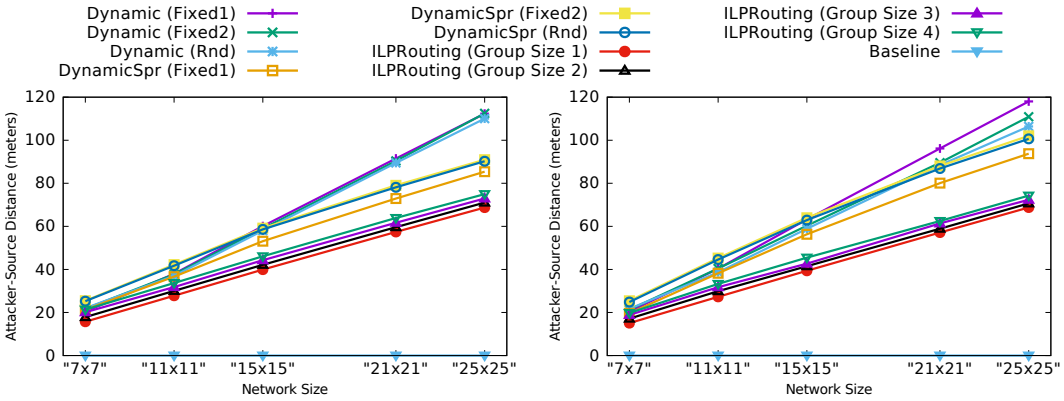
(b) Source Period 0.25 seconds

Figure 8.4: Results showing the time it took a normal message to reach the sink.

meaning that not all nodes in the network are involved with sending and receiving messages. But there is still a trade-off, as a large average latency is required to provide these levels of SLP. *Dynamic* and *DynamicSPR* have a very small latency as ⟨normal⟩ messages travel along the shortest path.

Finally, in terms of ability to pull the attacker further from the source *Dynamic* performs the best. This comes at a high cost, as it also sends the largest number of messages. *DynamicSPR* provides the next best ability to pull the attacker further from the source and finally *ILPRouting* performs the worst out of the SLP routing protocols. This result is interesting because the objective function of the ILP model that led to *ILPRouting* was to maximise the attacker's distance from the source. This means that the *ILPRouting* algorithm is incapable of achieving the model's aims, but manages to instead optimise the number of messages sent instead. In Section 9.2 ways to alter the model will be discussed that may allow it to output a different result that indicates an alternate way to provide SLP.

In summary, these results show that *Dynamic*, *DynamicSPR*, and *ILPRouting*



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure 8.5: Results showing the attacker's distance from the source.

are all useful in providing different objectives, but in doing so require different costs. *Dynamic* is the best a pulling an attacker to a far away location, but the cost to do is a large number of messages sent. *DynamicSPR* reduces the energy cost of *Dynamic*, but isn't as capable of pulling the attacker as far away. Finally, *ILPRouting* sends the lowest number of messages, but its trade-offs are a high latency and worse ability to pull an attacker away from the source.

This means that these techniques have certain properties that may make them unsuitable for certain applications. For example, in terms of animal monitoring a latency of a few seconds will not impact the ability to track the animals. In fact the minimum interval of location reporting seen from real deployments is every 15 minutes. But as the cost of the network is high, it is important to maximise its lifetime to ensure good value for money. This means that *ILPRouting* would be a suitable technique for this scenario. On the other hand, under a battlefield scenario the cost of hardware and lifetime (in terms of months) may be an aspect that deployers are willing to trade for more confusion regarding the location of assets. However, a latency of a few seconds would be impractical as real-time communication with minimum delay is required. This means that *Dynamic* or *DynamicSPR* are techniques more suited to these conditions. Selecting the technique to use for different scenarios is still an open problem. However, there is existing work that uses decision theory to select the dominating SLP routing protocol according to metrics (gathered from simulation) important for a specific scenario. These techniques have been applied to the *DynamicSPR* and *ILPRouting* algorithms presented in this thesis in [61].

8.2 Impact of Network Configuration

When analysing the results for the three algorithms presented in this work, one result that stood out was that larger network sizes provided a better level of source location privacy (as shown in Figure 8.2). The reason behind this is that the larger networks investigated had a larger distance between the sink and the source, which provides additional time for the SLP routing protocols to recover from poor situations. For example, if the attacker has made moves towards the sink due to collisions or noise that prevented ⟨fake⟩ messages from reaching it, then *Dynamic* or *DynamicSPR* has more opportunities to pull the attacker back to where it should be if the attacker has a greater distance to travel. An important consideration to take away from this is that when designing a network to monitor assets it is better to use many devices with a very short wireless range than it is to use few devices with a very large wireless range. This increases the number of hops that the attacker will need to travel, and decreases the range in which it will receive information. On one hand, this is beneficial to the network deployer as the energy cost of transmitting with a lower power output is less than a high power output, meaning the network's lifetime will be longer. The downside is that there is a higher monetary cost in purchasing the hardware and higher administration cost in maintaining it.

Another important aspect is the rate at which messages are sent from the source. This is a factor that was previously identified to lead to better performance when the rate is low (the source period between messages is high) [179], and has also been confirmed by the results presented in this work. The reason for this is that fewer collisions occur between messages, allowing SLP-aware routing to perform better. A less busy network will provide a greater level of SLP, but the frequency of updates from the network will be lower. Therefore, network maintainers will need to consider the frequency of reports and other network traffic, because there is a trade-off between the frequency of event reporting and the SLP provided.

So while the algorithms presented have their own trade-offs (between messages sent for *Dynamic* and *DynamicSPR*, and latency for *ILPRouting*) there is also a trade-off in the way in which the WSN is set up and used. More nodes with a shorter range will perform better than fewer nodes with a longer range, and less network traffic will allow higher SLP provision than high network traffic.

8.3 Impact of Bursty Network Traffic

The techniques developed in this thesis are developed around the idea that an application will need to periodically report information about an asset to some observers. In Section 8.1 it was stated that the techniques developed are useful in

different scenarios, either when the network is delay tolerance (*ILPRouting* should be used) or when the network is delay intolerant (in which *Dynamic* and *DynamicSPR* should be used). If the assumption of periodic messages generation is relaxed and alternate message generation patterns occur (such as bursty traffic) then changes would need to be made in some instances. *ILPRouting* would not need changes due to it routing messages from the source. But *Dynamic* and *DynamicSPR* would need to have the way ⟨fake⟩ messages are generated as ⟨normal⟩ messages travel along the shortest path. One possibility is to obtain the probability distribution which represents the interval between bursts of messages and use this to generate ⟨fake⟩ message bursts at intervals produced by the distribution. It is likely that parts of the timing analysis derivation would need to be altered. The duty cycle for *DynamicSPR* would need to be reconsidered due to the probabilistic generation of messages and the uncertain interval between messages.

8.4 Categorisation of Techniques

In [74] it was proposed that SLP techniques could be decomposed into three components: (i) decoy selection, (ii) control message routing, and (ii) decoy message routing. Each of these components either had a temporal or spatial aspect. The *Dynamic* algorithm was used as an example of spatial selection as the fake sources attempt to lure the attacker to a different location in the network. *DynamicSPR* remains a spatial approach as it uses the same fundamental approach as *Dynamic* does, but uses an alternate fake source allocation approach.

Two examples were given for a temporal approach: Phantom Routing and a tree-based technique. These two were categorised as temporal because they provided SLP by taking different routes, meaning an attacker failed to receive that message. In this sense the attacker was delayed along its route to the source. The *ILPRouting* protocol takes a similar approach to these techniques, as it delays messages so they are grouped together. In doing so the attacker can make fewer good moves towards the source over some period of time. However, a better categorisation of these techniques is that they are a hybrid of spacial and temporal, since they involve both delay and spatially separate message paths to delay the attacker.

What this means for *Dynamic*, *DynamicSPR* and *ILPRouting* is that enough redundancy needs to be provided either in terms of space or time. Spatial redundancy can simply be in the form of over-provisioning a WSN by deploying a larger than needed network. Temporal redundancy will be the network maintainer's ability to accept delay in message reception. These two forms of redundancy indicate what sorts of applications these protocols may be better suited to. For example,

animal monitoring will be tolerant of high latencies, so *ILPRouting* would be suitable. However, a WSN deployment in a battlefield scenario (where the valuable assets are military personnel) would not be able to tolerate a high latency, so *Dynamic* or *DynamicSPR* would be the preferred protocol.

The lack of spatial redundancy was observed when attempting to run experiments for *ILPRouting* on the FlockLab testbed. The topology of the testbed meant that there was insufficient space around the sink for messages to be routed. Due to the lack of space *ILPRouting* performed poorly, as one of its assumptions is that there is sufficient spatial redundancy for messages to be routed around the sink.

8.5 Alternative Base Routing Protocols

For *Dynamic*, *DynamicSPR* and *ILPRouting*, flooding was used to obtain the safety periods that were used to calculate the capture ratio. In *Dynamic* and *DynamicSPR*, flooding was also used as the base routing protocol for ⟨normal⟩ messages. The reason behind this is that flooding is a very poor routing protocol for applications that require SLP, as it leaks information about the source node no matter the location of an attacker. This is why it was important to develop SLP routing protocols against it, because it leaks maximal information.

One of the major issues with flooding is that it sends many messages which leads to large energy consumption. Another issue is that it can have poor delivery ratios due to the lack of retransmissions. Other protocols such as the Collection Tree Protocol (CTP) [57] or Ad-hoc On Demand Distance Vector Routing (AODV) [144] can provide much better performance in terms of messages sent and delivery ratio. They are also worse for the attacker as less information is leaked to it due to the fewer messages sent over a smaller area of the network.

For *Dynamic* and *DynamicSPR*, applications may wish to use CTP or AODV as the base routing protocol. However, as the real source no longer sends messages via flooding then neither should the fake sources. The fake sources would need to send ⟨fake⟩ messages along single path routes akin to CTP or AODV. These fake routes would need to be along a path that competes with the real path. It is likely that multiple junction points along the real path are necessary to ensure an attacker is led off the real path. This sort of technique would be similar to some tree-based SLP techniques, such as [115]. By making this change the *Dynamic* and *DynamicSPR* would be able to become more efficient in terms of the number of messages sent. This is not applicable to *ILPRouting* as the routing protocol has already been replaced.

8.6 Optimising for Different Metrics

In this thesis the focus has been on creating near optimal routing protocols to provide SLP. To obtain near optimality in the capture ratio, a reduction in performance needed to be traded off in other metrics. For example, *Dynamic* and *DynamicSPR* have a high energy usage due to the number of messages sent, and *ILPRouting* has a high delivery latency due to the way delay is used to group messages. All three techniques provided the ability to parametrise them, which controlled the amount of trade-off between capture ratio and the specific metrics. *Dynamic* and *DynamicSPR* investigated 3 different approaches to the number of ⟨fake⟩ messages sent per source period, and *ILPRouting* investigated 4 different delays to use. The problem is that it is hard to avoid trade-offs between these two parameters. For SLP to be provided, messages need to either avoid travelling the shortest path (increasing latency), or if they continue to travel along the shortest path then some additional broadcasts need to be used to obscure the real source (increasing the number of messages sent). Therefore, network maintainers will need to make a choice with respect to which metric they are willing to handle reduced performance in, in order to provide SLP.

8.7 Using Messages as a Proxy for Energy Consumption

Throughout the literature and also in this thesis, the number of messages sent by an SLP-aware routing protocol has been used as a proxy for the amount of energy that the protocol consumed. The rationale behind this is that sending and receiving messages tends to be among the most energy expensive tasks performed by a WSN as shown in Table 2.1. However, the time spent transmitting or receiving tends to be very small compared to the length of time that a sensor node will operate over. This is shown in Figure 6.16g, where fewer messages are sent per second as the source period increase, but the average power consumption per node decreases at a much slower rate such that the graphs appear flat. This means that reduction in the time spent transmitting has a small effect and the cost of leaving the CPU and radio idle causes much of the power consumption. This is reinforced when *DynamicSPR* is tested with a duty cycle in Figure 7.16g and Figure 7.17g where less power is consumed when fewer messages are sent. What this means is that the messages sent metric is not very accurate in capturing the energy savings of a protocol without a duty cycle. However, it becomes much more accurate when combined with an appropriate duty cycle protocol.

Chapter 9

Conclusions and Future Work

This thesis has presented ways to model, design and evaluate source location privacy protocols for wireless sensor networks that are near optimal in terms of their capture ratio. However, there are a number of ways in which the techniques presented could be expanded further. This chapter concludes with a summary of the work presented and the implications of it, and then will detail the future work opened by this research.

9.1 Conclusion

This thesis has focused on developing near optimal source location privacy-aware routing protocols for wireless sensor networks. These routing protocols are near optimal in the sense that the capture ratio is near 0%. The aim is to prevent an attacker who is following the protocol messages sent by a source node from capturing the asset the source has detected. The time the attacker has to capture the source is bound by a safety period which captures the amount of time an asset will be detected for. The capture ratio is calculated as the percentage of simulation or experimental instances in which the attacker reached the source within the safety period.

Initially an information theoretic model was developed to allow analysis of arbitrary routing protocol specified in terms of a routing matrix. This model led to the idea of *competing paths* which encompass the way a routing protocol should be perturbed to reduce the information an attacker trying to capture the source gains. Also presented was a way of representing routing protocols as a matrix and then comparing the information revealed to an attacker against a different routing protocol.

The information theoretic model focused on a spatial analysis of SLP-aware routing protocols, however, many protocols also have a temporal aspect to their privacy provision. To model the temporal nature of SLP-aware routing, the SLP

problem was modelled using integer linear programming. Using a solver, this model was capable of outputting an optimal SLP-aware broadcast schedule that considered the time at which messages should be broadcast. The optimal schedule was then used as inspiration when developing the *ILPRouting* protocol that trades off optimality for the ability to work in multiple scenarios. The *ILPRouting* protocol aims to group messages together to reduce the number of moves the attacker can make towards the source.

Next, a timing analysis was performed on when events in a technique that use fake sources to provide SLP would occur. This timing analysis provided a way for nodes in the network to determine parameters on-line rather than having them specified at compile time, which allowed the *Dynamic* protocol to react to changing network conditions. To reduce the energy cost of *Dynamic*, *DynamicSPR* was developed with the intention to allocate fewer fake sources that would otherwise counter the work other fake sources are doing. Finally, the timing analysis was extended in order to develop a duty cycle to further increase the energy savings of *DynamicSPR*. The duty cycle aimed to prioritise ⟨fake⟩ message delivery to ensure reliable provision of SLP.

There have been two main categories of outputs from this thesis, one practical and the other theoretical. The first is the three protocols developed which can be used to provide SLP-aware routing. These protocols are useful for network administrators that are deploying a network to monitor valuable assets. The performance has been investigated, so network administrators will be aware of the trade-offs when using these protocols. The high number of messages sent by *DynamicSPR* has also been offset by the duty cycle developed, making it practical for deployments. The second are the models (information theoretic analysis, ILP model, and fake source timing analysis) used to obtain these protocols. The models have been developed in such a way that they can be applied to other techniques, for example, the information theoretic analysis can be used on any protocol that can be specified in terms of the routing matrix. The ILP model can be modified in order to investigate ways to structure new techniques by altering the constraints and using new objective functions. The timing analysis of when events occur for fake source can also be applied to different fake source SLP techniques, such as Source Simulation. These models will assist in the development and comparison of future SLP-aware routing protocols.

In summary, the modelling of the SLP problem using different techniques has produced multiple near optimal SLP-aware routing protocols. These protocols achieve their near optimality by trading-off performance in other metrics, either the number of messages sent, or the latency of the messages. The models used to obtain

these routing protocols can be used to compare other SLP-aware routing protocols and assist in the development of new techniques with different trade-offs.

9.2 Future Work

There are a number of future avenues of work that could be undertaken by expanding on ideas presented in this thesis. This section will first describe a number of immediate areas in which the work presented could be explored further, and then present a number of larger issues regarding SLP provision in new areas.

9.2.1 Expand the Integer Linear Programming Model

In Chapter 5 the SLP problem was modelled using ILP in order to obtain an optimal schedule for a specific network configuration. This model is based upon a number of assumptions that limits the kinds of schedules that can can be produced. However, there are ways in which the ILP model could be altered to support generating different broadcast schedules. For example, the ILP model only allows generating schedules that route ⟨normal⟩ messages from the source to the sink. The model should be adjusted to allow scheduling ⟨fake⟩ message broadcasts. In doing so this opens up the model to generate techniques such as *Dynamic*, *DynamicSPR*, or fog routing [44]. This would require adding constraints on how ⟨fake⟩ messages are generated, as well as how they are routed. Further changes that should be investigated are to alter the routing constraints, which in many cases mean loosening the restrictions.

One possible example of a constraint that could be loosened is **ctR4** that states "Once a message is broadcasted by a node it is not broadcasted by that node again". This constraint could potentially be removed with the aim to obtain solutions such as fog routing, or solutions that route messages in rings [100] around the source before delivering them. Another routing constraint that could potentially be loosened is **ctR6** which states "All messages sent by the sources must reach the sink". The model could potentially be changed to only require a minimum number of messages be delivered (essentially a target delivery ratio). This would allow the model to trade off delivery ratio to provide SLP, whereas it currently requires a 100% delivery ratio.

In [175] a linear programming model was used to vary the communication range of nodes to investigate the impact it had on protecting an asset-detecting node. The ILP model presented in this thesis involved a communication range that can be specified as a parameter, meaning it is fixed for all nodes and the entire time the model is run for. An alternative would be to allow the range to be a decision variable per node that changes over time. This would allow each node to have a changeable range and for the ILP solver to assign values to it over time.

These are four potential modifications that could be made to the ILP model of SLP. Each combination could give rise to a different output that would provide insight into using different strategies to developed SLP-aware routing protocols. It is also worth attempting to have the ILP solver attempt to produce an output for larger network sizes and different topologies. However, the maximum size presented in this work was a 5x5 network as the solver required a large amount of RAM (nearly 100 GB) and several days to compute the solution. This means that producing output for larger networks is likely to be difficult. This is a problem inherent in ILP models with a large state, but extra computational power might help in obtaining an optimal solution [90] for larger state spaces.

9.2.2 Investigate Alternate ILP Objective Functions

If no changes were to be made to the way the ILP model of SLP is formulated in Chapter 5, then new schedules could be obtained by changing the objective function. In Chapter 5 a single objective function was investigated that aimed to maximise the attacker's distance from the source when the safety period expires. However, there are other objectives that are worth investigating using this model. These objectives will vary depending on the metric that network maintainers wish to optimise for and will likely give rise to different ways in which to perform the routing.

Another factor to consider is that many of these objectives have multiple components. For example, an objective might be to maximise the attacker's distance from the source whilst also trying to minimise another metric important to the network (such as energy usage). The ILP solver used to write the model provides a function called `staticLex`[1] that allows specifying a multi-criteria policy, where the first objective is most important and not worth a reduction in its value to improve subsequent objectives. An alternate is to specify a component of the objective function in such a way that produces a poor value when SLP is not provided. Several different objectives that are worth investigating will now be presented.

Prevent the Attacker From Finding the Source

Rather than trying to optimise for any specific metric, simply look for solutions in which the attacker does not find the source. This is a weaker objective than the one

[1] `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.1/ilog.odms.ide.help/OPL_Studio/opllang_quickref/topics/tlr_oplf_staticLex.html`

used in Chapter 5, where the attacker's distance from the source was maximised.

$$\begin{aligned} \text{minimise} \quad & \sum_{e \in A_{\mathbb{A}},\, e.v \in \mathcal{S}} \mathscr{P}_{\mathbb{A}}(\widehat{P_{safety}}, e) \\ \text{subject to} \quad & \text{Routing and Attacker Constraints.} \end{aligned} \tag{9.1}$$

Minimise Number of Messages Sent

Minimise the number of messages that all nodes send. This can be used as an approximation to minimising energy usage, because sending and receiving messages tends to be the most energy expensive task a WSN performs. A large cost is added when the attacker finds the source, to prevent these solutions from being chosen.

$$\begin{aligned} \text{minimise} \quad & \sum_{e \in A_{\mathbb{A}},\, e.v \in \mathcal{S}} 1000 \mathscr{P}_{\mathbb{A}}(\widehat{P_{safety}}, e) + \sum_{\substack{n \in V \\ m \in \mathcal{M} \\ \tau \in \mathcal{T}}} \mathscr{B}(n, m, \tau) \\ \text{subject to} \quad & \text{Routing and Attacker Constraints.} \end{aligned} \tag{9.2}$$

Minimise the Number of Attacker Moves

When the attacker starts at the sink, the smallest possible number of moves it can make is 1 when it moves away from the sink. If no more attacker moves are made and the source is not directly connected to the sink, then SLP should be provided. This objective function should show how to minimise these moves.

$$\begin{aligned} \text{minimise} \quad & \sum_{\substack{e \in A_{\mathbb{A}},\, e.u \neq e.v \\ m \in \mathcal{M} \\ \tau \in \mathcal{T}}} (\mathscr{B}(e.v, m, \tau) \wedge \mathscr{P}_{\mathbb{A}}(\tau, e)) \\ \text{subject to} \quad & \text{Routing and Attacker Constraints.} \end{aligned} \tag{9.3}$$

Maximise the Number of Attacker Moves

Whist minimising the number of attacker moves is a way to provide SLP, an attacker will be aware that it is losing information about messages being sent. By maximising the number of attacker moves, the attacker should respond to every possible message — but in a way that prevents it from finding the source.

$$\begin{aligned} \text{maximise} \quad & -\sum_{e \in A_{\mathbb{A}},\, e.v \in \mathcal{S}} 1000 \mathscr{P}_{\mathbb{A}}(\widehat{P_{safety}}, e) + \sum_{\substack{e \in A_{\mathbb{A}},\, e.u \neq e.v \\ m \in \mathcal{M} \\ \tau \in \mathcal{T}}} (\mathscr{B}(e.v, m, \tau) \wedge \mathscr{P}_{\mathbb{A}}(\tau, e)) \\ \text{subject to} \quad & \text{Routing and Attacker Constraints.} \end{aligned} \tag{9.4}$$

Minimise the Message Latency

For some WSN applications latency is an important performance attribute. Finding a solution that provides SLP, whilst minimising latency will be important.

$$\texttt{LAT}(m) = \min_{\substack{q \in \mathcal{Q} \\ n \in \overrightarrow{N}(q) \\ \tau \in \mathcal{T}}} \begin{cases} t & \mathscr{B}(n, m, t) \\ 1000 & \text{otherwise} \end{cases} - \min_{\substack{s \in \mathcal{S} \\ \tau \in \mathcal{T}}} \begin{cases} t & \mathscr{B}(s, m, t) \\ 1000 & \text{otherwise} \end{cases} \tag{9.5}$$

$$\begin{aligned} \text{minimise} \quad & \sum_{e \in A_{\mathbb{A}},\, e.v \in \mathcal{S}} 1000 \mathscr{P}_{\mathbb{A}}(\widehat{P_{safety}}, e) + \sum_{m \in \mathcal{M}} \texttt{LAT}(m) \\ \text{subject to} \quad & \text{Routing and Attacker Constraints.} \end{aligned} \tag{9.6}$$

Conclusion

There are many different objective functions that could be used to find optimal broadcast schedules for different objectives. The objectives that are important depend on the context in which SLP is deployed. Multiple objectives can be optimised for using this formulation, for example, both latency and the number of messages sent could be optimised for by specifying certain weights for the two components of the objective function. That the many different objectives can be calculated points to the flexibility of the ILP model of SLP.

9.2.3 New Configurations

This work has focused on situations in which there is a single attacker trying to capture a single source which is reporting data back to a single sink. By focusing on this situation the problem for optimality for a single attacker, source, and sink could be well understood. However, there are likely to be a wide range of different configurations that will be used in real world scenarios that need further investigation.

If multiple sources are present in the network then there will be additional challenges to providing SLP compared to when a single source is present. For example, both fake sources [93] and phantom routing [59] perform worse when multiple sources are present. This means that both *Dynamic* and *DynamicSPR* are likely to perform poorly with multiple sources. However, *ILPRouting* is likely to have better performance due to it performing retransmissions. The problem is the multiple sources can exist in positions where the messages sent by each source competes and undoes the SLP-provision work by the other sources. Future work will need to ensure that protecting one source does not lead to the location of another source from being revealed. Some options may including using clustering techniques

to aggregate information about a herd of animals into what appears to be a single source. Alternately the routes SLP-providing messages take should be allocated to ensure that they interact with messages from other source in a way that increases the attacker's uncertainty of which direction to explore.

In this work it is assumed that there is a single sink located at the centre of the network. In some situations multiple sinks may be used to collect data from the WSN. In this situation the techniques presented could be modified to take advantage of this. For example, *ILPRouting* could alternate sending ⟨normal⟩ messages to two different sinks. By sending to a different sink the route could avoid the area it previously travelled to, reducing the chance an attacker would be in a location to overhear each message. However, multiple sinks would not be important for *Dynamic* and *DynamicSPR* because flooding is used and all nodes in the network receive the ⟨normal⟩ and ⟨fake⟩ messages.

If multiple sinks are used as part of a strategy to provide SLP then it will be important to place them in good positions. Determining an optimal placement of sinks and the way in which messages are routed to them are both likely to be difficult problems to solve. Techniques like ILP might be able to offer insight as to where in the network the sinks should be located, along with how to route messages to the different sinks.

9.2.4 Dealing with Different Attackers

In this thesis, a single distributed eavesdropping attacker that follows the network traffic to capture the source has been assumed. The attacker fails to capture the source if it is not co-located within a specified time, termed the safety period. There are a number of different capabilities this attacker could be equipped with. This section will describe those alternate options.

Message Detection Ability

In Section 3.5 it was chosen to assume attackers have the ability to perfectly determine whether a message is new or not. If an attacker does not have this ability then *Protectionless* flooding would be able to defeat the attacker, as it would follow the flood of messages away from the source. Assuming the attacker has the ability to perfectly determine if a message is new or not is the worst case behaviour, but is unlikely to be realistic as messages are encrypted and their contents change with each hop. An attacker might be capable of of detecting the probability that a message is new. However, the impact of moving in response to certain packets that it has already seen will reduce the attacker's ability to capture the source.

An alternative to this would be to specify the speed at which attackers move. Once an attacker receives a message it would take them some time to move to the node that sent the message. During this time period the attacker would not be able to eavesdrop messages. This approach was not used in this work because it would require setting a speed limit for attackers, thus limiting the generality of the attacker model.

Multiple Attackers

In this work source location privacy techniques were developed against a single mobile attacker. The aim of this work was to present near optimal techniques for a single attacker and multiple attackers were out of its scope. It is possible that multiple attackers physically present in the network may try to violate SLP. Due to their larger combined visible area, multiple attackers are likely to prove more effective against SLP techniques. However, there are a number of disadvantages multiple cooperating attackers could face. If the attackers communicate wirelessly then that is information the WSN could use to detect their presence and adjust how SLP is being provided, or the network maintainers could use to detect the attackers. For example, the WSN could avoid routing messages through areas in which the attackers have been detected. An alternate non-technological aspect to consider is that when attackers are in different positions then rangers or police have a greater chance to find one of them. This means that attackers will have to trade-off a greater risk of detection with the ability to capture the source in less time.

Recording History

The attacker presented in this work does not maintain any history about the process of it attempting to locate the source. The reason behind this is that attempting to use history in certain ways is likely to lead to worse performance. A simple example is the attacker choosing not to move to locations it has previously visited. This approach can easily lead an attacker to get trapped at a node at which it has visited all neighbours. This is unlikely to be realistic behaviour for real attackers that are trying to be modelled. A more complex example, would be if an attacker waits at one location to receive multiple messages and then make a move based on the messages received at that location, it will have increased the time it takes to make a single move. This strategy will reduce the number of moves the attacker can make, reducing its chance of capturing the source. This is also an issue with the *backtracking* attacker that backtracks to the last known location with possible options when it reaches a dead end, as it looses time it could be exploring the network by backtracking along the route previously visited.

Alternately, the attacker could build up a history of messages received and moves taken over time as they respond to messages. By continuing to immediately move, the attacker does not gain rich information about its current location, but it allows it to gain sparse information across the network. An analysis of this information could be performed to gain insight into the likely direction of the real source and possibly fake sources if present. However, because of the low quantity of data and the high amount of noise present, this analysis is likely to be hard to perform.

Temporal Analysis

Another action that an attacker could perform is a temporal analysis of when it receives messages at a given node. By doing so the attacker could aim to detect if fake sources are present and attempt to estimate their direction. A problem with this is that WSNs are unreliable, which causes messages to not always take the shortest path. An example of this is shown in Figure 7.1, where the different latencies of a packet travelling to nodes specific hops away from it is shown. There is often a difference of 20 ms to 40 ms in the time it takes messages to arrive. Experiments have previously shown that it takes about 6 ms for a message to be processed and forwarded by a node. This noise makes an attacker doing timing analysis more difficult, as the attacker will rely on timings to gauge the distance of real and fake sources among other properties of the network.

Proactive Attacker

This thesis has focused on an attacker that moves in reaction to receiving a new message. Alternatively the attacker could choose to proactively move to a different location without receiving a message. In some ways this is the ideal attacker (from the network's perspective) as the aim of SLP is to make using the context from messages a worse choice than performing an exhaustive search of the network. However, it is likely that in many circumstances an attacker that proactively moves based on its gathered knowledge will perform well. For example, an attacker could wait at a location to gather multiple messages and then make a large move in one direction. Other strategies may also perform well, so it is worth investigating how existing techniques perform against a proactive attacker, and how to develop new techniques to defend against one.

Game Theory Perspective

It could be possible to consider the attacker's behaviour and implementation of an SLP-aware routing protocol from a game theoretic perspective. The optimal outcome of a developed routing protocol would be if the attacker ignores it and instead performs a random search of the network. A routing protocol might be able to be developed such that the utility to the network maintainer is balanced where the attacker doesn't find the source and the energy cost is reasonable. It may be better to analyse the objective of the attacker's distance from the source, rather than if the attacker has captured the source, for similar reasons as to why that was the objective for *ILPRouting*.

Summary

Overall, there are a number of additional capabilities that an attacker could be provided with. One of the downsides of any additional capability is if it takes times to execute. With a non-zero time to execute this will delay the attacker on its way to the source and decrease the chance that the attacker will capture the source within the safety period. Therefore, it is often better for the attacker to perform simple analysis with a small time cost (ideally zero time cost). Future work should investigate these intuitions by testing different SLP-aware routing protocol with different attacker strategies. Future work should also consider how modelling of real poachers has been performed [84] and use that to inform the modelling of attackers in the SLP problem.

9.2.5 Providing SLP in Highly Mobile Networks

The problem of source location privacy has thus far been focused on protecting the location context of a mobile asset being monitored by a stationary WSN. However, there are instances of highly mobile networks in which it may be necessary to provide SLP.

In 2011 authorities in Egypt shut down the countries internet in response to mass protests against the government[2] with the aim to limit the flow of information. One aspect of this was to hinder the ability of the protesters to organise themselves. In 2014 a similar situation occurred in Hong Kong where authorities selectively blocked certain communication platforms. In response to this the protesters turned to a peer-to-peer communication app called FireChat[3] to work around the communication restrictions. As the app relied on mesh networking it was able to provide

[2] `https://www.nytimes.com/2011/01/29/technology/internet/29cutoff.html`

[3] `edition.cnn.com/2014/10/16/tech/mobile/tomorrow-transformed-firechat`

communications without being vulnerable to centralised filtering. However, it may be the case that the mesh networking communications are vulnerable to context privacy leakages. For example, as messages are sent by organisers of the protest the authorities may want to find their location to arrest them. This could be achieved by having an agent among the protesters following the wireless signals using a directional antenna. As a directional antenna is likely to be noticeable, multiple agents could be used to triangulate a signal's source hop-by-hop. Alternately authorities might choose to leverage telecommunications infrastructure to obtain a global view of network traffic.

This sort of traffic analysis attack to find the location of the organisers of a protest might be difficult, as there are many other members of the network who are also sending their own messages. So future work should be to first investigate if a mobile mesh network is vulnerable to a traffic analysis attack to determine the message source. If the network is vulnerable then it is likely that different techniques will need to be developed, compared to the techniques used in stationary WSNs.

Another kind of highly mobile network in which SLP is worth providing are VANETs [49, 160]. Location privacy is important to prevent other road users from knowing a vehicles origin, route, and destination. As a vehicle leaks identity in certain ways that cannot be protected (by the number plate), location privacy is not attempted to be provided against authorities. As the route messages take are dependent on the route that cars travel, other techniques are used to provide SLP. Many of the techniques focus on using temporary pseudonyms, whereby changing pseudonyms periodically decorrelates a vehicle from the route it has taken. Several approaches to changing the pseudonyms have been approached, including doing so based on the velocity of the vehicle [28], changing at social spots [116], and using game theory to encourage rational behaviour [54]. Routing based techniques from WSNs may be applicable to VANET SLP and deserves future investigation.

9.2.6 Summary

Overall there are a large number of avenues for future work, including providing SLP in WSNs under more challenging scenarios. But there are also a number of areas (such as VANETs or ad-hoc messaging apps) in which the techniques and experience from providing SLP in WSNs could also be applied.

Appendix A

Result Reproduction

The results presented in this thesis were obtained with reproducibility in mind. All results obtained from simulations (both TOSSIM and COOJA) are deterministic for a given random seed. Only the results obtained from the testbeds are not expected to behave in exactly the same way due to wireless noise that cannot be controlled, however, with sufficient repeats the results should be similar.

As part of this work a simulation framework used to run simulations, gather results, and analyse them was developed. The source code for this framework and the algorithms presented here can be found at (1). The modified version of TinyOS version 2.1.2 used for developing these algorithms can be found at (2). The source code for the ILP model presented in Chapter 5 can be found at (3).

1. `bitbucket.org/MBradbury/slp-algorithms-tinyos`
2. `github.com/MBradbury/tinyos-main/tree/bradbury_2_1_2`
3. `bitbucket.org/MBradbury/slp-attacker-ilp/raw/a4e326e/SLP/SLP.mod`

The raw results generated by TOSSIM and COOJA can be found at the following location: `doi.org/10.5281/zenodo.1209158`. This includes TOSSIM results for *Protectionless*, *Dynamic*, *DynamicSPR*, *ILPRouting*, and *Static*. It also includes COOJA results for *Protectionless*, *DynamicSPR*, *DynamicSPR* with duty cycling, and *DynamicSPR* with TinyOS LPL. The raw results obtained from running the code on the FlockLab testbed can be found at the following location: `doi.org/10.5281/zenodo.1229054`. This includes the profiling performed on FlockLab as well as results for *Protectionless*, *DynamicSPR*, and *DynamicSPR* with duty cycling.

Appendix B

Conditional Entropy and Mutual Information

In Chapter 4 the entropy and Jensen-Shannon divergence was used to compare $\mathcal{R}_{\mathcal{N}}$ and $\mathcal{R}_{\mathcal{S}}$. The aim of creating a properly perturbed path was to create a route longer than the safety period the attacker would follow with no uncertainty. Using conditional entropy and mutual information there are further analyses that can be performed on a single routing matrix to understand the routing uncertainty. For future work, increasing the uncertainty of a routing matrix, in ways similar to differential privacy, may be another way to provide SLP.

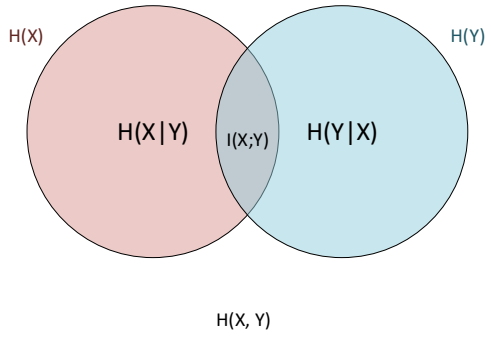


Figure B.1: Relation of mutual information

Mutual information between transitions taken at two times (λ and μ), given the attacker starts at q, is denoted by $I(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q)$, which is defined in Equation B.1 and shown in Figure B.1. To calculate $I(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q)$ the entropy on $\mathcal{X}_\lambda$ denoted by $H(\mathcal{X}_\lambda \mid \mathbb{A}_{\mathcal{X}0} = q)$ is required, which is shown in Equation 4.2. The log component uses base 2 so the units of entropy are in bits.

$$I(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q) = H(\mathcal{X}_\lambda \mid \mathbb{A}_{\mathcal{X}0} = q) - H(\mathcal{X}_\lambda \mid \mathcal{X}_\mu, \mathbb{A}_{\mathcal{N}0} = q) \tag{B.1}$$

$H(\mathcal{X}_\lambda \mid \mathbb{A}_{\mathcal{X}0} = q)$ denotes the marginal entropy on $\mathcal{X}_\lambda$ when the attacker starts at q and $H(\mathcal{X}_\lambda \mid \mathcal{X}_\mu, \mathbb{A}_{\mathcal{X}0} = q)$ denotes the conditional entropy on $\mathcal{X}_\lambda$ given $\mathcal{X}_\mu$ when the attacker starts at q. To obtain $I(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q)$, $H(\mathcal{X}_\lambda \mid \mathcal{X}_\mu, \mathbb{A}_{\mathcal{X}0} = q)$ needs to be calculated, where n is a transition the attacker could take at λ and f is a transition the attacker could take at μ.

$$\begin{aligned}
&H(\mathcal{X}_\lambda \mid \mathcal{X}_\mu, \mathbb{A}_{\mathcal{X}0} = q) \\
&= \sum_{f \in \Gamma} \Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) H(\mathcal{X}_\lambda \mid \mathcal{X}_\mu = f, \mathbb{A}_{\mathcal{X}0} = q) \\
&= \sum_{n \in \Gamma} \sum_{f \in \Gamma} \Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) \log_2 \frac{\Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right)}{\Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right)}
\end{aligned} \tag{B.2}$$

Substituting Equation B.2 into Equation B.1, mutual information is then given by Equation B.3. When calculating $I\left(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q\right)$, if $\Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) = 0$, or $\Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) = 0$, or $\Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) = 0$ that combination of transitions is treated as having a probability of 0 when calculating I.

$$\begin{aligned}
&I(\mathcal{X}_\lambda; \mathcal{X}_\mu \mid \mathbb{A}_{\mathcal{X}0} = q) \\
&= \sum_{n \in \Gamma} \sum_{f \in \Gamma} \Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) \log_2 \frac{\Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right)}{\Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) \Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right)}
\end{aligned} \tag{B.3}$$

The mutual information between times λ and μ for a routing matrix indicates much the uncertainty of $\mathcal{X}_\lambda$ decreases when $\mathcal{X}_\mu$ is known (and vice versa). When $\lambda = \mu$ the uncertainty decrease is equal to $H(\mathcal{X}_\lambda)$ as there is no longer any uncertainty. When the entropy is 0, the mutual information is also 0 as there can be no decrease in uncertainty. This means that mutual information indicates the decrease of uncertainty of taking a transition at λ when the transition taken at μ is known.

The joint probability of the attacker taking transition n at λ and f at μ, given that the attacker starts at q, is represented by $\Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right)$. This probability is defined in Equation B.4.

$$\begin{aligned}
&\Pr\left(\mathcal{X}_\lambda = n, \mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) \\
&= \begin{cases}
\Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) & n = f \wedge \lambda = \mu \\
0 & n \neq f \wedge \lambda = \mu \\
\Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}\lambda} = n_2\right) \Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}0} = q\right) & \lambda < \mu \\
\Pr\left(\mathcal{X}_\lambda = n \mid \mathbb{A}_{\mathcal{X}\mu} = f_2\right) \Pr\left(\mathcal{X}_\mu = f \mid \mathbb{A}_{\mathcal{X}0} = q\right) & \mu < \lambda
\end{cases}
\end{aligned} \tag{B.4}$$

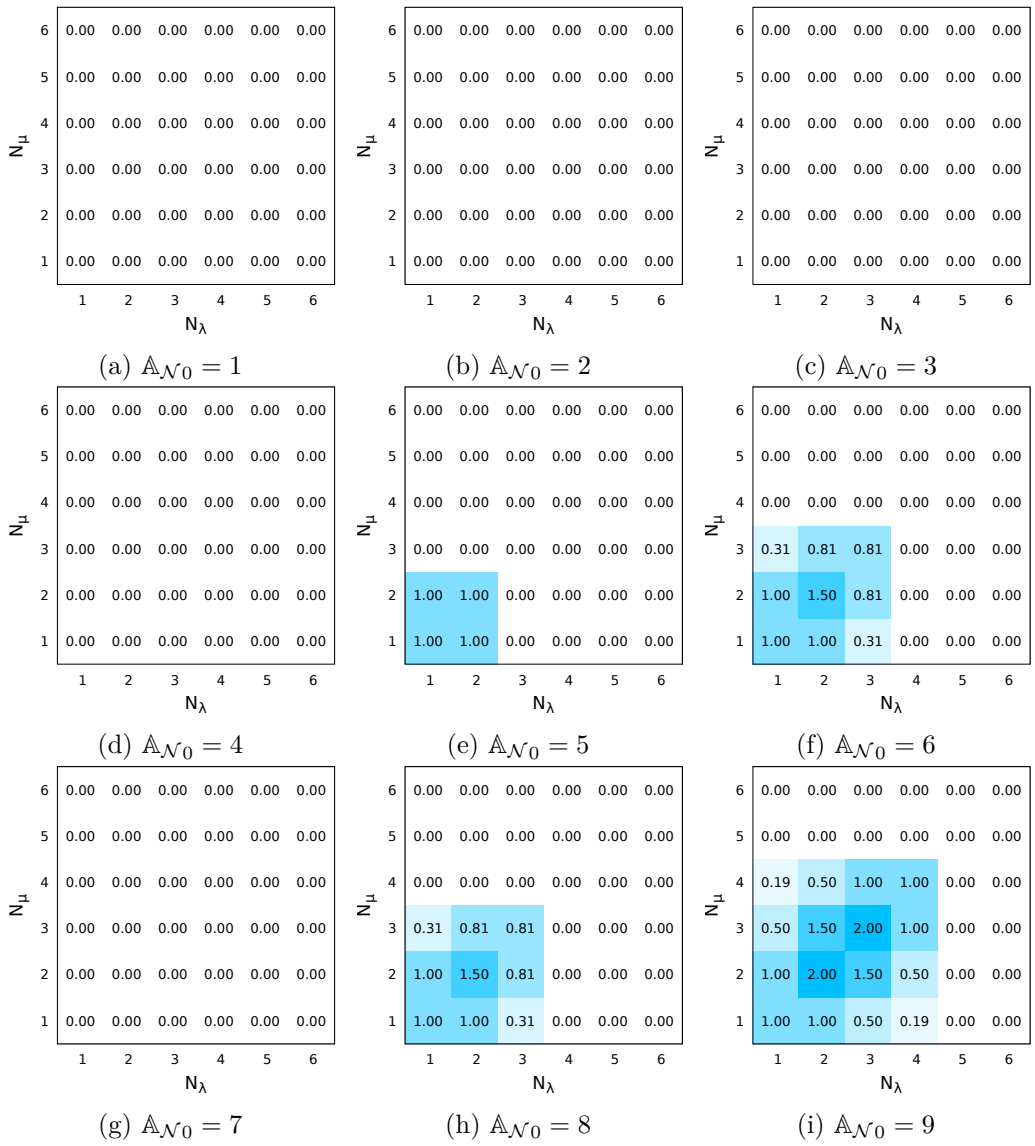


Figure B.2: Mutual Information between $\mathcal{N}_\lambda$ and $\mathcal{N}_\mu$ $I(\mathcal{N}_\lambda; \mathcal{N}_\mu \mid \mathbb{A}_{\mathcal{N}0} = q)$

Figure B.2 shows the mutual information between different times in $\mathcal{R}_\mathcal{N}$ and Figure B.3 shows the conditional entropy. There are no graphs showing the mutual information or conditional entropy for $\mathcal{R}_\mathcal{S}$ as the entropy is 0 which means that both the mutual information and conditional entropy would also be 0. When the attacker starts at nodes 1, 2, 3, 4, or 7 the mutual information and conditional entropy is also 0 for $\mathcal{R}_\mathcal{N}$ because there is no uncertainty about the path that the attacker takes from those starting nodes to the source.

When the transition at time μ is known and μ is after the attacker has captured the source, the conditional entropy is equal to the entropy as knowing the attacker stays at the source does not reveal information about the earlier transitions.

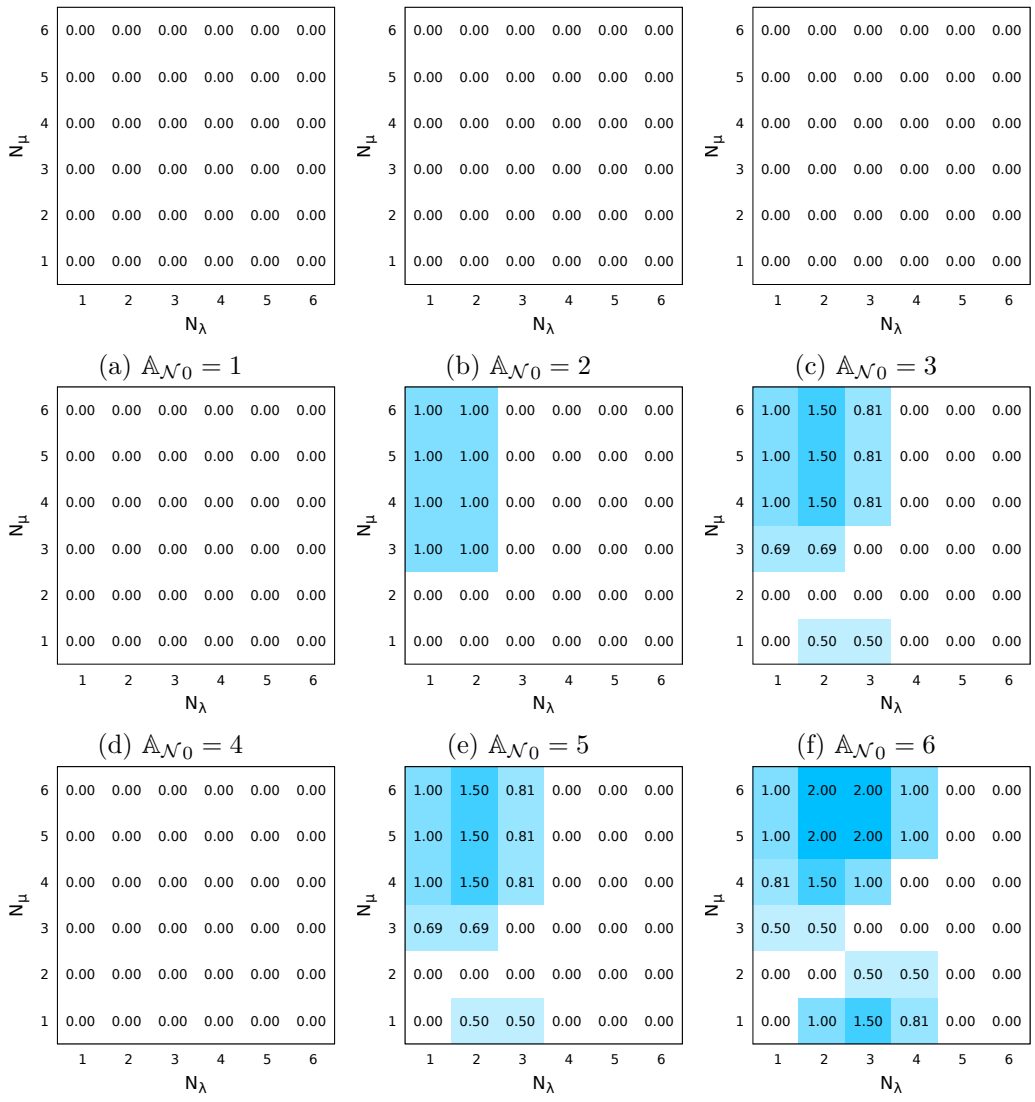


Figure B.3: Conditional Entropy $H(\mathcal{N}_\lambda \mid \mathcal{N}_\mu, \mathbb{A}_{\mathcal{N}0} = q)$

However, knowledge of which transition is taken at μ does affect the uncertainty of the transitions that could be taken at λ.

In conclusion, conditional entropy is useful in understanding how much information an attacker can gain about taking a transition at λ if it knows the transition taken at μ, and mutual information is useful in understanding the dependency between taking transitions at two times. However, these measures are only useful for understanding how transitions at different times in one routing matrix relate. Measures such as the Jensen-Shannon divergence should be used to compare two different routing matrices.

Appendix C

Dynamic: Extra Results

As stated in Subsection 6.3.3 the *Dynamic* algorithm has previously been published with different strategies to calculate the number of ⟨fake⟩ messages to send [24, 25]. Since then the implementation of *Dynamic* has been modified to better handle certain errors that would previously have led to lost messages. In this thesis *Dynamic* was presented with three strategies that send fewer messages than in the two strategies presented in the publications. In this appendix the two old strategies will be described and results will be compared between the old and new strategies.

C.1 Strategies

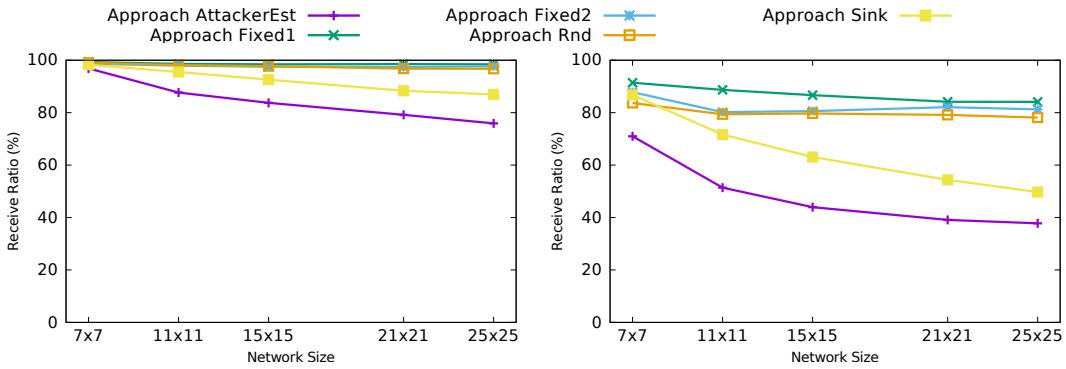
C.1.1 Pull From Attacker

This approach (`AttackerEst`) aims to pull the attacker back from its estimated position assuming no SLP protection. In this case it is assumed that TFS nodes propagate away from the source at the same rate that an attacker moves towards the source. This can be a reasonable assumption when the duration of the TFS is equal to the source period. This means that a TFS will need to send twice the $\Delta_{sink}(j)$ to dissuade the attacker back from its position. $\Delta_{sink}(j)$ message are needed to pull back from the sink to the TFS, another $\Delta_{sink}(j)$ messages are needed to pull the attacker from its position back to the sink.

$$\#_{\mathcal{F}}(j) = \max_{\perp}(1, 2\Delta_{sink}(j)) \tag{C.1}$$

C.1.2 Pull From Sink

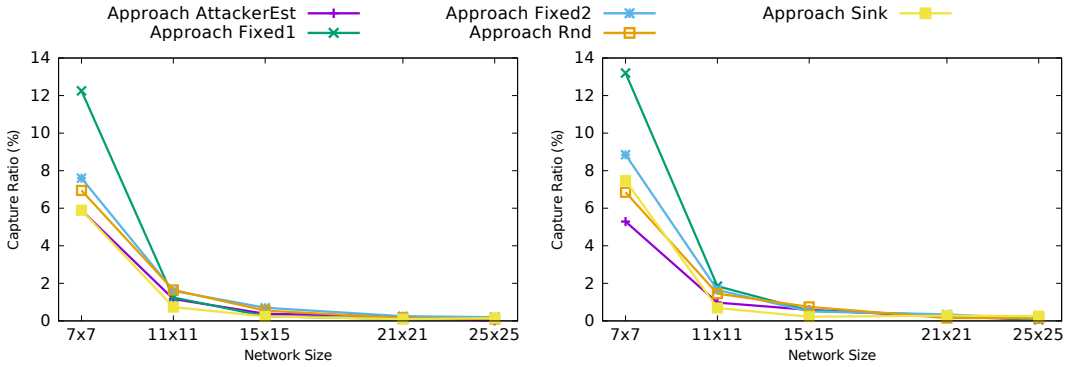
This approach (`Sink`) aims to pull the attacker back from the sink's location. This approach is less aggressive compared to the previous approach and is not as focused on trying to pull the attacker all the way back, but instead keeping it in a location



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure C.1: Results showing the percentage of ⟨normal⟩ messages received at sink.



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

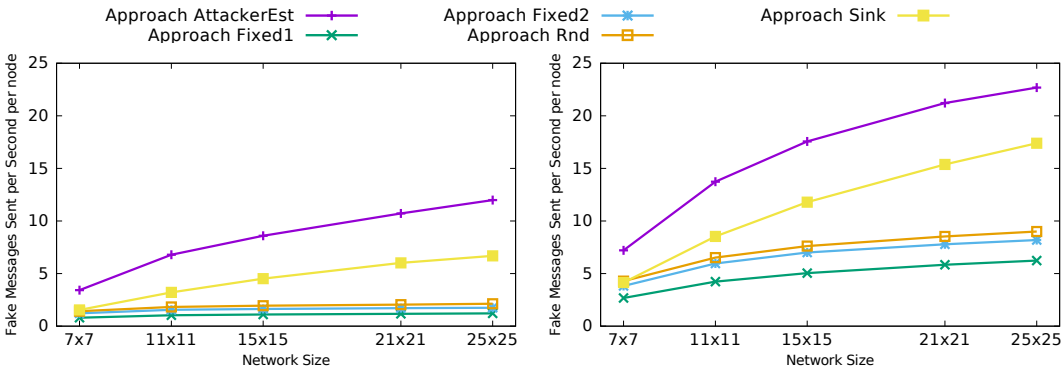
Figure C.2: Results showing the capture ratio.

between the TFS and the source. An important benefit of this approach is that $\Delta_{src}(j) - \Delta_{ss}$ is used to calculate the sink distance. This means that a TFS closer to the source will send fewer messages than TFS further away.

$$\#_{\mathcal{F}}(j) = \max_{\perp}\left(1, \begin{cases} \Delta_{sink}(j) & \text{if } \perp \in \{\Delta_{src}(j), \Delta_{ss}\} \\ \Delta_{src}(j) - \Delta_{ss} & \text{otherwise} \end{cases}\right) \tag{C.2}$$

C.2 Results

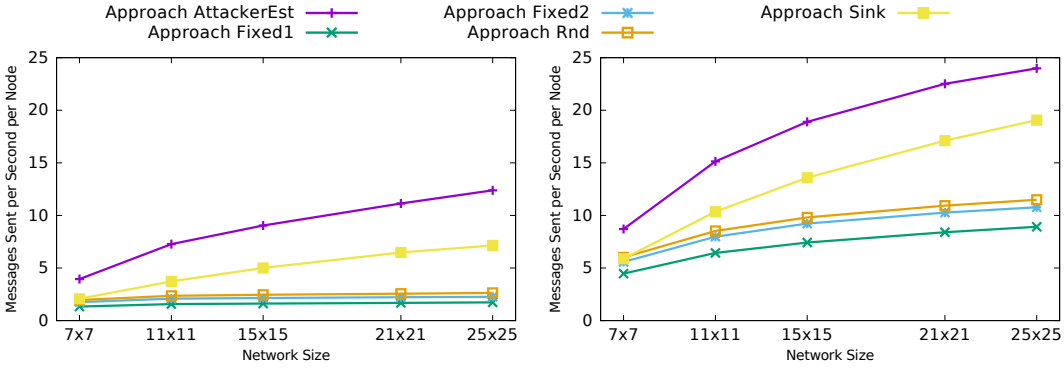
The results for the two old *Dynamic* approaches are presented here. Only results for a source period of 2.0 and 0.25 are presented. These results show that the `AttackerEst` and `Sink` approaches are as good or better in terms of capure ratio than the `Fixed1`, `Fixed2`, or `Rnd` approaches to the number of ⟨fake⟩ messages to send. This is because `AttackerEst` and `Sink` send a much larger number of ⟨fake⟩ messages (sometime more than 100% extra). But sending this many messages has a



(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure C.3: Results showing the average number of ⟨fake⟩ messages sent per node per second.



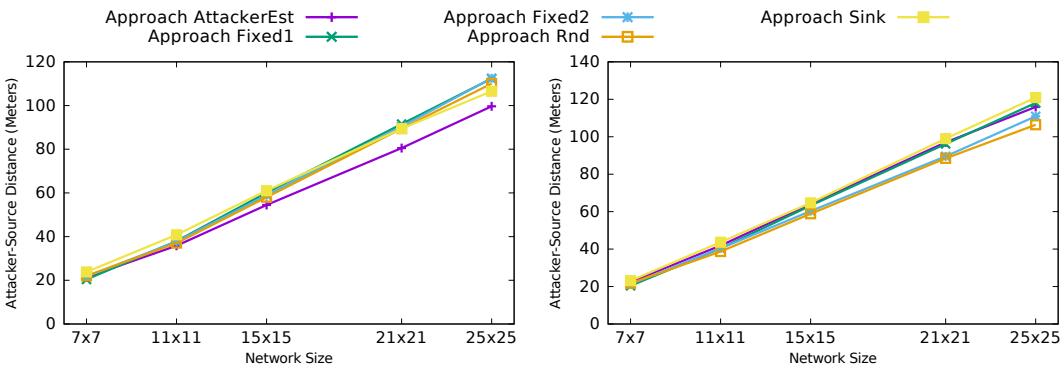
(a) Source Period 2.0 seconds

(b) Source Period 0.25 seconds

Figure C.4: Results showing the average number of messages sent per node per second.

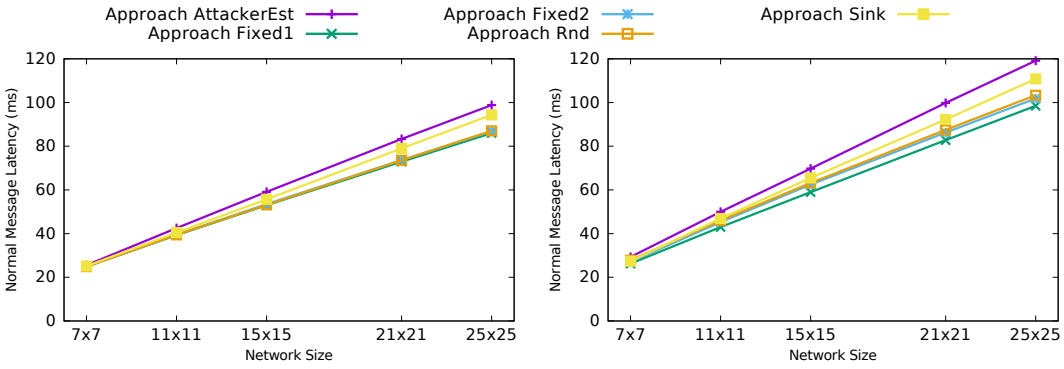
high cost in terms of energy cost, and it also leads to poor performance in terms of delivery ratio where `AttackerEst` delivers half the value of `Fixed1`. Message delivery latency tends to be higher for `AttackerEst` and `Sink`. These two approaches are also more capable of pulling the attacker back to further locations when the source period is high.

Overall, the cost in terms of number of messages sent and the very low delivery ratio makes the `AttackerEst` and `Sink` approaches much worse than `Fixed1`, `Fixed2`, or `Rnd`. The reason that different results were obtained in this thesis and in [25] was because the protocol's implementation had been improved to better handle certain failure conditions between publication of [25] and this work. Because of the improved reliability `AttackerEst` and `Sink` ended up sending many more messages than they used to. This is why results for `Fixed1`, `Fixed2`, and `Rnd` were presented in the main thesis instead of `AttackerEst` and `Sink`.



(a) Source Period 2.0 seconds (b) Source Period 0.25 seconds

Figure C.5: Results showing the attacker's distance from the source.



(a) Source Period 2.0 seconds (b) Source Period 0.25 seconds

Figure C.6: Results showing the time it took a normal message to reach the sink.

Appendix D

ILPRouting: Direct to Sink Results

In Section 5.5 *ILPRouting* results for different probabilities of sending messages directly to the sink were omitted as they were said to provide worse performance than the ones presented. This appendix will present those results and demonstrate that they are worse than the ones presented in the results section of Chapter 5.

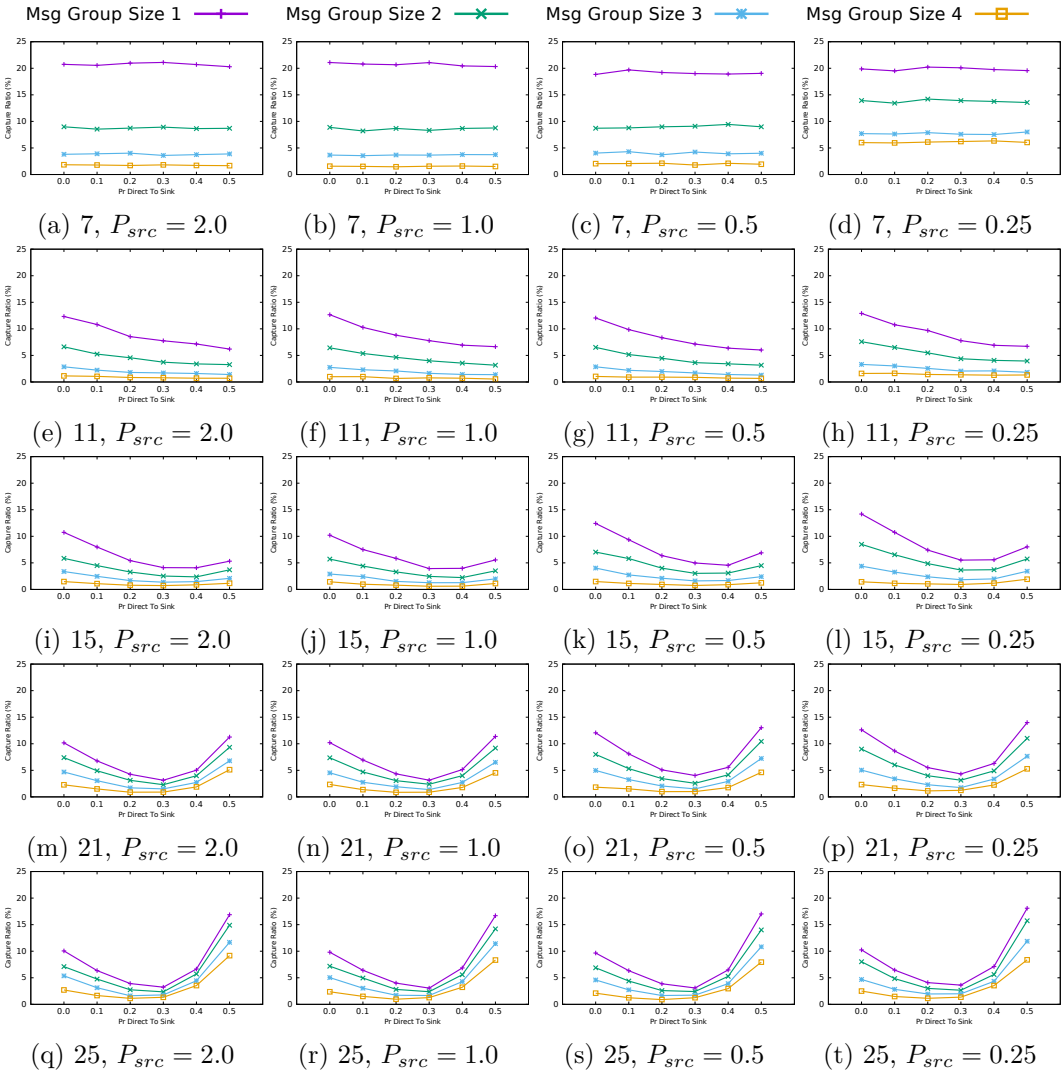
D.1 Capture Ratio

It was expected that as more messages travel directly towards the sink the capture ratio would increase. However, a number of different capture ratio patterns are observed when varying the probability messages travel directly towards the sink and other parameters.

For the very small network of size 7 varying this parameter had no effect on the capture ratio. This is because the first 10 messages sent by the source will never travel directly to the sink and will always try to travel around the sink to approach it from behind. This parameter was set to allow some time for the attacker to be pulled away and also because the direct to sink behaviour was only observed for later messages in the ILP model output. As the size 7 network is small so is the safety period, and the safety period is small enough that not enough messages are sent for this parameter to have an effect.

For small networks sized 11 and 15, a higher probability of sending directly to the sink led to a lower capture ratio.

For large networks sized 21 an 25, a there is an optimum of the parameter around 20% or 30% as a higher value of 40% starts to lead to a higher capture ratio.



(a) 7, $P_{src} = 2.0$ (b) 7, $P_{src} = 1.0$ (c) 7, $P_{src} = 0.5$ (d) 7, $P_{src} = 0.25$

(e) 11, $P_{src} = 2.0$ (f) 11, $P_{src} = 1.0$ (g) 11, $P_{src} = 0.5$ (h) 11, $P_{src} = 0.25$

(i) 15, $P_{src} = 2.0$ (j) 15, $P_{src} = 1.0$ (k) 15, $P_{src} = 0.5$ (l) 15, $P_{src} = 0.25$

(m) 21, $P_{src} = 2.0$ (n) 21, $P_{src} = 1.0$ (o) 21, $P_{src} = 0.5$ (p) 21, $P_{src} = 0.25$

(q) 25, $P_{src} = 2.0$ (r) 25, $P_{src} = 1.0$ (s) 25, $P_{src} = 0.5$ (t) 25, $P_{src} = 0.25$

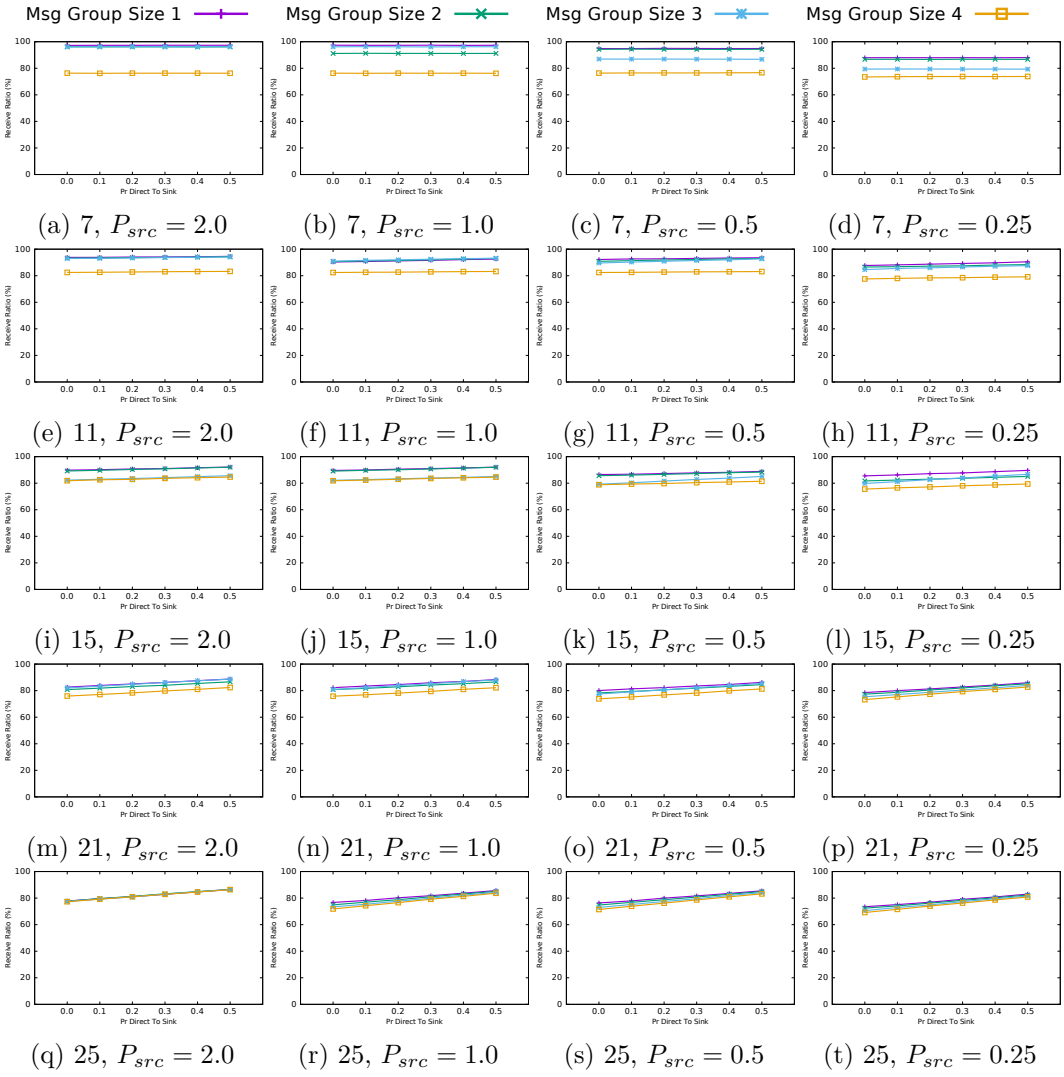
Figure D.1: *ILPRouting* results showing the capture ratio for different probabilities of sending messages directly to the sink.

D.2 Receive Ratio

A larger receive ratio is observed with a higher probability of sending messages directly towards the sink. This is because messages take the shortest path more often and because fewer hops need to be traversed there is a lower probability of a message being lost.

D.3 Messages Sent per Second

There is no change in the number of messages sent per second for the majority of parameter combinations. For the large networks of size 21 and 25 when the



(a) 7, $P_{src} = 2.0$ (b) 7, $P_{src} = 1.0$ (c) 7, $P_{src} = 0.5$ (d) 7, $P_{src} = 0.25$

(e) 11, $P_{src} = 2.0$ (f) 11, $P_{src} = 1.0$ (g) 11, $P_{src} = 0.5$ (h) 11, $P_{src} = 0.25$

(i) 15, $P_{src} = 2.0$ (j) 15, $P_{src} = 1.0$ (k) 15, $P_{src} = 0.5$ (l) 15, $P_{src} = 0.25$

(m) 21, $P_{src} = 2.0$ (n) 21, $P_{src} = 1.0$ (o) 21, $P_{src} = 0.5$ (p) 21, $P_{src} = 0.25$

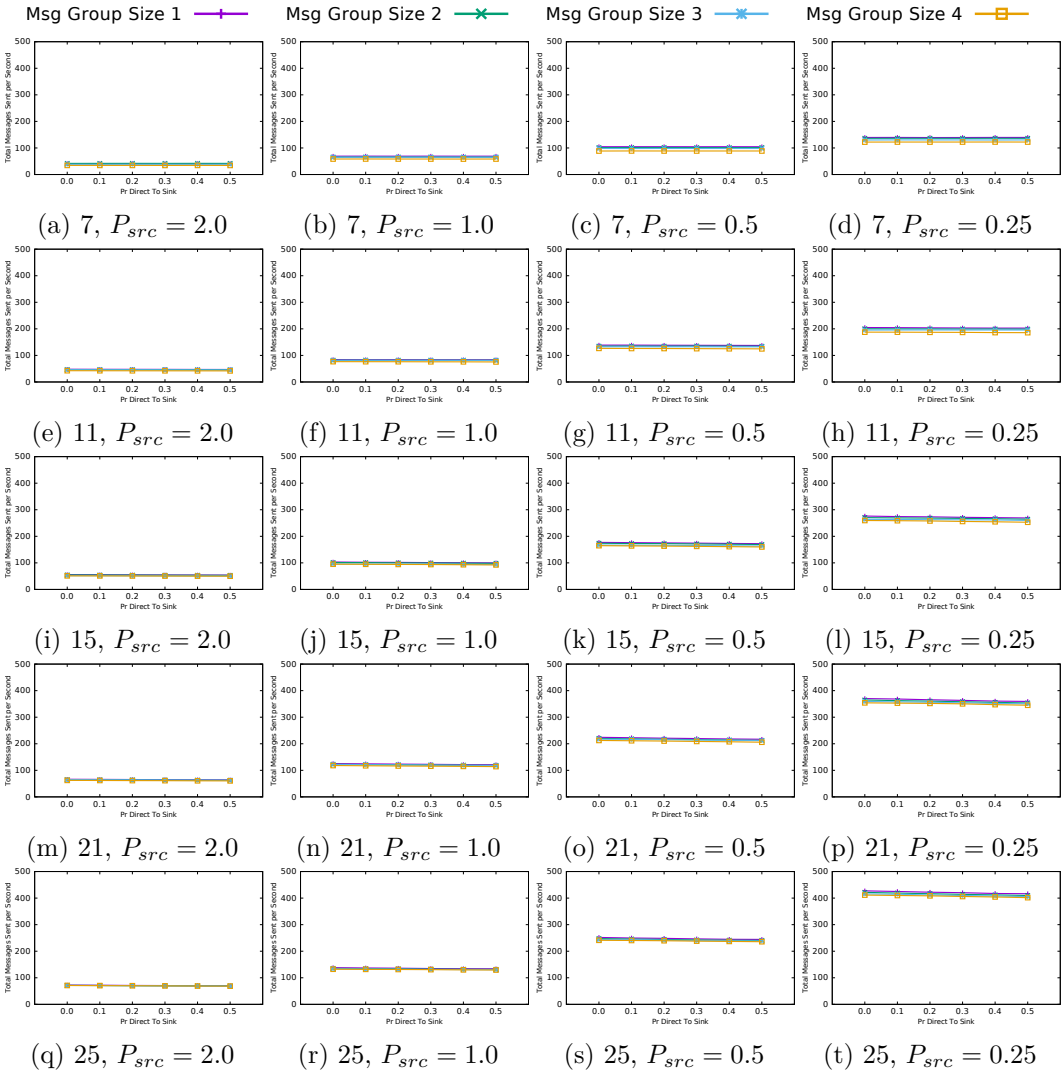
(q) 25, $P_{src} = 2.0$ (r) 25, $P_{src} = 1.0$ (s) 25, $P_{src} = 0.5$ (t) 25, $P_{src} = 0.25$

Figure D.2: *ILPRouting* results showing the receive ratio for different probabilities of sending messages directly to the sink.

source period is 0.25, it can be observed that there is a slight decrease for higher probabilities of sending messages directly to the sink. This is because on these larger networks, messages that avoid the sink can take longer routes when avoiding the sink compared to smaller networks. So there is a greater potential for message sent reductions in these larger networks.

D.4 Latency

As the probability a message is sent directly to the sink increases the latency decreases. This is because the probability of the message going along the shortest



(a) 7, $P_{src} = 2.0$ (b) 7, $P_{src} = 1.0$ (c) 7, $P_{src} = 0.5$ (d) 7, $P_{src} = 0.25$

(e) 11, $P_{src} = 2.0$ (f) 11, $P_{src} = 1.0$ (g) 11, $P_{src} = 0.5$ (h) 11, $P_{src} = 0.25$

(i) 15, $P_{src} = 2.0$ (j) 15, $P_{src} = 1.0$ (k) 15, $P_{src} = 0.5$ (l) 15, $P_{src} = 0.25$

(m) 21, $P_{src} = 2.0$ (n) 21, $P_{src} = 1.0$ (o) 21, $P_{src} = 0.5$ (p) 21, $P_{src} = 0.25$

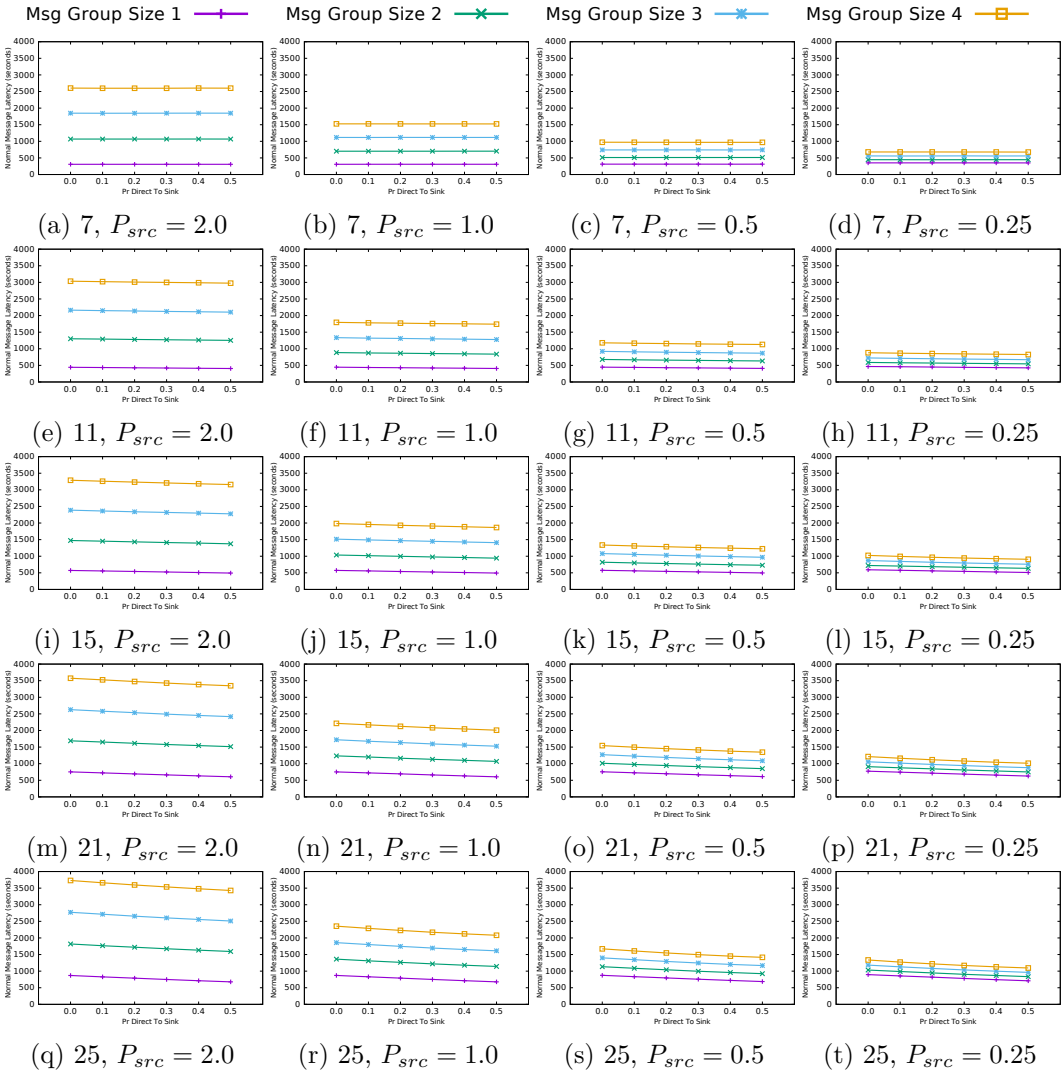
(q) 25, $P_{src} = 2.0$ (r) 25, $P_{src} = 1.0$ (s) 25, $P_{src} = 0.5$ (t) 25, $P_{src} = 0.25$

Figure D.3: *ILPRouting* results showing the messages sent per second for different probabilities of sending messages directly to the sink.

path is increasing, so messages take less time to reach the sink.

D.5 Attacker Distance

With a larger probability of sending messages directly to the sink the attacker distance becomes lower. This is to be expected as fewer messages are approaching the sink from behind, so there is a greater pull from the source along the shortest path.



(a) 7, $P_{src} = 2.0$ (b) 7, $P_{src} = 1.0$ (c) 7, $P_{src} = 0.5$ (d) 7, $P_{src} = 0.25$

(e) 11, $P_{src} = 2.0$ (f) 11, $P_{src} = 1.0$ (g) 11, $P_{src} = 0.5$ (h) 11, $P_{src} = 0.25$

(i) 15, $P_{src} = 2.0$ (j) 15, $P_{src} = 1.0$ (k) 15, $P_{src} = 0.5$ (l) 15, $P_{src} = 0.25$

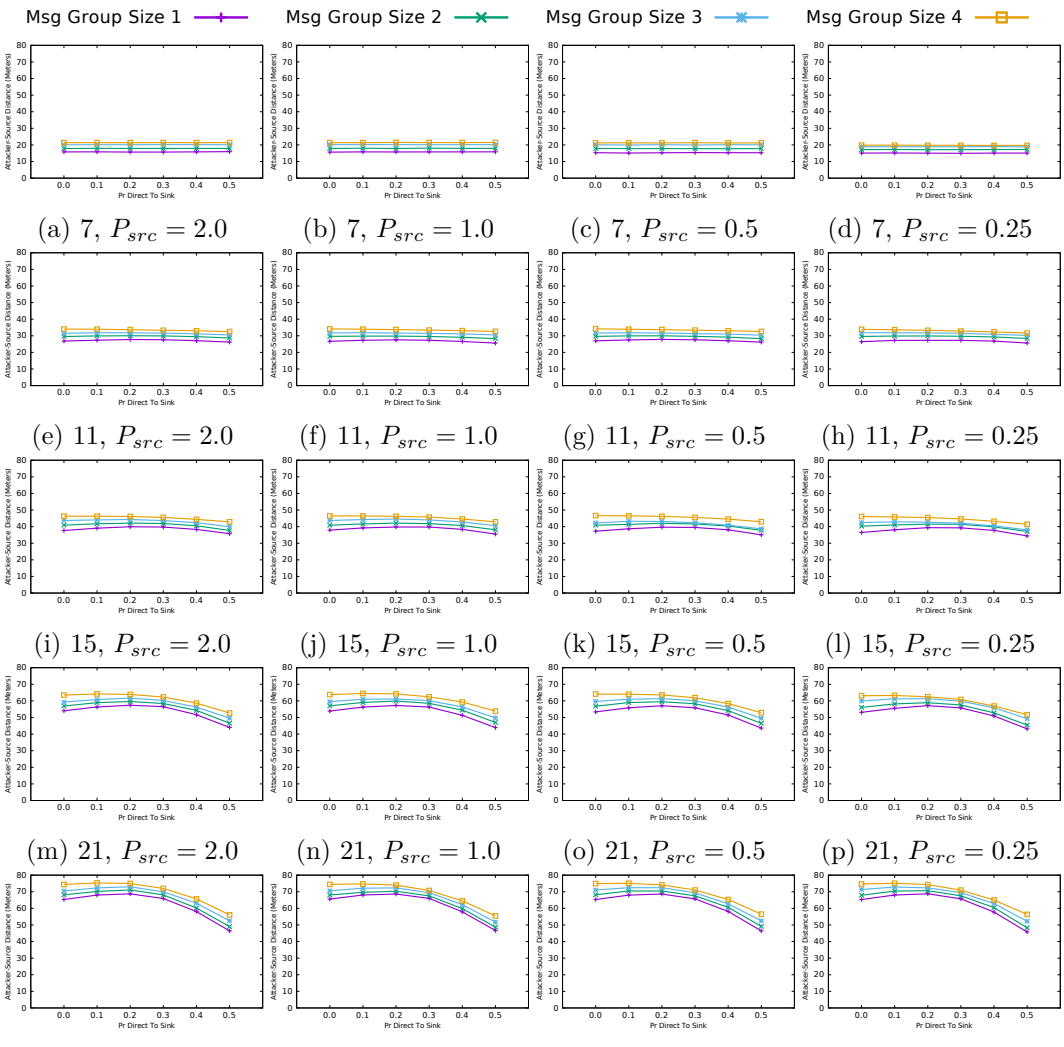
(m) 21, $P_{src} = 2.0$ (n) 21, $P_{src} = 1.0$ (o) 21, $P_{src} = 0.5$ (p) 21, $P_{src} = 0.25$

(q) 25, $P_{src} = 2.0$ (r) 25, $P_{src} = 1.0$ (s) 25, $P_{src} = 0.5$ (t) 25, $P_{src} = 0.25$

Figure D.4: *ILPRouting* results showing the message latency for different probabilities of sending messages directly to the sink.

D.6 Conclusions

Overall, sending messages in *ILPRouting* with a 20% or 30% probability of sending the message directly to the sink tends to produce the best results. The results showing 20% were focused on in Chapter 5 because of the low capture ratio and the high attacker distance it produced.



(a) 7, $P_{src} = 2.0$ (b) 7, $P_{src} = 1.0$ (c) 7, $P_{src} = 0.5$ (d) 7, $P_{src} = 0.25$

(e) 11, $P_{src} = 2.0$ (f) 11, $P_{src} = 1.0$ (g) 11, $P_{src} = 0.5$ (h) 11, $P_{src} = 0.25$

(i) 15, $P_{src} = 2.0$ (j) 15, $P_{src} = 1.0$ (k) 15, $P_{src} = 0.5$ (l) 15, $P_{src} = 0.25$

(m) 21, $P_{src} = 2.0$ (n) 21, $P_{src} = 1.0$ (o) 21, $P_{src} = 0.5$ (p) 21, $P_{src} = 0.25$

(q) 25, $P_{src} = 2.0$ (r) 25, $P_{src} = 1.0$ (s) 25, $P_{src} = 0.5$ (t) 25, $P_{src} = 0.25$

Figure D.5: *ILPRouting* results showing the attacker distance for different probabilities of sending messages directly to the sink.

Appendix E

Confidence Interval Tables

In this appendix the sample mean and 95% confidence intervals will be presented. Columns will be included in the tables for the parameters of the protocol which includes the source period P_{src}, also shown will be the number of repeats (R). The remaining six columns are key metrics of how the protocols perform and will include the mean result and the confidence interval. These six columns are: (i) the capture ratio, (ii) the time taken to capture ($\mathcal{TT}$), (iii) the receive ratio, (iv) the number of messages sent per second, (v) the delivery latency of ⟨normal⟩ messages, and (vi) the attacker distance from the source when the simulation terminated.

All results presented in this Thesis are present in these tables, except for *Static* which has been omitted due to their long length and the limited role the results for *Static* has in the results analysis. Rather than specify the full list of duty cycle or LPL parameters, numbers will be used which refer to the parameters specified in Table 7.1 and Table 7.2.

E.1 TOSSIM

E.1.1 Protectionless

Table E.1: Confidence Intervals for Protectionless on TOSSIM Network Size 7

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	100000	100.0 ± 0.0	2.0 ± 0.0	90 ± 0.1	106 ± 0	24 ± 0	0 ± 0
0.5	100000	100.0 ± 0.0	3.3 ± 0.0	96 ± 0.1	65 ± 0	24 ± 0	0 ± 0
1.0	100000	100.0 ± 0.0	5.9 ± 0.0	100 ± 0.0	36 ± 0	23 ± 0	0 ± 0
2.0	100000	100.0 ± 0.0	11.3 ± 0.0	100 ± 0.0	19 ± 0	23 ± 0	0 ± 0

Table E.2: Confidence Intervals for Protectionless on TOSSIM Network Size 11

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	100000	100.0 ± 0.0	3.0 ± 0.0	93 ± 0.1	339 ± 0	37 ± 0	0 ± 0
0.5	100000	100.0 ± 0.0	5.3 ± 0.0	98 ± 0.0	193 ± 0	37 ± 0	0 ± 0
1.0	100000	100.0 ± 0.0	9.9 ± 0.0	99 ± 0.0	103 ± 0	37 ± 0	0 ± 0
2.0	100000	100.0 ± 0.0	19.3 ± 0.0	99 ± 0.0	53 ± 0	37 ± 0	0 ± 0

Table E.3: Confidence Intervals for Protectionless on TOSSIM Network Size 15

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	100000	100.0 ± 0.0	4.1 ± 0.0	95 ± 0.0	705 ± 0	50 ± 0	0 ± 0
0.5	100000	100.0 ± 0.0	7.4 ± 0.0	98 ± 0.0	387 ± 0	50 ± 0	0 ± 0
1.0	100000	100.0 ± 0.0	14.0 ± 0.0	99 ± 0.0	203 ± 0	49 ± 0	0 ± 0
2.0	100000	100.0 ± 0.0	27.6 ± 0.1	99 ± 0.0	103 ± 0	49 ± 0	0 ± 0

Table E.4: Confidence Intervals for Protectionless on TOSSIM Network Size 21

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	100000	100.0 ± 0.0	5.6 ± 0.0	97 ± 0.0	1501 ± 0	68 ± 0	0 ± 0
0.5	100000	100.0 ± 0.0	10.6 ± 0.0	99 ± 0.0	802 ± 0	68 ± 0	0 ± 0
1.0	100000	100.0 ± 0.0	20.3 ± 0.0	99 ± 0.0	414 ± 0	68 ± 0	0 ± 0
2.0	100000	100.0 ± 0.0	40.2 ± 0.1	99 ± 0.0	210 ± 0	68 ± 0	0 ± 0

Table E.5: Confidence Intervals for Protectionless on TOSSIM Network Size 25

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	100000	100.0 ± 0.0	6.7 ± 0.0	97 ± 0.0	2196 ± 1	81 ± 0	0 ± 0
0.5	100000	100.0 ± 0.0	12.7 ± 0.0	99 ± 0.0	1160 ± 0	80 ± 0	0 ± 0
1.0	100000	100.0 ± 0.0	24.6 ± 0.1	99 ± 0.0	595 ± 0	80 ± 0	0 ± 0
2.0	100000	100.0 ± 0.0	48.8 ± 0.2	99 ± 0.0	301 ± 0	80 ± 0	0 ± 0

E.1.2 Dynamic

Table E.6: Confidence Intervals for Dynamic on TOSSIM Network Size 7

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	AEst	20377	5.3 ± 0.3	7.1 ± 0.0	71 ± 0.2	427 ± 2	29 ± 0	22 ± 0
0.25	Fxd1	2000	13.2 ± 1.5	7.0 ± 0.0	91 ± 0.5	219 ± 2	26 ± 0	20 ± 0
0.25	Fxd2	2000	8.8 ± 1.2	7.1 ± 0.0	88 ± 0.5	273 ± 3	27 ± 0	21 ± 0

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	RND	2000	6.9 ± 1.1	7.1 ± 0.0	84 ± 0.6	295 ± 2	28 ± 0	22 ± 0
0.25	SINK	20479	7.5 ± 0.4	7.1 ± 0.0	87 ± 0.2	288 ± 1	27 ± 0	23 ± 0
0.5	AEst	20455	5.3 ± 0.3	9.4 ± 0.0	82 ± 0.2	377 ± 2	28 ± 0	22 ± 0
0.5	Fxd1	2000	11.2 ± 1.4	9.2 ± 0.0	95 ± 0.4	163 ± 1	26 ± 0	20 ± 0
0.5	Fxd2	2000	7.7 ± 1.2	9.3 ± 0.0	94 ± 0.4	212 ± 2	26 ± 0	21 ± 0
0.5	RND	2000	6.8 ± 1.1	9.4 ± 0.0	95 ± 0.3	235 ± 2	26 ± 0	22 ± 0
0.5	SINK	20473	5.7 ± 0.3	9.4 ± 0.0	93 ± 0.1	235 ± 1	26 ± 0	23 ± 0
1.0	AEst	20446	5.4 ± 0.3	13.9 ± 0.0	93 ± 0.1	290 ± 1	26 ± 0	22 ± 0
1.0	Fxd1	2000	11.4 ± 1.4	13.6 ± 0.1	99 ± 0.2	109 ± 1	25 ± 0	20 ± 0
1.0	Fxd2	2000	7.4 ± 1.2	13.8 ± 0.1	98 ± 0.3	143 ± 1	25 ± 0	22 ± 0
1.0	RND	2000	5.7 ± 1.0	14.0 ± 0.1	98 ± 0.2	160 ± 1	25 ± 0	22 ± 0
1.0	SINK	20485	5.5 ± 0.3	13.9 ± 0.0	97 ± 0.1	166 ± 1	25 ± 0	24 ± 0
2.0	AEst	20439	5.9 ± 0.3	23.4 ± 0.0	97 ± 0.1	193 ± 1	26 ± 0	21 ± 0
2.0	Fxd1	2000	12.2 ± 1.4	22.8 ± 0.2	99 ± 0.1	66 ± 1	25 ± 0	20 ± 1
2.0	Fxd2	2000	7.6 ± 1.2	23.2 ± 0.1	99 ± 0.2	86 ± 1	25 ± 0	22 ± 0
2.0	RND	2000	7.0 ± 1.1	23.3 ± 0.1	99 ± 0.2	96 ± 1	25 ± 0	22 ± 0
2.0	SINK	20477	5.9 ± 0.3	23.4 ± 0.0	98 ± 0.1	101 ± 0	25 ± 0	24 ± 0

Table E.7: Confidence Intervals for Dynamic on TOSSIM Network Size 11

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	AEst	20295	1.0 ± 0.1	9.2 ± 0.0	51 ± 0.2	1832 ± 5	50 ± 0	42 ± 0
0.25	Fxd1	2002	1.8 ± 0.6	9.2 ± 0.0	89 ± 0.4	779 ± 4	43 ± 0	40 ± 1
0.25	Fxd2	2002	1.6 ± 0.6	9.2 ± 0.0	80 ± 0.5	965 ± 6	45 ± 0	40 ± 1
0.25	RND	2000	1.5 ± 0.5	9.2 ± 0.0	79 ± 0.5	1031 ± 7	46 ± 0	39 ± 1
0.25	SINK	20492	0.7 ± 0.1	9.2 ± 0.0	72 ± 0.2	1253 ± 4	47 ± 0	44 ± 0
0.5	AEst	19587	0.8 ± 0.1	13.5 ± 0.0	65 ± 0.2	1695 ± 6	48 ± 0	39 ± 0
0.5	Fxd1	2001	1.8 ± 0.6	13.5 ± 0.0	96 ± 0.2	547 ± 4	41 ± 0	39 ± 1
0.5	Fxd2	2002	1.1 ± 0.5	13.5 ± 0.0	92 ± 0.4	707 ± 5	42 ± 0	39 ± 1
0.5	RND	2002	1.1 ± 0.5	13.5 ± 0.0	89 ± 0.3	781 ± 6	42 ± 0	40 ± 1
0.5	SINK	20496	0.6 ± 0.1	13.5 ± 0.0	83 ± 0.1	1023 ± 4	44 ± 0	43 ± 0
1.0	AEst	19265	0.9 ± 0.1	22.2 ± 0.0	79 ± 0.2	1309 ± 5	45 ± 0	37 ± 0
1.0	Fxd1	2002	1.1 ± 0.5	22.2 ± 0.0	98 ± 0.2	338 ± 2	40 ± 0	38 ± 1
1.0	Fxd2	2002	1.4 ± 0.5	22.2 ± 0.0	97 ± 0.2	446 ± 4	40 ± 0	38 ± 1
1.0	RND	2002	1.3 ± 0.5	22.2 ± 0.0	96 ± 0.2	505 ± 4	40 ± 0	37 ± 1
1.0	SINK	20497	0.7 ± 0.1	22.2 ± 0.0	90 ± 0.1	728 ± 3	42 ± 0	41 ± 0
2.0	AEst	15454	1.2 ± 0.2	40.0 ± 0.0	88 ± 0.1	880 ± 4	42 ± 0	36 ± 0
2.0	Fxd1	2002	1.2 ± 0.5	40.0 ± 0.1	99 ± 0.1	190 ± 1	39 ± 0	38 ± 1
2.0	Fxd2	2002	1.6 ± 0.5	40.0 ± 0.1	98 ± 0.2	253 ± 2	39 ± 0	38 ± 1
2.0	RND	2001	1.6 ± 0.6	40.0 ± 0.1	98 ± 0.2	285 ± 2	39 ± 0	37 ± 1
2.0	SINK	20501	0.7 ± 0.1	40.0 ± 0.0	96 ± 0.1	451 ± 2	40 ± 0	41 ± 0

Table E.8: Confidence Intervals for Dynamic on TOSSIM Network Size 15

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	AEst	17723	0.6 ± 0.1	11.3 ± 0.0	44 ± 0.2	4252 ± 12	70 ± 0	64 ± 0
0.25	Fxd1	2002	0.5 ± 0.3	11.3 ± 0.0	87 ± 0.4	1671 ± 9	59 ± 0	63 ± 1
0.25	Fxd2	2002	0.5 ± 0.3	11.3 ± 0.0	81 ± 0.4	2075 ± 12	62 ± 0	60 ± 1
0.25	RND	2001	0.7 ± 0.4	11.3 ± 0.0	80 ± 0.4	2207 ± 13	63 ± 0	59 ± 1
0.25	SINK	20500	0.2 ± 0.1	11.3 ± 0.0	63 ± 0.2	3057 ± 8	65 ± 0	65 ± 0
0.5	AEst	18802	0.4 ± 0.1	17.7 ± 0.0	58 ± 0.2	3984 ± 13	66 ± 0	58 ± 0
0.5	Fxd1	2001	0.7 ± 0.4	17.7 ± 0.0	95 ± 0.2	1134 ± 7	55 ± 0	59 ± 1
0.5	Fxd2	2001	0.2 ± 0.2	17.7 ± 0.0	89 ± 0.3	1449 ± 10	57 ± 0	59 ± 1
0.5	RND	2002	0.3 ± 0.3	17.7 ± 0.0	87 ± 0.3	1593 ± 11	58 ± 0	58 ± 1
0.5	SINK	20499	0.3 ± 0.1	17.7 ± 0.0	77 ± 0.1	2503 ± 8	62 ± 0	59 ± 0
1.0	AEst	18825	0.4 ± 0.1	30.5 ± 0.0	72 ± 0.2	3031 ± 12	63 ± 0	55 ± 0
1.0	Fxd1	2002	0.3 ± 0.2	30.5 ± 0.0	97 ± 0.2	667 ± 4	54 ± 0	60 ± 1
1.0	Fxd2	2002	0.3 ± 0.3	30.5 ± 0.0	96 ± 0.2	882 ± 7	54 ± 0	59 ± 1
1.0	RND	2002	0.5 ± 0.3	30.5 ± 0.0	96 ± 0.2	994 ± 8	55 ± 0	58 ± 1
1.0	SINK	20499	0.3 ± 0.1	30.5 ± 0.0	86 ± 0.1	1769 ± 6	59 ± 0	61 ± 0
2.0	AEst	16174	0.4 ± 0.1	56.6 ± 0.0	84 ± 0.1	2034 ± 9	59 ± 0	54 ± 0
2.0	Fxd1	2001	0.2 ± 0.2	56.7 ± 0.0	98 ± 0.2	365 ± 2	53 ± 0	60 ± 1
2.0	Fxd2	2001	0.7 ± 0.4	56.6 ± 0.1	98 ± 0.2	483 ± 4	54 ± 0	59 ± 1
2.0	RND	2002	0.5 ± 0.3	56.6 ± 0.1	98 ± 0.2	553 ± 4	53 ± 0	58 ± 1
2.0	SINK	20498	0.2 ± 0.1	56.6 ± 0.0	93 ± 0.1	1126 ± 4	56 ± 0	61 ± 0

Table E.9: Confidence Intervals for Dynamic on TOSSIM Network Size 21

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	AEst	4961	0.2 ± 0.1	14.4 ± 0.0	39 ± 0.4	9928 ± 49	100 ± 0	97 ± 1
0.25	Fxd1	2002	0.3 ± 0.2	14.4 ± 0.0	84 ± 0.3	3705 ± 18	83 ± 0	96 ± 1
0.25	Fxd2	2002	0.3 ± 0.3	14.4 ± 0.0	82 ± 0.3	4528 ± 25	86 ± 0	90 ± 1
0.25	RND	2002	0.1 ± 0.2	14.4 ± 0.0	79 ± 0.3	4821 ± 25	87 ± 0	88 ± 1
0.25	SINK	3603	0.3 ± 0.2	14.4 ± 0.0	54 ± 0.4	7545 ± 45	92 ± 0	99 ± 1
0.5	AEst	3261	0.2 ± 0.1	24.0 ± 0.0	51 ± 0.4	9365 ± 66	91 ± 0	93 ± 1
0.5	Fxd1	2001	0.2 ± 0.2	24.0 ± 0.0	93 ± 0.2	2399 ± 13	76 ± 0	93 ± 1
0.5	Fxd2	2002	0.1 ± 0.1	24.0 ± 0.0	88 ± 0.3	3055 ± 19	80 ± 0	92 ± 1
0.5	RND	2002	0.2 ± 0.2	24.0 ± 0.0	88 ± 0.3	3417 ± 21	81 ± 0	90 ± 1
0.5	SINK	5782	0.3 ± 0.1	24.0 ± 0.0	70 ± 0.2	6081 ± 33	86 ± 0	90 ± 1
1.0	AEst	2971	0.2 ± 0.1	43.1 ± 0.0	66 ± 0.4	7108 ± 63	87 ± 0	85 ± 1
1.0	Fxd1	2002	0.3 ± 0.2	43.1 ± 0.0	96 ± 0.2	1386 ± 8	74 ± 0	91 ± 1
1.0	Fxd2	2002	0.1 ± 0.2	43.1 ± 0.0	95 ± 0.2	1817 ± 13	75 ± 0	91 ± 1
1.0	RND	2002	0.2 ± 0.2	43.1 ± 0.0	94 ± 0.2	2057 ± 14	75 ± 0	89 ± 1
1.0	SINK	4960	0.2 ± 0.1	43.1 ± 0.0	81 ± 0.2	4340 ± 28	83 ± 0	87 ± 1
2.0	AEst	2523	0.2 ± 0.2	81.7 ± 0.1	79 ± 0.3	4912 ± 45	83 ± 0	81 ± 1
2.0	Fxd1	2002	0.1 ± 0.1	81.8 ± 0.0	99 ± 0.1	743 ± 5	73 ± 0	92 ± 1
2.0	Fxd2	2002	0.2 ± 0.2	81.8 ± 0.1	97 ± 0.2	980 ± 8	74 ± 0	90 ± 1

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	RND	2002	0.1 ± 0.2	81.8 ± 0.1	97 ± 0.2	1127 ± 9	74 ± 0	89 ± 1
2.0	SINK	4388	0.1 ± 0.1	81.8 ± 0.0	88 ± 0.2	2857 ± 19	79 ± 0	89 ± 1

Table E.10: Confidence Intervals for Dynamic on TOSSIM Network Size 25

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	AEst	1298	0.1 ± 0.2	16.5 ± 0.0	38 ± 0.8	14989 ± 139	119 ± 1	116 ± 1
0.25	Fxd1	2001	0.0 ± 0.1	16.5 ± 0.0	84 ± 0.3	5574 ± 21	98 ± 0	118 ± 1
0.25	Fxd2	2002	0.1 ± 0.1	16.5 ± 0.0	81 ± 0.3	6734 ± 34	102 ± 0	111 ± 1
0.25	RND	2002	0.1 ± 0.2	16.5 ± 0.0	78 ± 0.3	7181 ± 32	103 ± 0	106 ± 1
0.25	SINK	1995	0.3 ± 0.2	16.5 ± 0.0	50 ± 0.6	11912 ± 97	111 ± 1	121 ± 1
0.5	AEst	1998	0.1 ± 0.1	28.3 ± 0.0	48 ± 0.6	14085 ± 121	109 ± 1	115 ± 1
0.5	Fxd1	2002	0.1 ± 0.2	28.2 ± 0.0	92 ± 0.2	3497 ± 18	90 ± 0	116 ± 1
0.5	Fxd2	2001	0.1 ± 0.1	28.2 ± 0.0	88 ± 0.3	4447 ± 26	95 ± 0	114 ± 1
0.5	RND	2000	0.1 ± 0.1	28.2 ± 0.0	88 ± 0.3	4968 ± 28	95 ± 0	110 ± 1
0.5	SINK	2000	0.1 ± 0.1	28.3 ± 0.0	66 ± 0.4	9401 ± 76	102 ± 0	114 ± 1
1.0	AEst	1501	0.1 ± 0.2	51.6 ± 0.1	63 ± 0.5	10863 ± 116	102 ± 0	109 ± 2
1.0	Fxd1	2002	0.1 ± 0.2	51.6 ± 0.0	97 ± 0.2	2011 ± 12	87 ± 0	114 ± 1
1.0	Fxd2	2002	0.1 ± 0.2	51.6 ± 0.0	94 ± 0.2	2613 ± 21	89 ± 0	113 ± 1
1.0	RND	2001	0.1 ± 0.1	51.6 ± 0.0	93 ± 0.2	2984 ± 20	90 ± 0	112 ± 1
1.0	SINK	2000	0.1 ± 0.1	51.6 ± 0.0	79 ± 0.3	6749 ± 57	98 ± 0	104 ± 1
2.0	AEst	585	0.0 ± 0.0	99.1 ± 0.0	76 ± 0.6	7743 ± 120	99 ± 0	100 ± 3
2.0	Fxd1	2001	0.2 ± 0.2	99.1 ± 0.1	98 ± 0.1	1081 ± 9	86 ± 0	112 ± 1
2.0	Fxd2	2002	0.2 ± 0.2	99.1 ± 0.1	98 ± 0.1	1409 ± 11	87 ± 0	112 ± 1
2.0	RND	2002	0.1 ± 0.1	99.1 ± 0.1	97 ± 0.1	1646 ± 12	87 ± 0	110 ± 1
2.0	SINK	1373	0.1 ± 0.2	99.1 ± 0.1	87 ± 0.3	4463 ± 52	94 ± 0	107 ± 2

E.1.3 DynamicSPR

Table E.11: Confidence Intervals for DynamicSPR on TOSSIM Network Size 7

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	20492	9.2 ± 0.4	7.1 ± 0.0	98 ± 0.1	213 ± 0	24 ± 0	20 ± 0
0.25	Fxd2	20476	3.0 ± 0.2	7.2 ± 0.0	84 ± 0.2	274 ± 1	27 ± 0	25 ± 0
0.25	RND	20484	3.3 ± 0.2	7.2 ± 0.0	87 ± 0.2	267 ± 1	26 ± 0	25 ± 0
0.5	Fxd1	20497	7.5 ± 0.4	9.4 ± 0.0	98 ± 0.1	155 ± 0	24 ± 0	21 ± 0
0.5	Fxd2	20473	2.8 ± 0.2	9.5 ± 0.0	98 ± 0.1	205 ± 0	24 ± 0	25 ± 0
0.5	RND	20485	3.2 ± 0.2	9.5 ± 0.0	97 ± 0.1	202 ± 0	25 ± 0	25 ± 0
1.0	Fxd1	20491	7.4 ± 0.4	13.9 ± 0.0	99 ± 0.0	101 ± 0	24 ± 0	21 ± 0
1.0	Fxd2	20482	2.9 ± 0.2	14.1 ± 0.0	99 ± 0.0	134 ± 0	24 ± 0	26 ± 0
1.0	RND	20486	3.0 ± 0.2	14.1 ± 0.0	99 ± 0.0	133 ± 0	24 ± 0	25 ± 0
2.0	Fxd1	20495	6.8 ± 0.3	23.5 ± 0.0	99 ± 0.0	61 ± 0	24 ± 0	21 ± 0

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	Fxd2	20474	3.1 ± 0.2	23.7 ± 0.0	99 ± 0.0	80 ± 0	24 ± 0	25 ± 0
2.0	RND	20483	3.1 ± 0.2	23.7 ± 0.0	99 ± 0.0	80 ± 0	24 ± 0	25 ± 0

Table E.12: Confidence Intervals for DynamicSPR on TOSSIM Network Size 11

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	20497	2.3 ± 0.2	9.2 ± 0.0	94 ± 0.1	652 ± 1	39 ± 0	38 ± 0
0.25	Fxd2	20500	0.5 ± 0.1	9.2 ± 0.0	85 ± 0.1	833 ± 2	43 ± 0	45 ± 0
0.25	RND	20500	0.6 ± 0.1	9.2 ± 0.0	88 ± 0.1	807 ± 2	42 ± 0	45 ± 0
0.5	Fxd1	20500	2.0 ± 0.2	13.5 ± 0.0	98 ± 0.1	438 ± 0	38 ± 0	38 ± 0
0.5	Fxd2	20498	0.6 ± 0.1	13.5 ± 0.0	95 ± 0.1	547 ± 1	39 ± 0	43 ± 0
0.5	RND	20499	0.7 ± 0.1	13.5 ± 0.0	94 ± 0.1	542 ± 1	39 ± 0	43 ± 0
1.0	Fxd1	20497	1.8 ± 0.2	22.2 ± 0.0	98 ± 0.0	265 ± 0	38 ± 0	36 ± 0
1.0	Fxd2	20502	0.6 ± 0.1	22.2 ± 0.0	98 ± 0.0	335 ± 1	38 ± 0	42 ± 0
1.0	RND	20500	0.8 ± 0.1	22.2 ± 0.0	98 ± 0.0	332 ± 1	38 ± 0	41 ± 0
2.0	Fxd1	20498	1.7 ± 0.2	40.0 ± 0.0	99 ± 0.0	148 ± 0	38 ± 0	37 ± 0
2.0	Fxd2	20499	0.6 ± 0.1	40.1 ± 0.0	99 ± 0.0	186 ± 0	38 ± 0	42 ± 0
2.0	RND	20499	0.8 ± 0.1	40.1 ± 0.0	99 ± 0.0	185 ± 0	38 ± 0	42 ± 0

Table E.13: Confidence Intervals for DynamicSPR on TOSSIM Network Size 15

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	20496	0.7 ± 0.1	11.3 ± 0.0	91 ± 0.1	1330 ± 1	54 ± 0	56 ± 0
0.25	Fxd2	20500	0.2 ± 0.1	11.3 ± 0.0	87 ± 0.1	1715 ± 3	58 ± 0	64 ± 0
0.25	RND	20497	0.3 ± 0.1	11.3 ± 0.0	89 ± 0.1	1631 ± 3	57 ± 0	63 ± 0
0.5	Fxd1	20497	0.9 ± 0.1	17.7 ± 0.0	98 ± 0.0	860 ± 1	51 ± 0	51 ± 0
0.5	Fxd2	20497	0.4 ± 0.1	17.7 ± 0.0	93 ± 0.1	1038 ± 2	53 ± 0	58 ± 0
0.5	RND	20499	0.3 ± 0.1	17.7 ± 0.0	93 ± 0.1	1035 ± 2	53 ± 0	58 ± 0
1.0	Fxd1	20497	0.7 ± 0.1	30.5 ± 0.0	98 ± 0.0	492 ± 1	51 ± 0	53 ± 0
1.0	Fxd2	20499	0.3 ± 0.1	30.5 ± 0.0	98 ± 0.0	599 ± 1	51 ± 0	59 ± 0
1.0	RND	20499	0.3 ± 0.1	30.5 ± 0.0	98 ± 0.0	592 ± 1	51 ± 0	58 ± 0
2.0	Fxd1	20499	0.7 ± 0.1	56.6 ± 0.0	99 ± 0.0	266 ± 0	51 ± 0	53 ± 0
2.0	Fxd2	20494	0.2 ± 0.1	56.7 ± 0.0	99 ± 0.0	324 ± 1	51 ± 0	59 ± 0
2.0	RND	20500	0.4 ± 0.1	56.6 ± 0.0	99 ± 0.0	321 ± 1	51 ± 0	59 ± 0

Table E.14: Confidence Intervals for DynamicSPR on TOSSIM Network Size 21

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	20498	0.1 ± 0.0	14.4 ± 0.0	89 ± 0.1	2847 ± 3	76 ± 0	80 ± 0
0.25	Fxd2	18346	0.1 ± 0.0	14.4 ± 0.0	88 ± 0.1	3579 ± 8	79 ± 0	88 ± 0

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	RND	20498	0.1 ± 0.0	14.4 ± 0.0	88 ± 0.1	3391 ± 7	78 ± 0	87 ± 0
0.5	Fxd1	20497	0.3 ± 0.1	24.0 ± 0.0	97 ± 0.0	1728 ± 2	71 ± 0	72 ± 0
0.5	Fxd2	20497	0.1 ± 0.0	24.0 ± 0.0	93 ± 0.1	2052 ± 4	73 ± 0	81 ± 0
0.5	RND	20498	0.1 ± 0.0	24.0 ± 0.0	95 ± 0.1	2028 ± 4	73 ± 0	81 ± 0
1.0	Fxd1	20500	0.2 ± 0.1	43.1 ± 0.0	98 ± 0.0	958 ± 1	70 ± 0	73 ± 0
1.0	Fxd2	20497	0.1 ± 0.0	43.1 ± 0.0	98 ± 0.0	1121 ± 2	70 ± 0	79 ± 0
1.0	RND	20499	0.1 ± 0.0	43.1 ± 0.0	98 ± 0.0	1103 ± 2	71 ± 0	79 ± 0
2.0	Fxd1	20497	0.3 ± 0.1	81.8 ± 0.0	99 ± 0.0	506 ± 0	70 ± 0	73 ± 0
2.0	Fxd2	9632	0.1 ± 0.1	81.8 ± 0.0	99 ± 0.0	594 ± 2	70 ± 0	79 ± 0
2.0	RND	20502	0.1 ± 0.0	81.8 ± 0.0	99 ± 0.0	584 ± 1	70 ± 0	78 ± 0

Table E.15: Confidence Intervals for DynamicSPR on TOSSIM Network Size 25

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	10271	0.1 ± 0.0	16.5 ± 0.0	89 ± 0.1	4217 ± 6	90 ± 0	94 ± 0
0.25	Fxd2	4914	0.1 ± 0.1	16.5 ± 0.0	88 ± 0.2	5190 ± 24	93 ± 0	102 ± 1
0.25	RND	5081	0.0 ± 0.0	16.5 ± 0.0	88 ± 0.2	4945 ± 19	92 ± 0	101 ± 1
0.5	Fxd1	5462	0.1 ± 0.1	28.3 ± 0.0	97 ± 0.1	2458 ± 4	84 ± 0	87 ± 1
0.5	Fxd2	5036	0.0 ± 0.0	28.3 ± 0.0	93 ± 0.1	2914 ± 12	87 ± 0	97 ± 1
0.5	RND	5069	0.1 ± 0.1	28.3 ± 0.0	95 ± 0.1	2850 ± 11	86 ± 0	95 ± 1
1.0	Fxd1	11076	0.1 ± 0.1	51.6 ± 0.0	98 ± 0.1	1350 ± 2	83 ± 0	85 ± 0
1.0	Fxd2	5026	0.1 ± 0.1	51.6 ± 0.0	98 ± 0.1	1546 ± 6	83 ± 0	90 ± 1
1.0	RND	5296	0.1 ± 0.1	51.6 ± 0.0	97 ± 0.1	1527 ± 6	83 ± 0	90 ± 1
2.0	Fxd1	11399	0.2 ± 0.1	99.1 ± 0.0	99 ± 0.0	708 ± 1	82 ± 0	85 ± 0
2.0	Fxd2	5285	0.0 ± 0.1	99.1 ± 0.0	99 ± 0.1	813 ± 3	82 ± 0	91 ± 1
2.0	RND	5289	0.1 ± 0.1	99.1 ± 0.0	99 ± 0.1	798 ± 3	82 ± 0	90 ± 1

E.1.4 ILPRouting

Table E.16: Confidence Intervals for ILPRouting on TOSSIM Network Size 7

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	0	1	16903	19.9 ± 0.6	7.1 ± 0.0	88 ± 0.1	140 ± 0	351 ± 1	15 ± 0
0.25	0	2	16626	13.9 ± 0.5	7.2 ± 0.0	87 ± 0.1	136 ± 0	447 ± 1	17 ± 0
0.25	0	3	17646	7.7 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	560 ± 1	19 ± 0
0.25	0	4	17937	6.0 ± 0.3	7.4 ± 0.0	73 ± 0.1	122 ± 0	680 ± 1	20 ± 0
0.25	10	1	21493	19.5 ± 0.5	7.1 ± 0.0	88 ± 0.1	140 ± 0	351 ± 1	15 ± 0
0.25	10	2	16535	13.4 ± 0.5	7.2 ± 0.0	87 ± 0.1	137 ± 0	447 ± 1	17 ± 0
0.25	10	3	15311	7.6 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	558 ± 1	19 ± 0
0.25	10	4	15860	6.0 ± 0.4	7.4 ± 0.0	74 ± 0.1	122 ± 0	680 ± 1	20 ± 0
0.25	20	1	21491	20.2 ± 0.5	7.1 ± 0.0	88 ± 0.1	139 ± 0	352 ± 1	15 ± 0
0.25	20	2	16469	14.2 ± 0.5	7.2 ± 0.0	87 ± 0.1	136 ± 0	446 ± 1	17 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	20	3	21492	7.9 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	559 ± 1	19 ± 0
0.25	20	4	21490	6.1 ± 0.3	7.4 ± 0.0	74 ± 0.1	122 ± 0	679 ± 1	20 ± 0
0.25	30	1	18567	20.1 ± 0.6	7.1 ± 0.0	88 ± 0.1	139 ± 0	351 ± 1	15 ± 0
0.25	30	2	16426	13.9 ± 0.5	7.2 ± 0.0	87 ± 0.1	136 ± 0	446 ± 1	17 ± 0
0.25	30	3	15333	7.6 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	561 ± 1	19 ± 0
0.25	30	4	16063	6.2 ± 0.4	7.4 ± 0.0	74 ± 0.1	122 ± 0	678 ± 1	20 ± 0
0.25	40	1	17918	19.8 ± 0.6	7.1 ± 0.0	88 ± 0.1	140 ± 0	351 ± 1	15 ± 0
0.25	40	2	16541	13.8 ± 0.5	7.2 ± 0.0	87 ± 0.1	137 ± 0	446 ± 1	17 ± 0
0.25	40	3	15323	7.5 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	558 ± 1	19 ± 0
0.25	40	4	15982	6.3 ± 0.4	7.4 ± 0.0	74 ± 0.1	122 ± 0	678 ± 1	20 ± 0
0.25	50	1	16060	19.6 ± 0.6	7.1 ± 0.0	88 ± 0.1	140 ± 0	350 ± 1	15 ± 0
0.25	50	2	15634	13.6 ± 0.5	7.2 ± 0.0	87 ± 0.1	136 ± 0	447 ± 1	17 ± 0
0.25	50	3	16386	8.0 ± 0.4	7.3 ± 0.0	79 ± 0.1	130 ± 0	560 ± 1	19 ± 0
0.25	50	4	17432	6.0 ± 0.4	7.4 ± 0.0	74 ± 0.1	122 ± 0	677 ± 1	20 ± 0
0.5	0	1	16416	18.8 ± 0.6	9.3 ± 0.0	95 ± 0.1	105 ± 0	312 ± 1	15 ± 0
0.5	0	2	14108	8.7 ± 0.5	9.5 ± 0.0	94 ± 0.1	101 ± 0	513 ± 1	18 ± 0
0.5	0	3	14224	4.0 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	741 ± 1	20 ± 0
0.5	0	4	19739	2.0 ± 0.2	9.9 ± 0.0	76 ± 0.1	89 ± 0	972 ± 1	21 ± 0
0.5	10	1	21497	19.7 ± 0.5	9.3 ± 0.0	95 ± 0.1	105 ± 0	313 ± 1	15 ± 0
0.5	10	2	18720	8.8 ± 0.4	9.5 ± 0.0	94 ± 0.1	102 ± 0	511 ± 1	18 ± 0
0.5	10	3	13788	4.3 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	741 ± 1	20 ± 0
0.5	10	4	13788	2.1 ± 0.2	9.9 ± 0.0	76 ± 0.1	89 ± 0	972 ± 2	21 ± 0
0.5	20	1	21496	19.2 ± 0.5	9.3 ± 0.0	95 ± 0.1	105 ± 0	312 ± 1	15 ± 0
0.5	20	2	17358	9.0 ± 0.4	9.5 ± 0.0	94 ± 0.1	102 ± 0	513 ± 1	18 ± 0
0.5	20	3	14633	3.7 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	741 ± 1	20 ± 0
0.5	20	4	19175	2.1 ± 0.2	9.9 ± 0.0	77 ± 0.1	89 ± 0	971 ± 1	21 ± 0
0.5	30	1	14310	19.0 ± 0.6	9.3 ± 0.0	95 ± 0.1	105 ± 0	313 ± 1	15 ± 0
0.5	30	2	15649	9.1 ± 0.5	9.5 ± 0.0	94 ± 0.1	101 ± 0	512 ± 1	18 ± 0
0.5	30	3	13994	4.2 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	742 ± 1	20 ± 0
0.5	30	4	14786	1.8 ± 0.2	9.9 ± 0.0	76 ± 0.1	89 ± 0	969 ± 2	21 ± 0
0.5	40	1	17604	18.9 ± 0.6	9.3 ± 0.0	95 ± 0.1	105 ± 0	312 ± 1	15 ± 0
0.5	40	2	15101	9.4 ± 0.5	9.5 ± 0.0	94 ± 0.1	101 ± 0	512 ± 1	18 ± 0
0.5	40	3	19817	3.9 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	740 ± 1	20 ± 0
0.5	40	4	20233	2.1 ± 0.2	9.9 ± 0.0	77 ± 0.1	89 ± 0	970 ± 1	21 ± 0
0.5	50	1	21991	19.0 ± 0.5	9.3 ± 0.0	95 ± 0.1	105 ± 0	312 ± 1	15 ± 0
0.5	50	2	21992	9.0 ± 0.4	9.5 ± 0.0	94 ± 0.1	102 ± 0	513 ± 1	18 ± 0
0.5	50	3	21222	4.0 ± 0.3	9.7 ± 0.0	87 ± 0.1	98 ± 0	742 ± 1	20 ± 0
0.5	50	4	21997	1.9 ± 0.2	9.9 ± 0.0	77 ± 0.1	89 ± 0	969 ± 1	21 ± 0
1.0	0	1	13813	21.1 ± 0.7	13.8 ± 0.0	97 ± 0.1	69 ± 0	308 ± 1	16 ± 0
1.0	0	2	21495	8.9 ± 0.4	14.3 ± 0.0	91 ± 0.1	66 ± 0	702 ± 1	18 ± 0
1.0	0	3	13778	3.7 ± 0.3	14.6 ± 0.0	96 ± 0.1	65 ± 0	1117 ± 2	20 ± 0
1.0	0	4	13966	1.6 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1526 ± 2	21 ± 0
1.0	10	1	16562	20.8 ± 0.6	13.8 ± 0.0	97 ± 0.1	69 ± 0	309 ± 1	16 ± 0
1.0	10	2	15575	8.2 ± 0.4	14.3 ± 0.0	91 ± 0.1	66 ± 0	702 ± 1	18 ± 0
1.0	10	3	21351	3.6 ± 0.2	14.6 ± 0.0	96 ± 0.1	65 ± 0	1116 ± 1	20 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	10	4	15153	1.5 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1526 ± 2	21 ± 0
1.0	20	1	21496	20.7 ± 0.5	13.8 ± 0.0	97 ± 0.1	69 ± 0	309 ± 1	16 ± 0
1.0	20	2	17929	8.7 ± 0.4	14.3 ± 0.0	91 ± 0.1	66 ± 0	703 ± 1	18 ± 0
1.0	20	3	20941	3.7 ± 0.3	14.6 ± 0.0	96 ± 0.1	65 ± 0	1118 ± 1	20 ± 0
1.0	20	4	18864	1.5 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1525 ± 2	21 ± 0
1.0	30	1	15537	21.1 ± 0.6	13.8 ± 0.0	97 ± 0.1	69 ± 0	309 ± 1	16 ± 0
1.0	30	2	14210	8.3 ± 0.5	14.3 ± 0.0	91 ± 0.2	66 ± 0	703 ± 1	18 ± 0
1.0	30	3	15189	3.6 ± 0.3	14.6 ± 0.0	96 ± 0.1	65 ± 0	1117 ± 1	20 ± 0
1.0	30	4	21493	1.6 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1525 ± 2	21 ± 0
1.0	40	1	14766	20.4 ± 0.7	13.8 ± 0.0	97 ± 0.1	69 ± 0	309 ± 1	16 ± 0
1.0	40	2	21497	8.7 ± 0.4	14.3 ± 0.0	91 ± 0.1	66 ± 0	703 ± 1	18 ± 0
1.0	40	3	21495	3.8 ± 0.3	14.6 ± 0.0	96 ± 0.1	65 ± 0	1117 ± 1	20 ± 0
1.0	40	4	15645	1.6 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1525 ± 2	21 ± 0
1.0	50	1	14824	20.3 ± 0.6	13.8 ± 0.0	97 ± 0.1	69 ± 0	309 ± 1	16 ± 0
1.0	50	2	15017	8.8 ± 0.5	14.3 ± 0.0	91 ± 0.2	66 ± 0	704 ± 1	18 ± 0
1.0	50	3	21998	3.7 ± 0.3	14.6 ± 0.0	96 ± 0.1	65 ± 0	1117 ± 1	20 ± 0
1.0	50	4	14525	1.5 ± 0.2	15.0 ± 0.0	76 ± 0.1	58 ± 0	1524 ± 2	22 ± 0
2.0	0	1	21494	20.7 ± 0.5	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	0	2	17485	9.0 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1068 ± 1	18 ± 0
2.0	0	3	17543	3.8 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1848 ± 2	20 ± 0
2.0	0	4	21497	1.8 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2603 ± 3	21 ± 0
2.0	10	1	18194	20.5 ± 0.6	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	10	2	17724	8.5 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1069 ± 1	18 ± 0
2.0	10	3	17429	3.9 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1846 ± 2	20 ± 0
2.0	10	4	21495	1.8 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2599 ± 4	21 ± 0
2.0	20	1	21494	21.0 ± 0.5	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	20	2	21494	8.7 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1068 ± 1	18 ± 0
2.0	20	3	21497	4.0 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1847 ± 2	20 ± 0
2.0	20	4	21495	1.7 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2599 ± 3	21 ± 0
2.0	30	1	21492	21.1 ± 0.5	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	30	2	18170	8.9 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1069 ± 1	18 ± 0
2.0	30	3	20334	3.6 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1848 ± 2	20 ± 0
2.0	30	4	21489	1.8 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2599 ± 4	21 ± 0
2.0	40	1	20980	20.7 ± 0.5	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	40	2	17552	8.6 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1070 ± 1	18 ± 0
2.0	40	3	17888	3.7 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1849 ± 2	20 ± 0
2.0	40	4	17942	1.7 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2603 ± 4	21 ± 0
2.0	50	1	18807	20.3 ± 0.6	23.1 ± 0.0	97 ± 0.1	41 ± 0	308 ± 1	16 ± 0
2.0	50	2	21996	8.7 ± 0.4	24.1 ± 0.0	96 ± 0.1	42 ± 0	1069 ± 1	18 ± 0
2.0	50	3	18072	3.9 ± 0.3	24.8 ± 0.0	96 ± 0.1	38 ± 0	1848 ± 2	20 ± 0
2.0	50	4	20188	1.7 ± 0.2	25.5 ± 0.0	76 ± 0.1	34 ± 0	2601 ± 4	21 ± 0

Table E.17: Confidence Intervals for ILPRouting on TOSSIM Network Size 11

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	0	1	21499	12.9 ± 0.4	9.1 ± 0.0	88 ± 0.1	205 ± 0	466 ± 1	26 ± 0
0.25	0	2	21491	7.6 ± 0.4	9.2 ± 0.0	87 ± 0.1	200 ± 0	589 ± 1	30 ± 0
0.25	0	3	21495	3.3 ± 0.2	9.3 ± 0.0	85 ± 0.1	195 ± 0	724 ± 1	32 ± 0
0.25	0	4	21497	1.6 ± 0.2	9.3 ± 0.0	78 ± 0.1	188 ± 0	881 ± 1	34 ± 0
0.25	10	1	21494	10.8 ± 0.4	9.1 ± 0.0	88 ± 0.1	204 ± 0	459 ± 1	27 ± 0
0.25	10	2	21496	6.5 ± 0.3	9.2 ± 0.0	87 ± 0.1	200 ± 0	578 ± 1	30 ± 0
0.25	10	3	21495	3.0 ± 0.2	9.3 ± 0.0	86 ± 0.1	196 ± 0	711 ± 1	32 ± 0
0.25	10	4	21497	1.6 ± 0.2	9.3 ± 0.0	78 ± 0.1	187 ± 0	867 ± 1	34 ± 0
0.25	20	1	21491	9.7 ± 0.4	9.2 ± 0.0	89 ± 0.1	204 ± 0	450 ± 1	27 ± 0
0.25	20	2	21496	5.5 ± 0.3	9.2 ± 0.0	87 ± 0.1	200 ± 0	571 ± 1	30 ± 0
0.25	20	3	21497	2.6 ± 0.2	9.3 ± 0.0	86 ± 0.1	196 ± 0	700 ± 1	32 ± 0
0.25	20	4	21500	1.4 ± 0.2	9.3 ± 0.0	78 ± 0.1	187 ± 0	856 ± 1	33 ± 0
0.25	30	1	21497	7.8 ± 0.4	9.2 ± 0.0	89 ± 0.1	203 ± 0	441 ± 1	27 ± 0
0.25	30	2	21495	4.4 ± 0.3	9.2 ± 0.0	87 ± 0.1	199 ± 0	562 ± 1	30 ± 0
0.25	30	3	21495	2.1 ± 0.2	9.3 ± 0.0	87 ± 0.1	196 ± 0	690 ± 1	32 ± 0
0.25	30	4	21495	1.3 ± 0.2	9.3 ± 0.0	79 ± 0.1	187 ± 0	845 ± 1	33 ± 0
0.25	40	1	21497	6.9 ± 0.3	9.2 ± 0.0	90 ± 0.1	203 ± 0	434 ± 1	27 ± 0
0.25	40	2	21495	4.1 ± 0.3	9.2 ± 0.0	88 ± 0.1	200 ± 0	554 ± 1	29 ± 0
0.25	40	3	21499	2.1 ± 0.2	9.3 ± 0.0	87 ± 0.1	196 ± 0	682 ± 1	31 ± 0
0.25	40	4	21491	1.3 ± 0.2	9.3 ± 0.0	79 ± 0.1	186 ± 0	836 ± 1	32 ± 0
0.25	50	1	21997	6.7 ± 0.3	9.2 ± 0.0	90 ± 0.1	203 ± 0	425 ± 1	26 ± 0
0.25	50	2	21996	3.9 ± 0.3	9.2 ± 0.0	88 ± 0.1	199 ± 0	545 ± 1	28 ± 0
0.25	50	3	22002	1.8 ± 0.2	9.3 ± 0.0	88 ± 0.1	196 ± 0	671 ± 1	30 ± 0
0.25	50	4	21996	1.3 ± 0.2	9.3 ± 0.0	79 ± 0.1	186 ± 0	828 ± 1	32 ± 0
0.5	0	1	21496	12.0 ± 0.4	13.4 ± 0.0	92 ± 0.1	139 ± 0	446 ± 1	27 ± 0
0.5	0	2	21498	6.5 ± 0.3	13.5 ± 0.0	91 ± 0.1	135 ± 0	678 ± 1	30 ± 0
0.5	0	3	21492	2.9 ± 0.2	13.6 ± 0.0	90 ± 0.1	132 ± 0	922 ± 1	32 ± 0
0.5	0	4	21495	1.0 ± 0.1	13.7 ± 0.0	82 ± 0.1	127 ± 0	1179 ± 1	34 ± 0
0.5	10	1	21500	9.9 ± 0.4	13.4 ± 0.0	93 ± 0.1	139 ± 0	438 ± 1	28 ± 0
0.5	10	2	21493	5.2 ± 0.3	13.5 ± 0.0	91 ± 0.1	135 ± 0	667 ± 1	30 ± 0
0.5	10	3	21495	2.2 ± 0.2	13.6 ± 0.0	90 ± 0.1	133 ± 0	908 ± 1	32 ± 0
0.5	10	4	21492	0.9 ± 0.1	13.7 ± 0.0	83 ± 0.1	126 ± 0	1167 ± 1	34 ± 0
0.5	20	1	21493	8.3 ± 0.4	13.4 ± 0.0	93 ± 0.1	138 ± 0	431 ± 1	28 ± 0
0.5	20	2	21496	4.5 ± 0.3	13.6 ± 0.0	92 ± 0.1	135 ± 0	658 ± 1	30 ± 0
0.5	20	3	21497	2.0 ± 0.2	13.7 ± 0.0	91 ± 0.1	132 ± 0	897 ± 1	32 ± 0
0.5	20	4	21496	0.9 ± 0.1	13.7 ± 0.0	83 ± 0.1	126 ± 0	1156 ± 1	34 ± 0
0.5	30	1	21498	7.1 ± 0.3	13.4 ± 0.0	93 ± 0.1	138 ± 0	424 ± 1	28 ± 0
0.5	30	2	21501	3.6 ± 0.3	13.6 ± 0.0	92 ± 0.1	135 ± 0	650 ± 1	30 ± 0
0.5	30	3	21495	1.7 ± 0.2	13.7 ± 0.0	91 ± 0.1	133 ± 0	886 ± 1	31 ± 0
0.5	30	4	21494	0.9 ± 0.1	13.7 ± 0.0	83 ± 0.1	126 ± 0	1146 ± 1	33 ± 0
0.5	40	1	21497	6.4 ± 0.3	13.4 ± 0.0	93 ± 0.1	138 ± 0	416 ± 1	27 ± 0
0.5	40	2	21500	3.4 ± 0.2	13.6 ± 0.0	93 ± 0.1	135 ± 0	641 ± 1	29 ± 0
0.5	40	3	21494	1.4 ± 0.2	13.7 ± 0.0	92 ± 0.1	132 ± 0	876 ± 1	31 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	40	4	21499	0.7 ± 0.1	13.7 ± 0.0	83 ± 0.1	125 ± 0	1138 ± 1	33 ± 0
0.5	50	1	22000	6.0 ± 0.3	13.4 ± 0.0	94 ± 0.1	138 ± 0	409 ± 1	26 ± 0
0.5	50	2	21996	3.2 ± 0.2	13.6 ± 0.0	93 ± 0.1	134 ± 0	633 ± 1	28 ± 0
0.5	50	3	21996	1.3 ± 0.1	13.7 ± 0.0	93 ± 0.1	132 ± 0	866 ± 1	30 ± 0
0.5	50	4	21997	0.7 ± 0.1	13.7 ± 0.0	83 ± 0.1	125 ± 0	1130 ± 1	33 ± 0
1.0	0	1	21492	12.7 ± 0.4	21.9 ± 0.0	90 ± 0.1	84 ± 0	445 ± 1	27 ± 0
1.0	0	2	21490	6.4 ± 0.3	22.2 ± 0.0	91 ± 0.1	82 ± 0	885 ± 1	30 ± 0
1.0	0	3	21493	2.7 ± 0.2	22.5 ± 0.0	91 ± 0.1	81 ± 0	1336 ± 1	32 ± 0
1.0	0	4	21494	1.0 ± 0.1	22.7 ± 0.0	82 ± 0.1	77 ± 0	1795 ± 2	34 ± 0
1.0	10	1	21491	10.3 ± 0.4	22.0 ± 0.0	91 ± 0.1	84 ± 0	437 ± 1	27 ± 0
1.0	10	2	21494	5.4 ± 0.3	22.3 ± 0.0	91 ± 0.1	82 ± 0	874 ± 1	30 ± 0
1.0	10	3	21499	2.3 ± 0.2	22.5 ± 0.0	92 ± 0.1	82 ± 0	1321 ± 1	32 ± 0
1.0	10	4	21496	1.0 ± 0.1	22.7 ± 0.0	83 ± 0.1	76 ± 0	1781 ± 1	34 ± 0
1.0	20	1	21495	8.8 ± 0.4	22.0 ± 0.0	91 ± 0.1	84 ± 0	430 ± 1	28 ± 0
1.0	20	2	21501	4.6 ± 0.3	22.3 ± 0.0	91 ± 0.1	82 ± 0	866 ± 1	30 ± 0
1.0	20	3	21498	2.1 ± 0.2	22.5 ± 0.0	92 ± 0.1	81 ± 0	1311 ± 1	32 ± 0
1.0	20	4	21499	0.7 ± 0.1	22.7 ± 0.0	83 ± 0.1	76 ± 0	1772 ± 1	34 ± 0
1.0	30	1	21498	7.8 ± 0.4	22.0 ± 0.0	92 ± 0.1	83 ± 0	421 ± 1	27 ± 0
1.0	30	2	21494	4.0 ± 0.3	22.3 ± 0.0	92 ± 0.1	82 ± 0	856 ± 1	30 ± 0
1.0	30	3	21501	1.6 ± 0.2	22.5 ± 0.0	92 ± 0.1	81 ± 0	1299 ± 1	31 ± 0
1.0	30	4	21497	0.8 ± 0.1	22.7 ± 0.0	83 ± 0.1	76 ± 0	1760 ± 1	33 ± 0
1.0	40	1	21498	6.9 ± 0.3	22.0 ± 0.0	92 ± 0.1	84 ± 0	414 ± 1	27 ± 0
1.0	40	2	21499	3.5 ± 0.2	22.3 ± 0.0	93 ± 0.1	82 ± 0	847 ± 1	29 ± 0
1.0	40	3	21500	1.4 ± 0.2	22.5 ± 0.0	93 ± 0.1	81 ± 0	1289 ± 1	31 ± 0
1.0	40	4	21500	0.7 ± 0.1	22.7 ± 0.0	83 ± 0.1	76 ± 0	1751 ± 1	33 ± 0
1.0	50	1	21998	6.6 ± 0.3	22.0 ± 0.0	93 ± 0.1	83 ± 0	407 ± 1	26 ± 0
1.0	50	2	22001	3.2 ± 0.2	22.3 ± 0.0	93 ± 0.1	82 ± 0	838 ± 1	28 ± 0
1.0	50	3	21998	1.4 ± 0.2	22.5 ± 0.0	93 ± 0.1	81 ± 0	1278 ± 1	30 ± 0
1.0	50	4	21998	0.5 ± 0.1	22.7 ± 0.0	83 ± 0.1	75 ± 0	1741 ± 1	33 ± 0
2.0	0	1	21497	12.3 ± 0.4	39.5 ± 0.0	94 ± 0.1	48 ± 0	442 ± 1	27 ± 0
2.0	0	2	21494	6.6 ± 0.3	40.1 ± 0.0	93 ± 0.1	47 ± 0	1301 ± 1	30 ± 0
2.0	0	3	21493	2.9 ± 0.2	40.6 ± 0.0	93 ± 0.1	46 ± 0	2163 ± 2	32 ± 0
2.0	0	4	21494	1.1 ± 0.1	40.9 ± 0.0	82 ± 0.1	42 ± 0	3036 ± 3	34 ± 0
2.0	10	1	21492	10.8 ± 0.4	39.5 ± 0.0	94 ± 0.1	48 ± 0	435 ± 1	27 ± 0
2.0	10	2	21500	5.3 ± 0.3	40.2 ± 0.0	93 ± 0.1	47 ± 0	1292 ± 1	30 ± 0
2.0	10	3	21498	2.2 ± 0.2	40.6 ± 0.0	93 ± 0.1	46 ± 0	2149 ± 2	32 ± 0
2.0	10	4	21496	1.1 ± 0.1	40.9 ± 0.0	83 ± 0.1	42 ± 0	3022 ± 2	34 ± 0
2.0	20	1	21499	8.5 ± 0.4	39.6 ± 0.0	94 ± 0.1	48 ± 0	427 ± 1	28 ± 0
2.0	20	2	21495	4.6 ± 0.3	40.2 ± 0.1	94 ± 0.1	47 ± 0	1281 ± 1	30 ± 0
2.0	20	3	21498	1.8 ± 0.2	40.6 ± 0.0	93 ± 0.1	46 ± 0	2138 ± 2	32 ± 0
2.0	20	4	21500	0.8 ± 0.1	40.9 ± 0.0	83 ± 0.1	42 ± 0	3009 ± 3	34 ± 0
2.0	30	1	21498	7.8 ± 0.4	39.6 ± 0.0	94 ± 0.1	47 ± 0	421 ± 1	28 ± 0
2.0	30	2	21500	3.7 ± 0.3	40.2 ± 0.0	94 ± 0.1	46 ± 0	1272 ± 1	30 ± 0
2.0	30	3	21498	1.7 ± 0.2	40.6 ± 0.0	94 ± 0.1	46 ± 0	2125 ± 2	32 ± 0
2.0	30	4	21499	0.8 ± 0.1	40.9 ± 0.0	83 ± 0.1	42 ± 0	2999 ± 2	33 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	40	1	21500	7.2 ± 0.3	39.7 ± 0.0	94 ± 0.1	47 ± 0	413 ± 1	27 ± 0
2.0	40	2	21500	3.4 ± 0.2	40.2 ± 0.0	94 ± 0.1	46 ± 0	1262 ± 1	29 ± 0
2.0	40	3	21498	1.6 ± 0.2	40.6 ± 0.0	94 ± 0.1	46 ± 0	2115 ± 2	31 ± 0
2.0	40	4	21496	0.7 ± 0.1	40.9 ± 0.0	83 ± 0.1	42 ± 0	2988 ± 2	33 ± 0
2.0	50	1	21998	6.2 ± 0.3	39.7 ± 0.0	95 ± 0.1	47 ± 0	406 ± 1	26 ± 0
2.0	50	2	21996	3.3 ± 0.2	40.2 ± 0.0	94 ± 0.1	46 ± 0	1253 ± 1	29 ± 0
2.0	50	3	21999	1.4 ± 0.2	40.6 ± 0.0	94 ± 0.1	46 ± 0	2103 ± 2	30 ± 0
2.0	50	4	22001	0.7 ± 0.1	40.9 ± 0.0	83 ± 0.1	42 ± 0	2976 ± 2	33 ± 0

Table E.18: Confidence Intervals for ILPRouting on TOSSIM Network Size 15

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	0	1	21498	14.2 ± 0.5	11.2 ± 0.0	86 ± 0.2	276 ± 0	592 ± 1	36 ± 0
0.25	0	2	21498	8.5 ± 0.4	11.3 ± 0.0	82 ± 0.2	270 ± 0	722 ± 1	40 ± 0
0.25	0	3	21493	4.4 ± 0.3	11.3 ± 0.0	80 ± 0.2	264 ± 0	870 ± 1	42 ± 0
0.25	0	4	21496	1.4 ± 0.2	11.4 ± 0.0	76 ± 0.1	259 ± 0	1025 ± 1	46 ± 0
0.25	10	1	21499	10.7 ± 0.4	11.2 ± 0.0	86 ± 0.2	274 ± 0	576 ± 1	38 ± 0
0.25	10	2	21495	6.5 ± 0.3	11.3 ± 0.0	82 ± 0.1	269 ± 0	703 ± 1	41 ± 0
0.25	10	3	21499	3.3 ± 0.2	11.3 ± 0.0	81 ± 0.1	264 ± 0	845 ± 1	43 ± 0
0.25	10	4	21497	1.1 ± 0.1	11.4 ± 0.0	76 ± 0.1	259 ± 0	994 ± 1	46 ± 0
0.25	20	1	21496	7.4 ± 0.4	11.2 ± 0.0	87 ± 0.1	273 ± 0	559 ± 1	39 ± 0
0.25	20	2	21497	4.9 ± 0.3	11.3 ± 0.0	83 ± 0.1	269 ± 0	685 ± 1	41 ± 0
0.25	20	3	21499	2.4 ± 0.2	11.3 ± 0.0	83 ± 0.1	263 ± 0	819 ± 1	43 ± 0
0.25	20	4	21497	1.0 ± 0.1	11.4 ± 0.0	77 ± 0.1	258 ± 0	968 ± 1	45 ± 0
0.25	30	1	21500	5.5 ± 0.3	11.3 ± 0.0	88 ± 0.1	271 ± 0	543 ± 1	39 ± 0
0.25	30	2	21497	3.6 ± 0.3	11.3 ± 0.0	84 ± 0.1	268 ± 0	668 ± 1	41 ± 0
0.25	30	3	21496	1.8 ± 0.2	11.3 ± 0.0	84 ± 0.1	262 ± 0	796 ± 1	42 ± 0
0.25	30	4	21498	0.9 ± 0.1	11.4 ± 0.0	78 ± 0.1	256 ± 0	946 ± 1	45 ± 0
0.25	40	1	21498	5.6 ± 0.3	11.3 ± 0.0	89 ± 0.1	270 ± 0	527 ± 1	38 ± 0
0.25	40	2	21497	3.7 ± 0.3	11.3 ± 0.0	84 ± 0.1	266 ± 0	651 ± 1	40 ± 0
0.25	40	3	21494	2.0 ± 0.2	11.3 ± 0.0	85 ± 0.1	262 ± 0	778 ± 1	41 ± 0
0.25	40	4	21497	1.2 ± 0.1	11.4 ± 0.0	79 ± 0.1	255 ± 0	926 ± 1	43 ± 0
0.25	50	1	22000	8.0 ± 0.4	11.2 ± 0.0	90 ± 0.1	269 ± 0	510 ± 1	34 ± 0
0.25	50	2	21997	5.7 ± 0.3	11.3 ± 0.0	85 ± 0.1	264 ± 0	635 ± 1	37 ± 0
0.25	50	3	21998	3.4 ± 0.2	11.3 ± 0.0	87 ± 0.1	260 ± 0	759 ± 1	38 ± 0
0.25	50	4	21995	1.9 ± 0.2	11.4 ± 0.0	79 ± 0.1	253 ± 0	908 ± 1	41 ± 0
0.5	0	1	21496	12.4 ± 0.4	17.5 ± 0.0	87 ± 0.2	177 ± 0	575 ± 1	37 ± 0
0.5	0	2	21497	7.0 ± 0.3	17.6 ± 0.0	86 ± 0.2	174 ± 0	819 ± 1	41 ± 0
0.5	0	3	21492	4.0 ± 0.3	17.8 ± 0.0	79 ± 0.1	167 ± 0	1080 ± 1	42 ± 0
0.5	0	4	21499	1.5 ± 0.2	17.8 ± 0.0	79 ± 0.1	165 ± 0	1338 ± 1	47 ± 0
0.5	10	1	21499	9.4 ± 0.4	17.6 ± 0.0	87 ± 0.2	176 ± 0	560 ± 1	39 ± 0
0.5	10	2	21497	5.8 ± 0.3	17.7 ± 0.0	86 ± 0.2	173 ± 0	799 ± 1	41 ± 0
0.5	10	3	21493	2.7 ± 0.2	17.8 ± 0.0	80 ± 0.1	167 ± 0	1056 ± 1	43 ± 0
0.5	10	4	21499	1.1 ± 0.1	17.9 ± 0.0	79 ± 0.1	164 ± 0	1310 ± 1	47 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	20	1	21502	6.4 ± 0.3	17.6 ± 0.0	87 ± 0.1	175 ± 0	544 ± 1	40 ± 0
0.5	20	2	21497	4.0 ± 0.3	17.7 ± 0.0	87 ± 0.1	172 ± 0	781 ± 1	42 ± 0
0.5	20	3	21497	2.1 ± 0.2	17.8 ± 0.0	82 ± 0.1	166 ± 0	1030 ± 1	43 ± 0
0.5	20	4	21499	0.9 ± 0.1	17.8 ± 0.0	80 ± 0.1	163 ± 0	1287 ± 1	46 ± 0
0.5	30	1	21500	5.0 ± 0.3	17.6 ± 0.0	88 ± 0.1	174 ± 0	529 ± 1	40 ± 0
0.5	30	2	21500	3.0 ± 0.2	17.7 ± 0.0	87 ± 0.1	171 ± 0	762 ± 1	42 ± 0
0.5	30	3	21496	1.6 ± 0.2	17.8 ± 0.0	83 ± 0.1	165 ± 0	1008 ± 1	42 ± 0
0.5	30	4	21501	0.7 ± 0.1	17.9 ± 0.0	80 ± 0.1	162 ± 0	1263 ± 1	46 ± 0
0.5	40	1	21501	4.5 ± 0.3	17.6 ± 0.0	88 ± 0.1	173 ± 0	512 ± 1	38 ± 0
0.5	40	2	21500	3.1 ± 0.2	17.7 ± 0.0	88 ± 0.1	170 ± 0	745 ± 1	40 ± 0
0.5	40	3	21501	1.7 ± 0.2	17.8 ± 0.0	84 ± 0.1	164 ± 0	989 ± 1	41 ± 0
0.5	40	4	21498	0.9 ± 0.1	17.8 ± 0.0	81 ± 0.1	161 ± 0	1243 ± 1	45 ± 0
0.5	50	1	21994	6.9 ± 0.3	17.6 ± 0.0	89 ± 0.1	172 ± 0	496 ± 1	35 ± 0
0.5	50	2	21997	4.5 ± 0.3	17.7 ± 0.0	88 ± 0.1	169 ± 0	728 ± 1	38 ± 0
0.5	50	3	21998	2.4 ± 0.2	17.8 ± 0.0	85 ± 0.1	163 ± 0	969 ± 1	39 ± 0
0.5	50	4	22000	1.2 ± 0.1	17.8 ± 0.0	81 ± 0.1	160 ± 0	1224 ± 1	43 ± 0
1.0	0	1	21495	10.2 ± 0.4	30.2 ± 0.0	90 ± 0.2	103 ± 0	573 ± 1	38 ± 0
1.0	0	2	21500	5.7 ± 0.3	30.4 ± 0.0	89 ± 0.2	101 ± 0	1037 ± 1	41 ± 0
1.0	0	3	21498	2.9 ± 0.2	30.6 ± 0.0	82 ± 0.2	96 ± 0	1514 ± 1	44 ± 0
1.0	0	4	21498	1.4 ± 0.2	30.8 ± 0.0	82 ± 0.2	95 ± 0	1984 ± 2	46 ± 0
1.0	10	1	21495	7.5 ± 0.4	30.3 ± 0.0	90 ± 0.2	102 ± 0	558 ± 1	39 ± 0
1.0	10	2	21502	4.4 ± 0.3	30.5 ± 0.0	90 ± 0.2	101 ± 0	1017 ± 1	42 ± 0
1.0	10	3	21500	2.4 ± 0.2	30.7 ± 0.0	83 ± 0.1	96 ± 0	1490 ± 1	44 ± 0
1.0	10	4	21499	1.0 ± 0.1	30.8 ± 0.0	82 ± 0.1	95 ± 0	1958 ± 2	46 ± 0
1.0	20	1	21498	5.8 ± 0.3	30.3 ± 0.0	91 ± 0.1	102 ± 0	540 ± 1	40 ± 0
1.0	20	2	21497	3.3 ± 0.2	30.5 ± 0.0	90 ± 0.1	100 ± 0	998 ± 1	42 ± 0
1.0	20	3	21497	1.5 ± 0.2	30.7 ± 0.0	83 ± 0.1	95 ± 0	1469 ± 1	45 ± 0
1.0	20	4	21497	0.8 ± 0.1	30.8 ± 0.0	83 ± 0.1	94 ± 0	1931 ± 2	46 ± 0
1.0	30	1	21499	3.9 ± 0.3	30.4 ± 0.0	91 ± 0.1	101 ± 0	525 ± 1	40 ± 0
1.0	30	2	21496	2.5 ± 0.2	30.6 ± 0.0	91 ± 0.1	100 ± 0	979 ± 1	42 ± 0
1.0	30	3	21496	1.3 ± 0.1	30.7 ± 0.0	84 ± 0.1	95 ± 0	1449 ± 1	44 ± 0
1.0	30	4	21499	0.6 ± 0.1	30.8 ± 0.0	84 ± 0.1	94 ± 0	1909 ± 1	46 ± 0
1.0	40	1	21497	4.0 ± 0.3	30.4 ± 0.0	91 ± 0.1	101 ± 0	509 ± 1	38 ± 0
1.0	40	2	21497	2.2 ± 0.2	30.6 ± 0.0	91 ± 0.1	99 ± 0	961 ± 1	41 ± 0
1.0	40	3	21498	1.3 ± 0.1	30.7 ± 0.0	84 ± 0.1	94 ± 0	1429 ± 1	43 ± 0
1.0	40	4	21499	0.6 ± 0.1	30.8 ± 0.0	84 ± 0.1	93 ± 0	1888 ± 1	45 ± 0
1.0	50	1	21998	5.6 ± 0.3	30.4 ± 0.0	92 ± 0.1	100 ± 0	493 ± 1	36 ± 0
1.0	50	2	21999	3.5 ± 0.2	30.5 ± 0.0	92 ± 0.1	99 ± 0	942 ± 1	38 ± 0
1.0	50	3	22000	2.0 ± 0.2	30.7 ± 0.0	85 ± 0.1	94 ± 0	1410 ± 1	41 ± 0
1.0	50	4	21995	1.1 ± 0.1	30.8 ± 0.0	85 ± 0.1	92 ± 0	1865 ± 1	43 ± 0
2.0	0	1	21497	10.7 ± 0.4	55.9 ± 0.0	90 ± 0.2	56 ± 0	572 ± 1	38 ± 0
2.0	0	2	21498	5.9 ± 0.3	56.5 ± 0.0	89 ± 0.2	55 ± 0	1474 ± 1	41 ± 0
2.0	0	3	21501	3.4 ± 0.2	56.9 ± 0.0	82 ± 0.2	52 ± 0	2387 ± 2	44 ± 0
2.0	0	4	21498	1.5 ± 0.2	57.2 ± 0.0	82 ± 0.2	51 ± 0	3289 ± 3	46 ± 0
2.0	10	1	21499	8.0 ± 0.4	56.1 ± 0.0	90 ± 0.2	55 ± 0	557 ± 1	39 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	10	2	21502	4.5 ± 0.3	56.6 ± 0.0	90 ± 0.2	54 ± 0	1454 ± 1	42 ± 0
2.0	10	3	21500	2.5 ± 0.2	56.9 ± 0.0	83 ± 0.1	52 ± 0	2362 ± 2	44 ± 0
2.0	10	4	21498	1.1 ± 0.1	57.2 ± 0.0	83 ± 0.1	51 ± 0	3259 ± 3	46 ± 0
2.0	20	1	21496	5.4 ± 0.3	56.3 ± 0.0	91 ± 0.1	55 ± 0	541 ± 1	40 ± 0
2.0	20	2	21497	3.3 ± 0.2	56.7 ± 0.0	90 ± 0.1	54 ± 0	1433 ± 1	42 ± 0
2.0	20	3	21500	1.7 ± 0.2	57.0 ± 0.0	84 ± 0.1	51 ± 0	2339 ± 2	44 ± 0
2.0	20	4	21500	0.8 ± 0.1	57.2 ± 0.0	83 ± 0.1	51 ± 0	3233 ± 2	46 ± 0
2.0	30	1	21499	4.1 ± 0.3	56.4 ± 0.0	91 ± 0.1	55 ± 0	525 ± 1	40 ± 0
2.0	30	2	21499	2.5 ± 0.2	56.7 ± 0.0	91 ± 0.1	54 ± 0	1412 ± 1	42 ± 0
2.0	30	3	21499	1.4 ± 0.2	57.0 ± 0.0	84 ± 0.1	51 ± 0	2319 ± 2	44 ± 0
2.0	30	4	21498	0.7 ± 0.1	57.2 ± 0.0	84 ± 0.1	50 ± 0	3207 ± 2	46 ± 0
2.0	40	1	21501	4.1 ± 0.3	56.4 ± 0.0	92 ± 0.1	54 ± 0	510 ± 1	38 ± 0
2.0	40	2	21497	2.4 ± 0.2	56.7 ± 0.0	91 ± 0.1	53 ± 0	1395 ± 1	40 ± 0
2.0	40	3	21500	1.5 ± 0.2	57.0 ± 0.1	85 ± 0.1	51 ± 0	2299 ± 2	42 ± 0
2.0	40	4	21501	0.9 ± 0.1	57.2 ± 0.0	84 ± 0.1	50 ± 0	3182 ± 2	44 ± 0
2.0	50	1	22000	5.3 ± 0.3	56.4 ± 0.0	92 ± 0.1	54 ± 0	493 ± 1	36 ± 0
2.0	50	2	22001	3.7 ± 0.2	56.7 ± 0.1	92 ± 0.1	53 ± 0	1374 ± 1	38 ± 0
2.0	50	3	22000	2.1 ± 0.2	57.0 ± 0.0	86 ± 0.1	51 ± 0	2277 ± 2	40 ± 0
2.0	50	4	21998	1.2 ± 0.1	57.3 ± 0.1	85 ± 0.1	50 ± 0	3159 ± 2	43 ± 0

Table E.19: Confidence Intervals for ILPRouting on TOSSIM Network Size 21

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	0	1	21496	12.6 ± 0.4	14.3 ± 0.0	79 ± 0.2	370 ± 1	774 ± 1	53 ± 0
0.25	0	2	21497	9.0 ± 0.4	14.4 ± 0.0	77 ± 0.2	365 ± 1	910 ± 1	56 ± 0
0.25	0	3	21492	5.0 ± 0.3	14.4 ± 0.0	75 ± 0.2	359 ± 1	1057 ± 1	60 ± 0
0.25	0	4	21495	2.3 ± 0.2	14.5 ± 0.0	73 ± 0.2	354 ± 1	1214 ± 1	63 ± 0
0.25	10	1	21500	8.6 ± 0.4	14.4 ± 0.0	80 ± 0.2	368 ± 1	744 ± 1	56 ± 0
0.25	10	2	21496	6.0 ± 0.3	14.4 ± 0.0	79 ± 0.2	362 ± 1	876 ± 1	58 ± 0
0.25	10	3	21498	3.4 ± 0.2	14.5 ± 0.0	77 ± 0.2	358 ± 1	1017 ± 1	61 ± 0
0.25	10	4	21496	1.6 ± 0.2	14.5 ± 0.0	75 ± 0.2	353 ± 0	1164 ± 1	63 ± 0
0.25	20	1	21501	5.5 ± 0.3	14.4 ± 0.0	81 ± 0.2	366 ± 1	715 ± 1	57 ± 0
0.25	20	2	21493	4.0 ± 0.3	14.4 ± 0.0	80 ± 0.2	361 ± 0	844 ± 1	59 ± 0
0.25	20	3	21498	2.3 ± 0.2	14.5 ± 0.0	79 ± 0.2	356 ± 0	978 ± 1	61 ± 0
0.25	20	4	21500	1.1 ± 0.1	14.5 ± 0.0	77 ± 0.2	352 ± 0	1118 ± 1	62 ± 0
0.25	30	1	21495	4.3 ± 0.3	14.4 ± 0.0	83 ± 0.2	363 ± 0	685 ± 1	56 ± 0
0.25	30	2	21499	3.1 ± 0.2	14.5 ± 0.0	82 ± 0.2	358 ± 0	811 ± 1	58 ± 0
0.25	30	3	21496	1.7 ± 0.2	14.5 ± 0.0	80 ± 0.2	354 ± 0	942 ± 1	60 ± 0
0.25	30	4	21500	1.2 ± 0.1	14.5 ± 0.0	79 ± 0.2	350 ± 0	1079 ± 1	61 ± 0
0.25	40	1	21497	6.3 ± 0.3	14.4 ± 0.0	84 ± 0.2	361 ± 0	655 ± 1	51 ± 0
0.25	40	2	21499	4.9 ± 0.3	14.4 ± 0.0	84 ± 0.1	356 ± 0	781 ± 1	53 ± 0
0.25	40	3	21497	3.4 ± 0.2	14.5 ± 0.0	82 ± 0.1	352 ± 0	909 ± 1	56 ± 0
0.25	40	4	21494	2.2 ± 0.2	14.5 ± 0.0	81 ± 0.1	347 ± 0	1043 ± 1	57 ± 0
0.25	50	1	21992	14.0 ± 0.5	14.3 ± 0.0	86 ± 0.1	359 ± 0	628 ± 1	43 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	50	2	21998	11.0 ± 0.4	14.3 ± 0.0	85 ± 0.1	354 ± 0	751 ± 1	45 ± 0
0.25	50	3	21996	7.7 ± 0.4	14.4 ± 0.0	84 ± 0.1	350 ± 0	880 ± 1	49 ± 0
0.25	50	4	21998	5.3 ± 0.3	14.4 ± 0.0	83 ± 0.1	345 ± 0	1014 ± 1	52 ± 0
0.5	0	1	21496	12.1 ± 0.4	23.8 ± 0.0	80 ± 0.2	224 ± 0	756 ± 1	53 ± 0
0.5	0	2	21495	8.0 ± 0.4	23.9 ± 0.0	78 ± 0.2	220 ± 0	1013 ± 1	57 ± 0
0.5	0	3	21497	5.0 ± 0.3	24.0 ± 0.0	77 ± 0.2	217 ± 0	1271 ± 1	60 ± 0
0.5	0	4	21494	1.8 ± 0.2	24.1 ± 0.0	74 ± 0.2	212 ± 0	1544 ± 1	64 ± 0
0.5	10	1	21500	8.1 ± 0.4	23.9 ± 0.0	81 ± 0.2	223 ± 0	726 ± 1	56 ± 0
0.5	10	2	21498	5.3 ± 0.3	24.0 ± 0.0	79 ± 0.2	219 ± 0	978 ± 1	59 ± 0
0.5	10	3	21497	3.3 ± 0.2	24.1 ± 0.0	79 ± 0.2	217 ± 0	1229 ± 1	61 ± 0
0.5	10	4	21497	1.5 ± 0.2	24.1 ± 0.0	75 ± 0.2	211 ± 0	1496 ± 1	64 ± 0
0.5	20	1	21500	5.1 ± 0.3	23.9 ± 0.0	82 ± 0.2	221 ± 0	698 ± 1	57 ± 0
0.5	20	2	21500	3.4 ± 0.2	24.0 ± 0.0	81 ± 0.2	217 ± 0	944 ± 1	59 ± 0
0.5	20	3	21501	2.1 ± 0.2	24.1 ± 0.0	81 ± 0.2	215 ± 0	1189 ± 1	61 ± 0
0.5	20	4	21497	1.0 ± 0.1	24.1 ± 0.0	77 ± 0.2	210 ± 0	1452 ± 1	64 ± 0
0.5	30	1	21497	4.0 ± 0.3	24.0 ± 0.0	84 ± 0.2	220 ± 0	667 ± 1	56 ± 0
0.5	30	2	21497	2.6 ± 0.2	24.0 ± 0.0	82 ± 0.2	216 ± 0	910 ± 1	58 ± 0
0.5	30	3	21500	1.5 ± 0.2	24.1 ± 0.0	82 ± 0.2	214 ± 0	1150 ± 1	60 ± 0
0.5	30	4	21499	1.0 ± 0.1	24.1 ± 0.0	78 ± 0.2	209 ± 0	1413 ± 1	62 ± 0
0.5	40	1	21498	5.6 ± 0.3	23.9 ± 0.0	85 ± 0.2	218 ± 0	640 ± 1	52 ± 0
0.5	40	2	21498	4.2 ± 0.3	24.0 ± 0.0	83 ± 0.1	214 ± 0	881 ± 1	54 ± 0
0.5	40	3	21498	2.9 ± 0.2	24.1 ± 0.0	84 ± 0.1	213 ± 0	1118 ± 1	56 ± 0
0.5	40	4	21499	1.8 ± 0.2	24.1 ± 0.0	80 ± 0.1	208 ± 0	1377 ± 1	58 ± 0
0.5	50	1	21999	13.0 ± 0.4	23.7 ± 0.0	86 ± 0.1	217 ± 0	612 ± 1	44 ± 0
0.5	50	2	21998	10.4 ± 0.4	23.8 ± 0.0	85 ± 0.1	213 ± 0	851 ± 1	46 ± 0
0.5	50	3	21998	7.2 ± 0.3	24.0 ± 0.0	85 ± 0.1	212 ± 0	1088 ± 1	50 ± 0
0.5	50	4	21992	4.6 ± 0.3	24.0 ± 0.0	81 ± 0.1	206 ± 0	1347 ± 1	53 ± 0
1.0	0	1	21497	10.2 ± 0.4	42.6 ± 0.0	82 ± 0.2	125 ± 0	753 ± 1	54 ± 0
1.0	0	2	21497	7.4 ± 0.3	42.9 ± 0.0	81 ± 0.2	123 ± 0	1235 ± 1	57 ± 0
1.0	0	3	21494	4.5 ± 0.3	43.1 ± 0.0	81 ± 0.2	122 ± 0	1720 ± 2	60 ± 0
1.0	0	4	21495	2.3 ± 0.2	43.2 ± 0.0	76 ± 0.2	118 ± 0	2214 ± 2	64 ± 0
1.0	10	1	21498	6.9 ± 0.3	42.8 ± 0.0	83 ± 0.2	124 ± 0	722 ± 1	56 ± 0
1.0	10	2	21499	4.7 ± 0.3	43.0 ± 0.0	82 ± 0.2	122 ± 0	1201 ± 1	59 ± 0
1.0	10	3	21499	2.8 ± 0.2	43.2 ± 0.0	82 ± 0.2	121 ± 0	1676 ± 1	61 ± 0
1.0	10	4	21499	1.4 ± 0.2	43.3 ± 0.0	77 ± 0.2	117 ± 0	2170 ± 2	64 ± 0
1.0	20	1	21500	4.3 ± 0.3	42.9 ± 0.0	85 ± 0.2	123 ± 0	693 ± 1	57 ± 0
1.0	20	2	21497	3.1 ± 0.2	43.1 ± 0.0	83 ± 0.2	121 ± 0	1165 ± 1	60 ± 0
1.0	20	3	21496	1.9 ± 0.2	43.2 ± 0.0	84 ± 0.2	121 ± 0	1636 ± 1	61 ± 0
1.0	20	4	21496	0.9 ± 0.1	43.3 ± 0.0	78 ± 0.2	116 ± 0	2125 ± 2	64 ± 0
1.0	30	1	21499	3.1 ± 0.2	43.0 ± 0.0	86 ± 0.2	123 ± 0	662 ± 1	56 ± 0
1.0	30	2	21499	2.4 ± 0.2	43.1 ± 0.0	84 ± 0.2	120 ± 0	1131 ± 1	59 ± 0
1.0	30	3	21499	1.4 ± 0.2	43.2 ± 0.1	85 ± 0.2	120 ± 0	1596 ± 1	60 ± 0
1.0	30	4	21500	0.9 ± 0.1	43.3 ± 0.0	79 ± 0.2	116 ± 0	2082 ± 2	62 ± 0
1.0	40	1	21499	5.1 ± 0.3	42.9 ± 0.0	87 ± 0.2	122 ± 0	632 ± 1	51 ± 0
1.0	40	2	21498	4.0 ± 0.3	43.0 ± 0.0	85 ± 0.1	119 ± 0	1101 ± 1	54 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	40	3	21500	2.7 ± 0.2	43.2 ± 0.0	87 ± 0.2	119 ± 0	1560 ± 1	56 ± 0
1.0	40	4	21498	1.8 ± 0.2	43.3 ± 0.0	81 ± 0.1	115 ± 0	2045 ± 1	59 ± 0
1.0	50	1	21999	11.4 ± 0.4	42.6 ± 0.0	88 ± 0.1	121 ± 0	605 ± 1	44 ± 0
1.0	50	2	21998	9.2 ± 0.4	42.8 ± 0.0	87 ± 0.1	119 ± 0	1069 ± 1	47 ± 0
1.0	50	3	21998	6.5 ± 0.3	43.0 ± 0.0	88 ± 0.1	118 ± 0	1527 ± 1	50 ± 0
1.0	50	4	21999	4.5 ± 0.3	43.1 ± 0.0	82 ± 0.1	114 ± 0	2009 ± 1	54 ± 0
2.0	0	1	21499	10.2 ± 0.4	80.8 ± 0.1	83 ± 0.2	66 ± 0	753 ± 1	54 ± 0
2.0	0	2	21498	7.4 ± 0.3	81.3 ± 0.0	81 ± 0.2	65 ± 0	1689 ± 2	57 ± 0
2.0	0	3	21497	4.7 ± 0.3	81.7 ± 0.0	82 ± 0.2	65 ± 0	2627 ± 2	59 ± 0
2.0	0	4	21496	2.3 ± 0.2	82.1 ± 0.0	76 ± 0.2	62 ± 0	3574 ± 3	64 ± 0
2.0	10	1	21497	6.8 ± 0.3	81.2 ± 0.0	84 ± 0.2	66 ± 0	722 ± 1	56 ± 0
2.0	10	2	21499	4.9 ± 0.3	81.5 ± 0.0	82 ± 0.2	64 ± 0	1652 ± 2	59 ± 0
2.0	10	3	21501	3.0 ± 0.2	81.9 ± 0.0	83 ± 0.2	65 ± 0	2580 ± 2	61 ± 0
2.0	10	4	21498	1.5 ± 0.2	82.1 ± 0.0	77 ± 0.2	62 ± 0	3525 ± 3	64 ± 0
2.0	20	1	21497	4.3 ± 0.3	81.4 ± 0.0	85 ± 0.2	65 ± 0	692 ± 1	57 ± 0
2.0	20	2	21494	3.1 ± 0.2	81.7 ± 0.0	83 ± 0.2	64 ± 0	1615 ± 1	60 ± 0
2.0	20	3	21501	1.7 ± 0.2	82.1 ± 0.2	85 ± 0.2	64 ± 0	2537 ± 2	62 ± 0
2.0	20	4	21498	0.9 ± 0.1	82.2 ± 0.1	78 ± 0.2	61 ± 0	3476 ± 3	64 ± 0
2.0	30	1	21502	3.1 ± 0.2	81.6 ± 0.0	86 ± 0.2	65 ± 0	662 ± 1	57 ± 0
2.0	30	2	21499	2.3 ± 0.2	81.8 ± 0.0	84 ± 0.2	63 ± 0	1579 ± 1	58 ± 0
2.0	30	3	21496	1.5 ± 0.2	82.0 ± 0.0	86 ± 0.2	64 ± 0	2492 ± 2	60 ± 0
2.0	30	4	21497	0.9 ± 0.1	82.2 ± 0.0	80 ± 0.2	61 ± 0	3427 ± 2	62 ± 0
2.0	40	1	21502	5.0 ± 0.3	81.4 ± 0.0	87 ± 0.2	64 ± 0	633 ± 1	52 ± 0
2.0	40	2	21498	4.0 ± 0.3	81.6 ± 0.0	85 ± 0.1	63 ± 0	1545 ± 1	54 ± 0
2.0	40	3	21497	2.7 ± 0.2	81.9 ± 0.0	87 ± 0.2	63 ± 0	2453 ± 2	56 ± 0
2.0	40	4	21500	1.9 ± 0.2	82.1 ± 0.0	81 ± 0.1	61 ± 0	3384 ± 2	59 ± 0
2.0	50	1	21996	11.3 ± 0.4	80.8 ± 0.0	89 ± 0.1	64 ± 0	605 ± 1	44 ± 0
2.0	50	2	21999	9.3 ± 0.4	81.1 ± 0.0	87 ± 0.1	62 ± 0	1514 ± 1	47 ± 0
2.0	50	3	22000	6.8 ± 0.3	81.5 ± 0.0	89 ± 0.1	63 ± 0	2417 ± 2	50 ± 0
2.0	50	4	22001	5.1 ± 0.3	81.8 ± 0.0	82 ± 0.1	60 ± 0	3347 ± 2	53 ± 0

Table E.20: Confidence Intervals for ILPRouting on TOSSIM Network Size 25

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	0	1	21498	10.2 ± 0.4	16.4 ± 0.0	73 ± 0.2	427 ± 1	895 ± 1	65 ± 0
0.25	0	2	21500	8.0 ± 0.4	16.5 ± 0.0	72 ± 0.2	422 ± 1	1033 ± 1	68 ± 0
0.25	0	3	21497	4.7 ± 0.3	16.5 ± 0.0	71 ± 0.2	416 ± 1	1182 ± 1	71 ± 0
0.25	0	4	21491	2.5 ± 0.2	16.6 ± 0.0	69 ± 0.2	411 ± 1	1336 ± 1	75 ± 0
0.25	10	1	21498	6.5 ± 0.3	16.5 ± 0.0	75 ± 0.2	425 ± 1	855 ± 1	68 ± 0
0.25	10	2	21495	4.8 ± 0.3	16.5 ± 0.0	74 ± 0.2	419 ± 1	989 ± 1	70 ± 0
0.25	10	3	21499	2.8 ± 0.2	16.6 ± 0.0	73 ± 0.2	414 ± 1	1129 ± 1	73 ± 0
0.25	10	4	21498	1.5 ± 0.2	16.6 ± 0.0	72 ± 0.2	410 ± 1	1273 ± 1	75 ± 0
0.25	20	1	21498	4.1 ± 0.3	16.5 ± 0.0	77 ± 0.2	422 ± 1	819 ± 1	69 ± 0
0.25	20	2	21500	3.0 ± 0.2	16.5 ± 0.0	76 ± 0.2	416 ± 1	946 ± 1	71 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	20	3	21500	2.0 ± 0.2	16.6 ± 0.0	75 ± 0.2	412 ± 1	1082 ± 1	72 ± 0
0.25	20	4	21498	1.1 ± 0.1	16.6 ± 0.0	74 ± 0.2	409 ± 1	1219 ± 1	74 ± 0
0.25	30	1	21499	3.6 ± 0.2	16.5 ± 0.0	79 ± 0.2	420 ± 1	779 ± 1	66 ± 0
0.25	30	2	21496	2.7 ± 0.2	16.5 ± 0.0	78 ± 0.2	414 ± 1	905 ± 1	68 ± 0
0.25	30	3	21495	2.0 ± 0.2	16.6 ± 0.0	77 ± 0.2	410 ± 1	1036 ± 1	69 ± 0
0.25	30	4	21501	1.4 ± 0.2	16.6 ± 0.0	76 ± 0.2	406 ± 1	1170 ± 1	71 ± 0
0.25	40	1	21496	7.1 ± 0.3	16.5 ± 0.0	81 ± 0.2	417 ± 1	743 ± 1	58 ± 0
0.25	40	2	21499	5.6 ± 0.3	16.5 ± 0.0	80 ± 0.2	412 ± 1	869 ± 1	60 ± 0
0.25	40	3	21496	4.3 ± 0.3	16.5 ± 0.0	79 ± 0.2	408 ± 1	999 ± 1	63 ± 0
0.25	40	4	21496	3.5 ± 0.2	16.5 ± 0.0	79 ± 0.1	404 ± 1	1130 ± 1	65 ± 0
0.25	50	1	21998	18.1 ± 0.5	16.3 ± 0.0	83 ± 0.1	416 ± 1	709 ± 1	46 ± 0
0.25	50	2	21998	15.7 ± 0.5	16.3 ± 0.0	82 ± 0.1	410 ± 0	834 ± 1	48 ± 0
0.25	50	3	22002	11.9 ± 0.4	16.4 ± 0.0	82 ± 0.1	405 ± 0	962 ± 1	52 ± 0
0.25	50	4	21998	8.4 ± 0.4	16.5 ± 0.0	81 ± 0.1	402 ± 0	1095 ± 1	56 ± 0
0.5	0	1	21496	9.7 ± 0.4	28.0 ± 0.1	76 ± 0.2	252 ± 0	874 ± 1	65 ± 0
0.5	0	2	21496	6.9 ± 0.3	28.1 ± 0.0	75 ± 0.2	247 ± 0	1134 ± 1	68 ± 0
0.5	0	3	21497	4.6 ± 0.3	28.3 ± 0.1	73 ± 0.2	244 ± 0	1400 ± 1	71 ± 0
0.5	0	4	21501	2.1 ± 0.2	28.3 ± 0.0	71 ± 0.2	241 ± 0	1671 ± 2	75 ± 0
0.5	10	1	21497	6.3 ± 0.3	28.1 ± 0.0	78 ± 0.2	250 ± 0	834 ± 1	68 ± 0
0.5	10	2	21494	4.4 ± 0.3	28.2 ± 0.0	77 ± 0.2	246 ± 0	1086 ± 1	70 ± 0
0.5	10	3	21499	2.7 ± 0.2	28.3 ± 0.0	75 ± 0.2	243 ± 0	1345 ± 1	72 ± 0
0.5	10	4	21497	1.2 ± 0.1	28.3 ± 0.0	74 ± 0.2	241 ± 0	1608 ± 1	75 ± 0
0.5	20	1	21497	3.9 ± 0.3	28.2 ± 0.0	80 ± 0.2	248 ± 0	795 ± 1	69 ± 0
0.5	20	2	21498	2.6 ± 0.2	28.2 ± 0.0	79 ± 0.2	244 ± 0	1041 ± 1	70 ± 0
0.5	20	3	21499	1.7 ± 0.2	28.3 ± 0.0	77 ± 0.2	242 ± 0	1293 ± 1	72 ± 0
0.5	20	4	21498	0.9 ± 0.1	28.4 ± 0.0	76 ± 0.2	239 ± 0	1548 ± 1	74 ± 0
0.5	30	1	21500	3.1 ± 0.2	28.2 ± 0.0	82 ± 0.2	246 ± 0	756 ± 1	66 ± 0
0.5	30	2	21499	2.4 ± 0.2	28.2 ± 0.0	81 ± 0.2	243 ± 0	997 ± 1	68 ± 0
0.5	30	3	21498	1.7 ± 0.2	28.3 ± 0.0	80 ± 0.2	240 ± 0	1245 ± 1	69 ± 0
0.5	30	4	21496	1.3 ± 0.1	28.3 ± 0.0	79 ± 0.2	238 ± 0	1496 ± 1	71 ± 0
0.5	40	1	21498	6.5 ± 0.3	28.1 ± 0.0	84 ± 0.2	245 ± 0	720 ± 1	58 ± 0
0.5	40	2	21499	5.3 ± 0.3	28.2 ± 0.0	83 ± 0.2	242 ± 0	959 ± 1	61 ± 0
0.5	40	3	21499	3.9 ± 0.3	28.2 ± 0.0	82 ± 0.2	239 ± 0	1203 ± 1	63 ± 0
0.5	40	4	21497	3.0 ± 0.2	28.3 ± 0.0	81 ± 0.1	237 ± 0	1453 ± 1	65 ± 0
0.5	50	1	21999	17.0 ± 0.5	27.7 ± 0.0	85 ± 0.1	244 ± 0	686 ± 1	46 ± 0
0.5	50	2	21997	14.0 ± 0.5	27.9 ± 0.0	85 ± 0.1	241 ± 0	924 ± 1	49 ± 0
0.5	50	3	22000	10.8 ± 0.4	28.0 ± 0.0	84 ± 0.1	238 ± 0	1168 ± 1	52 ± 0
0.5	50	4	22002	7.9 ± 0.4	28.1 ± 0.0	83 ± 0.1	236 ± 0	1416 ± 1	57 ± 0
1.0	0	1	21495	9.8 ± 0.4	51.1 ± 0.0	77 ± 0.2	138 ± 0	871 ± 1	66 ± 0
1.0	0	2	21493	7.2 ± 0.3	51.3 ± 0.0	75 ± 0.2	135 ± 0	1361 ± 2	68 ± 0
1.0	0	3	21495	5.0 ± 0.3	51.5 ± 0.0	73 ± 0.2	134 ± 0	1858 ± 2	71 ± 0
1.0	0	4	21493	2.4 ± 0.2	51.7 ± 0.0	72 ± 0.2	132 ± 0	2355 ± 2	74 ± 0
1.0	10	1	21499	6.4 ± 0.3	51.4 ± 0.2	78 ± 0.2	137 ± 0	830 ± 1	68 ± 0
1.0	10	2	21499	5.0 ± 0.3	51.5 ± 0.0	77 ± 0.2	135 ± 0	1310 ± 1	70 ± 0
1.0	10	3	21500	3.0 ± 0.2	51.6 ± 0.0	75 ± 0.2	133 ± 0	1801 ± 2	72 ± 0

P_{src} (sec)	PDS (%)	Grp Size	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	10	4	21500	1.5 ± 0.2	51.8 ± 0.0	74 ± 0.2	132 ± 0	2288 ± 2	75 ± 0
1.0	20	1	21499	4.0 ± 0.3	51.4 ± 0.0	80 ± 0.2	136 ± 0	789 ± 1	69 ± 0
1.0	20	2	21497	2.8 ± 0.2	51.6 ± 0.0	79 ± 0.2	134 ± 0	1265 ± 1	70 ± 0
1.0	20	3	21499	1.7 ± 0.2	51.7 ± 0.0	78 ± 0.2	132 ± 0	1746 ± 1	72 ± 0
1.0	20	4	21499	1.0 ± 0.1	51.8 ± 0.0	77 ± 0.2	131 ± 0	2227 ± 2	74 ± 0
1.0	30	1	21498	3.1 ± 0.2	51.5 ± 0.0	82 ± 0.2	135 ± 0	750 ± 1	66 ± 0
1.0	30	2	21498	2.4 ± 0.2	51.6 ± 0.0	81 ± 0.2	133 ± 0	1220 ± 1	67 ± 0
1.0	30	3	21496	1.7 ± 0.2	51.7 ± 0.0	80 ± 0.2	132 ± 0	1695 ± 1	69 ± 0
1.0	30	4	21498	1.3 ± 0.1	51.8 ± 0.0	79 ± 0.2	131 ± 0	2170 ± 2	71 ± 0
1.0	40	1	21498	6.9 ± 0.3	51.4 ± 0.2	84 ± 0.2	134 ± 0	712 ± 1	58 ± 0
1.0	40	2	21497	5.6 ± 0.3	51.4 ± 0.0	83 ± 0.2	132 ± 0	1178 ± 1	60 ± 0
1.0	40	3	21499	4.3 ± 0.3	51.5 ± 0.0	82 ± 0.2	131 ± 0	1651 ± 1	62 ± 0
1.0	40	4	21497	3.2 ± 0.2	51.6 ± 0.0	81 ± 0.2	130 ± 0	2122 ± 2	65 ± 0
1.0	50	1	22001	16.7 ± 0.5	50.6 ± 0.0	86 ± 0.1	134 ± 0	678 ± 1	47 ± 0
1.0	50	2	21998	14.2 ± 0.5	50.8 ± 0.0	85 ± 0.1	132 ± 0	1143 ± 1	49 ± 0
1.0	50	3	21999	11.4 ± 0.4	51.1 ± 0.0	84 ± 0.1	130 ± 0	1613 ± 1	52 ± 0
1.0	50	4	22000	8.3 ± 0.4	51.3 ± 0.0	84 ± 0.1	129 ± 0	2081 ± 1	55 ± 0
2.0	0	1	21496	10.1 ± 0.4	97.9 ± 0.1	78 ± 0.3	72 ± 0	869 ± 1	65 ± 0
2.0	0	2	21495	7.1 ± 0.3	98.5 ± 0.0	78 ± 0.2	71 ± 0	1818 ± 2	68 ± 0
2.0	0	3	21496	5.4 ± 0.3	98.8 ± 0.0	77 ± 0.2	71 ± 0	2773 ± 2	70 ± 0
2.0	0	4	21496	2.7 ± 0.2	99.2 ± 0.0	77 ± 0.2	71 ± 0	3731 ± 3	74 ± 0
2.0	10	1	21497	6.4 ± 0.3	98.4 ± 0.1	80 ± 0.2	72 ± 0	827 ± 1	68 ± 0
2.0	10	2	21501	4.8 ± 0.3	98.8 ± 0.0	80 ± 0.2	71 ± 0	1766 ± 2	70 ± 0
2.0	10	3	21497	3.1 ± 0.2	99.1 ± 0.0	79 ± 0.2	71 ± 0	2715 ± 2	72 ± 0
2.0	10	4	21498	1.7 ± 0.2	99.3 ± 0.0	79 ± 0.2	71 ± 0	3662 ± 3	75 ± 0
2.0	20	1	21502	3.9 ± 0.3	98.7 ± 0.0	81 ± 0.2	71 ± 0	788 ± 1	69 ± 0
2.0	20	2	21495	2.7 ± 0.2	99.0 ± 0.0	81 ± 0.2	70 ± 0	1720 ± 2	71 ± 0
2.0	20	3	21497	1.6 ± 0.2	99.2 ± 0.0	81 ± 0.2	70 ± 0	2656 ± 2	73 ± 0
2.0	20	4	21497	1.1 ± 0.1	99.4 ± 0.0	81 ± 0.2	70 ± 0	3596 ± 3	75 ± 0
2.0	30	1	21499	3.2 ± 0.2	98.8 ± 0.0	83 ± 0.2	71 ± 0	748 ± 1	66 ± 0
2.0	30	2	21500	2.3 ± 0.2	99.1 ± 0.1	83 ± 0.2	70 ± 0	1674 ± 2	68 ± 0
2.0	30	3	21498	1.8 ± 0.2	99.2 ± 0.0	83 ± 0.2	70 ± 0	2603 ± 2	70 ± 0
2.0	30	4	21495	1.3 ± 0.2	99.4 ± 0.2	83 ± 0.2	69 ± 0	3536 ± 3	72 ± 0
2.0	40	1	21496	6.6 ± 0.3	98.4 ± 0.0	85 ± 0.2	70 ± 0	711 ± 1	58 ± 0
2.0	40	2	21500	5.7 ± 0.3	98.7 ± 0.0	85 ± 0.2	70 ± 0	1632 ± 1	60 ± 0
2.0	40	3	21498	4.4 ± 0.3	98.9 ± 0.0	85 ± 0.2	69 ± 0	2557 ± 2	63 ± 0
2.0	40	4	21498	3.5 ± 0.2	99.1 ± 0.0	85 ± 0.2	69 ± 0	3481 ± 3	66 ± 0
2.0	50	1	21998	16.9 ± 0.5	96.9 ± 0.1	86 ± 0.1	70 ± 0	677 ± 1	46 ± 0
2.0	50	2	22001	14.9 ± 0.5	97.4 ± 0.1	86 ± 0.1	69 ± 0	1592 ± 1	49 ± 0
2.0	50	3	22000	11.7 ± 0.4	98.0 ± 0.1	86 ± 0.1	69 ± 0	2511 ± 2	52 ± 0
2.0	50	4	21998	9.2 ± 0.4	98.4 ± 0.1	86 ± 0.1	69 ± 0	3429 ± 2	56 ± 0

E.2 Cooja

E.2.1 Protectionless

Table E.21: Confidence Intervals for Protectionless on Cooja Network Size 7

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	5002	100.0 ± 0.0	2.1 ± 0.0	85 ± 0.3	113 ± 0	90 ± 0	0 ± 0
0.5	5002	100.0 ± 0.0	3.5 ± 0.0	87 ± 0.3	68 ± 0	89 ± 0	0 ± 0
1.0	5002	100.0 ± 0.0	6.4 ± 0.0	87 ± 0.3	37 ± 0	88 ± 0	0 ± 0
2.0	5002	100.0 ± 0.0	12.2 ± 0.0	100 ± 0.0	19 ± 0	96 ± 0	0 ± 0
4.0	502	100.0 ± 0.0	23.9 ± 0.0	100 ± 0.2	10 ± 0	97 ± 0	0 ± 0
8.0	502	100.0 ± 0.0	47.4 ± 0.0	99 ± 0.3	5 ± 0	97 ± 0	0 ± 0

Table E.22: Confidence Intervals for Protectionless on Cooja Network Size 9

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	502	100.0 ± 0.0	2.6 ± 0.0	92 ± 0.9	214 ± 1	122 ± 0	0 ± 0
0.5	502	100.0 ± 0.0	4.5 ± 0.0	98 ± 0.5	124 ± 1	121 ± 0	0 ± 0
1.0	502	100.0 ± 0.0	8.4 ± 0.0	99 ± 0.2	67 ± 0	120 ± 0	0 ± 0
2.0	502	100.0 ± 0.0	16.2 ± 0.0	100 ± 0.1	34 ± 0	121 ± 0	0 ± 0
4.0	502	100.0 ± 0.0	31.8 ± 0.1	100 ± 0.0	17 ± 0	123 ± 0	0 ± 0
8.0	502	100.0 ± 0.0	63.3 ± 0.2	100 ± 0.0	9 ± 0	124 ± 0	0 ± 0

Table E.23: Confidence Intervals for Protectionless on Cooja Network Size 11

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	502	100.0 ± 0.0	3.1 ± 0.0	95 ± 0.6	342 ± 2	151 ± 0	0 ± 0
0.5	502	100.0 ± 0.0	5.5 ± 0.0	98 ± 0.4	195 ± 1	151 ± 0	0 ± 0
1.0	502	100.0 ± 0.0	10.3 ± 0.0	100 ± 0.1	104 ± 0	151 ± 0	0 ± 0
2.0	502	100.0 ± 0.0	20.1 ± 0.0	99 ± 0.2	53 ± 0	153 ± 1	0 ± 0
4.0	502	100.0 ± 0.0	39.6 ± 0.0	99 ± 0.3	27 ± 0	157 ± 3	0 ± 0
8.0	502	100.0 ± 0.0	78.6 ± 0.0	99 ± 0.3	14 ± 0	154 ± 0	0 ± 0

E.2.2 DynamicSPR

Table E.24: Confidence Intervals for DynamicSPR on Cooja Network Size 7

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	502	1.2 ± 1.0	7.3 ± 0.0	84 ± 0.9	210 ± 1	126 ± 2	30 ± 1
0.25	Fxd2	502	0.6 ± 0.7	7.4 ± 0.0	78 ± 1.1	244 ± 1	135 ± 2	34 ± 1

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	Fxd1	502	0.0 ± 0.0	9.9 ± 0.0	92 ± 0.7	171 ± 1	109 ± 1	32 ± 1
0.5	Fxd2	502	0.0 ± 0.0	9.9 ± 0.0	90 ± 0.8	211 ± 1	123 ± 2	35 ± 1
1.0	Fxd1	502	0.0 ± 0.0	15.1 ± 0.0	98 ± 0.4	113 ± 1	105 ± 1	33 ± 1
1.0	Fxd2	502	0.0 ± 0.0	15.1 ± 0.0	93 ± 0.7	143 ± 1	113 ± 2	36 ± 0
2.0	Fxd1	502	0.0 ± 0.0	25.9 ± 0.0	98 ± 0.4	67 ± 0	106 ± 2	33 ± 1
2.0	Fxd2	502	0.0 ± 0.0	25.9 ± 0.0	97 ± 0.5	85 ± 1	112 ± 3	36 ± 0
4.0	Fxd1	502	0.0 ± 0.0	47.4 ± 0.0	98 ± 0.3	37 ± 0	111 ± 4	34 ± 0
4.0	Fxd2	502	0.0 ± 0.0	47.4 ± 0.0	98 ± 0.4	47 ± 0	114 ± 4	36 ± 0
8.0	Fxd1	502	0.0 ± 0.0	85.8 ± 0.2	98 ± 0.4	20 ± 0	136 ± 12	33 ± 0
8.0	Fxd2	502	0.0 ± 0.0	86.0 ± 0.2	97 ± 0.4	25 ± 0	131 ± 9	35 ± 0

Table E.25: Confidence Intervals for DynamicSPR on Cooja Network Size 9

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	502	0.6 ± 0.7	8.4 ± 0.0	90 ± 0.7	381 ± 2	162 ± 1	39 ± 1
0.25	Fxd2	502	0.0 ± 0.0	8.3 ± 0.0	82 ± 0.9	442 ± 3	178 ± 2	43 ± 1
0.5	Fxd1	502	0.0 ± 0.0	12.0 ± 0.0	93 ± 0.6	294 ± 2	152 ± 1	42 ± 1
0.5	Fxd2	502	0.0 ± 0.0	12.0 ± 0.0	92 ± 0.6	364 ± 3	159 ± 2	44 ± 1
1.0	Fxd1	502	0.0 ± 0.0	19.1 ± 0.0	91 ± 0.3	191 ± 1	143 ± 1	43 ± 1
1.0	Fxd2	502	0.0 ± 0.0	19.1 ± 0.0	89 ± 0.4	233 ± 2	152 ± 2	45 ± 1
2.0	Fxd1	502	0.0 ± 0.0	33.7 ± 0.0	98 ± 0.3	110 ± 1	144 ± 2	44 ± 1
2.0	Fxd2	502	0.0 ± 0.0	33.8 ± 0.0	97 ± 0.4	137 ± 1	146 ± 2	45 ± 1
4.0	Fxd1	502	0.0 ± 0.0	63.1 ± 0.0	97 ± 0.4	60 ± 0	149 ± 3	44 ± 1
4.0	Fxd2	502	0.0 ± 0.0	63.1 ± 0.0	96 ± 0.4	75 ± 1	149 ± 4	45 ± 1
8.0	Fxd1	502	0.0 ± 0.0	118.0 ± 0.2	98 ± 0.3	32 ± 0	154 ± 6	46 ± 1
8.0	Fxd2	502	0.0 ± 0.0	117.7 ± 0.2	97 ± 0.4	42 ± 0	154 ± 6	49 ± 0

Table E.26: Confidence Intervals for DynamicSPR on Cooja Network Size 11

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	399	0.3 ± 0.5	9.3 ± 0.0	87 ± 0.8	640 ± 3	207 ± 2	54 ± 1
0.25	Fxd2	380	0.0 ± 0.0	9.3 ± 0.0	80 ± 0.9	736 ± 3	223 ± 2	60 ± 1
0.5	Fxd1	397	0.0 ± 0.0	13.9 ± 0.0	93 ± 0.6	480 ± 2	194 ± 1	57 ± 1
0.5	Fxd2	365	0.0 ± 0.0	13.9 ± 0.0	91 ± 0.6	603 ± 3	206 ± 2	60 ± 1
1.0	Fxd1	402	0.0 ± 0.0	23.0 ± 0.0	97 ± 0.4	300 ± 1	183 ± 2	57 ± 1
1.0	Fxd2	368	0.0 ± 0.0	23.0 ± 0.0	95 ± 0.5	384 ± 2	192 ± 2	60 ± 1
2.0	Fxd1	431	0.0 ± 0.0	41.5 ± 0.0	98 ± 0.3	169 ± 1	179 ± 2	58 ± 1
2.0	Fxd2	392	0.0 ± 0.0	41.5 ± 0.0	97 ± 0.4	221 ± 1	185 ± 3	61 ± 0
4.0	Fxd1	388	0.0 ± 0.0	78.6 ± 0.0	98 ± 0.3	90 ± 0	184 ± 4	59 ± 1
4.0	Fxd2	360	0.0 ± 0.0	78.6 ± 0.0	98 ± 0.3	118 ± 1	185 ± 4	61 ± 0
8.0	Fxd1	390	0.0 ± 0.0	148.3 ± 0.2	98 ± 0.3	45 ± 0	196 ± 8	55 ± 1
8.0	Fxd2	369	0.0 ± 0.0	148.5 ± 0.2	97 ± 0.4	54 ± 1	199 ± 8	55 ± 1

E.2.3 DynamicSPR with Duty Cycling

Table E.27: Confidence Intervals for DynamicSPR with Duty Cycling on Cooja Network Size 7

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	5	502	40.0 ± 4.3	6.8 ± 0.1	79 ± 1.3	141 ± 2	152 ± 2	14 ± 1
0.25	Fxd1	4	502	29.9 ± 4.0	7.0 ± 0.1	85 ± 1.2	152 ± 2	144 ± 2	16 ± 1
0.25	Fxd1	3	502	25.1 ± 3.8	7.0 ± 0.1	84 ± 1.1	158 ± 2	145 ± 2	19 ± 1
0.25	Fxd1	1	494	5.7 ± 2.0	7.2 ± 0.0	17 ± 1.1	80 ± 1	137 ± 2	19 ± 1
0.25	Fxd2	5	502	4.0 ± 1.7	7.3 ± 0.0	72 ± 1.3	194 ± 2	152 ± 2	26 ± 1
0.25	Fxd2	4	502	23.3 ± 3.7	7.1 ± 0.0	73 ± 1.4	140 ± 1	154 ± 2	18 ± 1
0.25	Fxd2	3	502	22.5 ± 3.7	7.1 ± 0.0	72 ± 1.4	155 ± 2	155 ± 2	19 ± 1
0.25	Fxd2	1	412	0.0 ± 0.0	7.3 ± 0.0	12 ± 0.4	85 ± 1	138 ± 3	22 ± 1
0.5	Fxd1	5	502	12.4 ± 2.9	9.7 ± 0.0	91 ± 0.8	127 ± 1	149 ± 2	14 ± 1
0.5	Fxd1	4	502	10.8 ± 2.7	9.7 ± 0.1	88 ± 1.0	127 ± 1	150 ± 2	19 ± 1
0.5	Fxd1	3	502	10.6 ± 2.7	9.7 ± 0.1	86 ± 1.1	129 ± 1	144 ± 2	21 ± 1
0.5	Fxd1	2	502	10.2 ± 2.7	9.7 ± 0.1	89 ± 0.9	132 ± 1	135 ± 1	20 ± 1
0.5	Fxd1	1	502	3.0 ± 1.5	9.8 ± 0.0	87 ± 0.9	137 ± 1	137 ± 1	22 ± 1
0.5	Fxd2	5	502	5.6 ± 2.0	9.8 ± 0.1	82 ± 1.0	146 ± 1	146 ± 1	23 ± 1
0.5	Fxd2	4	502	1.6 ± 1.1	9.9 ± 0.0	87 ± 0.9	158 ± 1	139 ± 1	27 ± 1
0.5	Fxd2	3	502	1.2 ± 1.0	9.9 ± 0.0	88 ± 0.9	163 ± 1	139 ± 1	28 ± 1
0.5	Fxd2	2	502	0.4 ± 0.6	9.9 ± 0.0	87 ± 0.9	165 ± 1	138 ± 1	28 ± 1
0.5	Fxd2	1	502	16.5 ± 3.3	9.7 ± 0.1	89 ± 0.9	139 ± 1	145 ± 1	21 ± 1
1.0	Fxd1	5	502	12.2 ± 2.9	14.9 ± 0.1	95 ± 0.6	82 ± 0	152 ± 2	14 ± 1
1.0	Fxd1	4	502	9.8 ± 2.6	14.8 ± 0.1	96 ± 0.6	85 ± 0	148 ± 2	19 ± 1
1.0	Fxd1	3	502	8.8 ± 2.5	14.8 ± 0.1	95 ± 0.6	86 ± 0	141 ± 1	19 ± 1
1.0	Fxd1	2	502	12.0 ± 2.8	14.7 ± 0.1	96 ± 0.5	87 ± 0	134 ± 1	19 ± 1
1.0	Fxd1	1	502	4.6 ± 1.8	15.0 ± 0.1	96 ± 0.5	91 ± 0	134 ± 1	22 ± 1
1.0	Fxd2	5	502	3.2 ± 1.5	15.0 ± 0.1	93 ± 0.8	102 ± 1	151 ± 2	22 ± 1
1.0	Fxd2	4	502	2.8 ± 1.4	15.0 ± 0.1	91 ± 0.9	102 ± 1	149 ± 2	26 ± 1
1.0	Fxd2	3	502	2.2 ± 1.3	15.1 ± 0.1	91 ± 0.9	107 ± 1	146 ± 2	27 ± 1
1.0	Fxd2	2	502	2.8 ± 1.4	15.1 ± 0.1	92 ± 0.8	107 ± 1	138 ± 1	27 ± 1
1.0	Fxd2	1	502	1.4 ± 1.0	15.1 ± 0.0	93 ± 0.7	116 ± 1	138 ± 1	27 ± 1
2.0	Fxd1	5	502	21.9 ± 3.6	24.6 ± 0.2	95 ± 0.7	49 ± 0	149 ± 2	12 ± 1
2.0	Fxd1	4	502	18.5 ± 3.4	24.7 ± 0.3	97 ± 0.5	49 ± 0	139 ± 1	16 ± 1
2.0	Fxd1	3	502	8.2 ± 2.4	25.3 ± 0.2	97 ± 0.6	50 ± 0	143 ± 2	19 ± 1
2.0	Fxd1	2	502	15.5 ± 3.2	24.8 ± 0.3	96 ± 0.5	51 ± 0	134 ± 1	17 ± 1
2.0	Fxd1	1	502	4.6 ± 1.8	25.7 ± 0.1	96 ± 0.5	53 ± 0	135 ± 1	21 ± 1
2.0	Fxd2	5	502	1.0 ± 0.9	25.8 ± 0.1	95 ± 0.7	61 ± 0	153 ± 3	22 ± 1
2.0	Fxd2	4	502	2.0 ± 1.2	25.7 ± 0.1	96 ± 0.6	63 ± 0	147 ± 3	24 ± 1
2.0	Fxd2	3	502	3.0 ± 1.5	25.7 ± 0.1	97 ± 0.5	63 ± 1	141 ± 2	25 ± 1
2.0	Fxd2	2	502	3.6 ± 1.6	25.7 ± 0.1	97 ± 0.5	65 ± 1	139 ± 2	25 ± 1
2.0	Fxd2	1	502	2.6 ± 1.4	25.7 ± 0.1	97 ± 0.5	67 ± 1	139 ± 2	26 ± 1
4.0	Fxd1	5	502	9.8 ± 2.6	46.0 ± 0.4	94 ± 0.6	27 ± 0	152 ± 3	15 ± 1
4.0	Fxd1	4	502	22.3 ± 3.7	44.8 ± 0.5	97 ± 0.5	27 ± 0	139 ± 3	13 ± 1

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
4.0	Fxd1	3	502	22.7 ± 3.7	44.6 ± 0.6	97 ± 0.6	28 ± 0	140 ± 3	15 ± 1
4.0	Fxd1	2	502	22.3 ± 3.7	43.9 ± 0.6	96 ± 0.6	28 ± 0	134 ± 1	14 ± 1
4.0	Fxd1	1	502	5.2 ± 1.9	46.7 ± 0.3	96 ± 0.6	29 ± 0	140 ± 4	19 ± 1
4.0	Fxd2	5	502	6.6 ± 2.2	46.4 ± 0.4	97 ± 0.5	35 ± 0	145 ± 2	19 ± 1
4.0	Fxd2	4	502	3.8 ± 1.7	46.8 ± 0.3	97 ± 0.5	35 ± 0	137 ± 1	22 ± 1
4.0	Fxd2	3	502	4.6 ± 1.8	46.8 ± 0.3	97 ± 0.5	36 ± 0	139 ± 2	22 ± 1
4.0	Fxd2	2	502	3.4 ± 1.6	46.9 ± 0.3	97 ± 0.5	36 ± 0	137 ± 2	24 ± 1
4.0	Fxd2	1	502	1.0 ± 0.9	47.2 ± 0.2	98 ± 0.4	36 ± 0	138 ± 3	25 ± 1
8.0	Fxd1	5	502	3.6 ± 1.6	86.0 ± 0.4	97 ± 0.5	15 ± 0	135 ± 2	19 ± 1
8.0	Fxd1	4	502	1.8 ± 1.2	86.4 ± 0.3	97 ± 0.5	15 ± 0	139 ± 4	23 ± 1
8.0	Fxd1	3	502	1.2 ± 1.0	87.0 ± 0.2	97 ± 0.4	16 ± 0	140 ± 5	25 ± 1
8.0	Fxd1	2	502	2.6 ± 1.4	86.8 ± 0.3	98 ± 0.4	16 ± 0	136 ± 3	24 ± 1
8.0	Fxd1	1	502	0.0 ± 0.0	87.0 ± 0.1	98 ± 0.4	17 ± 0	137 ± 3	26 ± 1
8.0	Fxd2	5	502	1.8 ± 1.2	87.5 ± 0.4	95 ± 0.6	19 ± 0	140 ± 3	25 ± 1
8.0	Fxd2	4	502	0.6 ± 0.7	88.0 ± 0.2	97 ± 0.4	19 ± 0	137 ± 2	30 ± 1
8.0	Fxd2	3	502	0.0 ± 0.0	88.3 ± 0.0	97 ± 0.5	20 ± 0	139 ± 3	31 ± 1
8.0	Fxd2	2	502	0.4 ± 0.6	88.2 ± 0.2	97 ± 0.4	20 ± 0	137 ± 2	31 ± 1
8.0	Fxd2	1	502	0.0 ± 0.0	88.4 ± 0.0	98 ± 0.4	21 ± 0	138 ± 4	32 ± 1

Table E.28: Confidence Intervals for DynamicSPR with Duty Cycling on Cooja Network Size 9

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	5	502	16.3 ± 3.2	8.2 ± 0.0	84 ± 1.0	284 ± 2	187 ± 1	29 ± 2
0.25	Fxd1	4	502	7.6 ± 2.3	8.3 ± 0.0	84 ± 0.9	305 ± 2	192 ± 1	33 ± 1
0.25	Fxd1	3	502	2.8 ± 1.4	8.3 ± 0.0	83 ± 0.9	328 ± 2	198 ± 1	36 ± 1
0.25	Fxd1	1	451	0.2 ± 0.4	8.3 ± 0.0	8 ± 0.2	109 ± 1	180 ± 3	29 ± 0
0.25	Fxd2	5	502	0.0 ± 0.0	8.3 ± 0.0	71 ± 1.0	372 ± 2	207 ± 2	46 ± 1
0.25	Fxd2	4	502	35.3 ± 4.2	7.9 ± 0.1	84 ± 1.0	238 ± 2	189 ± 1	21 ± 2
0.25	Fxd2	3	502	31.3 ± 4.1	8.0 ± 0.1	83 ± 1.0	259 ± 3	190 ± 1	26 ± 2
0.25	Fxd2	1	344	0.3 ± 0.6	8.4 ± 0.0	7 ± 0.5	112 ± 1	213 ± 7	31 ± 1
0.5	Fxd1	5	502	5.6 ± 2.0	11.9 ± 0.0	84 ± 1.0	212 ± 1	176 ± 1	29 ± 1
0.5	Fxd1	4	502	7.0 ± 2.2	11.9 ± 0.1	88 ± 0.8	223 ± 2	181 ± 1	34 ± 1
0.5	Fxd1	3	502	1.8 ± 1.2	11.9 ± 0.0	85 ± 0.8	233 ± 2	184 ± 1	41 ± 1
0.5	Fxd1	2	502	3.0 ± 1.5	11.9 ± 0.0	86 ± 0.8	240 ± 1	185 ± 1	38 ± 1
0.5	Fxd1	1	470	0.2 ± 0.4	12.0 ± 0.0	88 ± 0.7	257 ± 1	193 ± 2	41 ± 1
0.5	Fxd2	5	502	1.0 ± 0.9	12.0 ± 0.0	89 ± 0.8	259 ± 2	185 ± 1	43 ± 1
0.5	Fxd2	4	502	0.0 ± 0.0	12.0 ± 0.0	86 ± 0.8	297 ± 1	188 ± 1	47 ± 1
0.5	Fxd2	3	502	0.0 ± 0.0	12.0 ± 0.0	86 ± 0.8	312 ± 2	192 ± 1	47 ± 1
0.5	Fxd2	2	502	0.0 ± 0.0	12.0 ± 0.0	86 ± 0.8	315 ± 2	193 ± 1	47 ± 1
0.5	Fxd2	1	315	2.9 ± 1.8	11.9 ± 0.0	89 ± 0.9	257 ± 3	197 ± 2	41 ± 1
1.0	Fxd1	5	502	5.2 ± 1.9	18.9 ± 0.1	90 ± 0.5	141 ± 1	177 ± 1	27 ± 1
1.0	Fxd1	4	502	2.8 ± 1.4	19.1 ± 0.1	90 ± 0.4	145 ± 1	177 ± 1	32 ± 1
1.0	Fxd1	3	502	1.8 ± 1.2	19.1 ± 0.0	90 ± 0.5	150 ± 1	178 ± 1	36 ± 1

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	Fxd1	2	502	2.8 ± 1.4	19.1 ± 0.0	90 ± 0.4	152 ± 1	178 ± 1	35 ± 1
1.0	Fxd1	1	502	0.0 ± 0.0	19.1 ± 0.0	90 ± 0.4	166 ± 1	178 ± 1	40 ± 1
1.0	Fxd2	5	502	1.8 ± 1.2	19.1 ± 0.0	81 ± 0.9	163 ± 1	180 ± 1	37 ± 1
1.0	Fxd2	4	502	0.8 ± 0.8	19.1 ± 0.0	84 ± 0.7	175 ± 1	183 ± 1	46 ± 1
1.0	Fxd2	3	502	0.0 ± 0.0	19.2 ± 0.0	83 ± 0.7	183 ± 1	187 ± 2	48 ± 1
1.0	Fxd2	2	502	0.0 ± 0.0	19.1 ± 0.0	84 ± 0.7	189 ± 1	187 ± 2	47 ± 1
1.0	Fxd2	1	502	0.0 ± 0.0	19.1 ± 0.0	85 ± 0.6	210 ± 1	189 ± 2	47 ± 1
2.0	Fxd1	5	502	4.2 ± 1.8	33.5 ± 0.1	97 ± 0.4	80 ± 0	178 ± 1	27 ± 1
2.0	Fxd1	4	502	3.0 ± 1.5	33.6 ± 0.1	97 ± 0.4	81 ± 1	178 ± 1	32 ± 1
2.0	Fxd1	3	502	4.4 ± 1.8	33.5 ± 0.1	97 ± 0.4	83 ± 1	179 ± 2	32 ± 1
2.0	Fxd1	2	502	2.8 ± 1.4	33.7 ± 0.1	97 ± 0.4	85 ± 1	179 ± 1	33 ± 1
2.0	Fxd1	1	502	0.0 ± 0.0	33.8 ± 0.0	97 ± 0.4	91 ± 0	180 ± 2	37 ± 1
2.0	Fxd2	5	502	1.2 ± 1.0	33.7 ± 0.1	97 ± 0.4	102 ± 1	178 ± 1	35 ± 1
2.0	Fxd2	4	502	0.4 ± 0.6	33.7 ± 0.1	97 ± 0.4	103 ± 1	180 ± 2	41 ± 1
2.0	Fxd2	3	502	0.4 ± 0.6	33.7 ± 0.0	97 ± 0.4	107 ± 1	179 ± 1	43 ± 1
2.0	Fxd2	2	502	0.2 ± 0.4	33.7 ± 0.0	97 ± 0.4	109 ± 1	180 ± 2	43 ± 1
2.0	Fxd2	1	502	0.2 ± 0.4	33.7 ± 0.0	97 ± 0.4	116 ± 1	177 ± 1	45 ± 1
4.0	Fxd1	5	502	3.2 ± 1.5	62.8 ± 0.2	91 ± 0.4	44 ± 0	179 ± 2	26 ± 1
4.0	Fxd1	4	502	4.8 ± 1.9	62.8 ± 0.2	91 ± 0.4	43 ± 0	179 ± 1	31 ± 1
4.0	Fxd1	3	502	2.8 ± 1.4	62.9 ± 0.1	91 ± 0.4	45 ± 0	183 ± 3	32 ± 1
4.0	Fxd1	2	502	4.0 ± 1.7	62.8 ± 0.2	92 ± 0.3	46 ± 0	179 ± 2	32 ± 1
4.0	Fxd1	1	502	1.0 ± 0.9	63.1 ± 0.1	91 ± 0.3	49 ± 0	179 ± 1	36 ± 1
4.0	Fxd2	5	502	1.6 ± 1.1	62.9 ± 0.2	92 ± 0.3	56 ± 1	180 ± 4	35 ± 1
4.0	Fxd2	4	502	0.6 ± 0.7	63.1 ± 0.1	91 ± 0.3	57 ± 0	180 ± 2	41 ± 1
4.0	Fxd2	3	502	0.4 ± 0.6	63.1 ± 0.1	91 ± 0.3	60 ± 0	180 ± 2	44 ± 1
4.0	Fxd2	2	502	1.2 ± 1.0	63.0 ± 0.1	92 ± 0.3	61 ± 1	178 ± 1	42 ± 1
4.0	Fxd2	1	502	0.0 ± 0.0	63.2 ± 0.0	91 ± 0.3	65 ± 1	181 ± 2	44 ± 1
8.0	Fxd1	5	502	2.8 ± 1.4	118.6 ± 0.5	98 ± 0.4	22 ± 0	184 ± 3	27 ± 1
8.0	Fxd1	4	502	4.6 ± 1.8	118.4 ± 0.6	98 ± 0.3	22 ± 0	186 ± 4	29 ± 1
8.0	Fxd1	3	502	3.2 ± 1.5	118.6 ± 0.6	98 ± 0.3	23 ± 0	186 ± 4	29 ± 1
8.0	Fxd1	2	502	3.0 ± 1.5	118.5 ± 0.5	98 ± 0.3	24 ± 0	181 ± 2	28 ± 1
8.0	Fxd1	1	502	1.4 ± 1.0	119.4 ± 0.3	98 ± 0.3	25 ± 0	184 ± 4	32 ± 1
8.0	Fxd2	5	502	1.0 ± 0.9	119.7 ± 0.3	98 ± 0.3	29 ± 0	185 ± 3	33 ± 1
8.0	Fxd2	4	502	1.4 ± 1.0	119.7 ± 0.4	98 ± 0.3	29 ± 0	187 ± 5	37 ± 1
8.0	Fxd2	3	502	0.8 ± 0.8	120.0 ± 0.2	98 ± 0.3	30 ± 0	186 ± 4	39 ± 1
8.0	Fxd2	2	502	0.4 ± 0.6	120.1 ± 0.2	98 ± 0.3	30 ± 0	186 ± 5	38 ± 1
8.0	Fxd2	1	502	0.2 ± 0.4	120.3 ± 0.1	98 ± 0.3	31 ± 0	185 ± 4	40 ± 1

Table E.29: Confidence Intervals for DynamicSPR with Duty Cycling on Cooja Network Size 11

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	5	430	10.9 ± 3.0	9.1 ± 0.0	78 ± 1.0	440 ± 4	234 ± 1	38 ± 2
0.25	Fxd1	4	313	1.6 ± 1.4	9.3 ± 0.0	79 ± 1.2	474 ± 3	241 ± 2	44 ± 2

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.25	Fxd1	3	255	0.4 ± 0.8	9.3 ± 0.0	79 ± 1.2	526 ± 4	250 ± 2	50 ± 2
0.25	Fxd1	1	172	0.0 ± 0.0	9.3 ± 0.0	6 ± 0.3	136 ± 2	229 ± 5	35 ± 1
0.25	Fxd2	5	379	0.0 ± 0.0	9.3 ± 0.0	71 ± 1.1	599 ± 3	261 ± 2	57 ± 1
0.25	Fxd2	4	226	36.3 ± 6.3	8.8 ± 0.1	88 ± 1.2	364 ± 5	232 ± 2	28 ± 3
0.25	Fxd2	3	204	27.5 ± 6.2	8.9 ± 0.1	86 ± 1.4	405 ± 8	237 ± 2	33 ± 3
0.25	Fxd2	1	153	0.0 ± 0.0	9.3 ± 0.0	6 ± 0.3	138 ± 2	252 ± 8	37 ± 1
0.5	Fxd1	5	455	11.2 ± 2.9	13.7 ± 0.1	87 ± 0.8	330 ± 3	228 ± 1	32 ± 2
0.5	Fxd1	4	412	1.9 ± 1.3	13.8 ± 0.0	86 ± 0.8	360 ± 3	230 ± 1	45 ± 2
0.5	Fxd1	3	375	2.1 ± 1.5	13.9 ± 0.0	87 ± 0.8	370 ± 3	232 ± 1	49 ± 2
0.5	Fxd1	2	340	2.9 ± 1.8	13.9 ± 0.0	89 ± 0.8	374 ± 3	233 ± 1	46 ± 2
0.5	Fxd1	1	155	0.0 ± 0.0	13.9 ± 0.0	88 ± 1.1	405 ± 4	239 ± 2	52 ± 2
0.5	Fxd2	5	459	0.4 ± 0.6	13.9 ± 0.0	78 ± 1.0	398 ± 3	237 ± 1	53 ± 1
0.5	Fxd2	4	392	0.0 ± 0.0	13.9 ± 0.0	80 ± 0.9	451 ± 3	241 ± 1	57 ± 1
0.5	Fxd2	3	414	0.0 ± 0.0	13.9 ± 0.0	82 ± 0.8	477 ± 4	244 ± 1	56 ± 1
0.5	Fxd2	2	372	0.0 ± 0.0	13.9 ± 0.0	82 ± 0.8	482 ± 3	244 ± 2	57 ± 1
0.5	Fxd2	1	127	3.9 ± 3.4	13.8 ± 0.1	84 ± 1.6	388 ± 9	246 ± 3	50 ± 3
1.0	Fxd1	5	460	8.0 ± 2.5	22.6 ± 0.1	96 ± 0.5	205 ± 1	226 ± 1	30 ± 1
1.0	Fxd1	4	454	5.3 ± 2.1	22.8 ± 0.1	96 ± 0.5	214 ± 2	227 ± 1	35 ± 1
1.0	Fxd1	3	450	1.6 ± 1.1	22.9 ± 0.1	95 ± 0.5	220 ± 2	225 ± 1	40 ± 1
1.0	Fxd1	2	408	3.9 ± 1.9	22.9 ± 0.1	96 ± 0.5	225 ± 2	224 ± 1	38 ± 1
1.0	Fxd1	1	424	0.0 ± 0.0	23.0 ± 0.0	94 ± 0.5	247 ± 1	227 ± 1	50 ± 1
1.0	Fxd2	5	478	1.7 ± 1.2	22.9 ± 0.1	92 ± 0.6	241 ± 2	228 ± 1	43 ± 1
1.0	Fxd2	4	427	0.0 ± 0.0	22.9 ± 0.0	92 ± 0.7	262 ± 2	231 ± 2	52 ± 1
1.0	Fxd2	3	417	0.0 ± 0.0	23.0 ± 0.0	93 ± 0.7	275 ± 2	231 ± 1	54 ± 1
1.0	Fxd2	2	412	0.0 ± 0.0	23.0 ± 0.0	92 ± 0.6	283 ± 2	233 ± 1	56 ± 1
1.0	Fxd2	1	391	0.0 ± 0.0	23.0 ± 0.0	93 ± 0.6	316 ± 3	235 ± 2	56 ± 1
2.0	Fxd1	5	455	6.6 ± 2.3	41.1 ± 0.2	97 ± 0.4	115 ± 1	228 ± 2	30 ± 1
2.0	Fxd1	4	494	5.1 ± 1.9	41.2 ± 0.1	97 ± 0.4	118 ± 1	229 ± 2	34 ± 1
2.0	Fxd1	3	450	3.3 ± 1.7	41.3 ± 0.1	97 ± 0.4	121 ± 1	229 ± 2	37 ± 1
2.0	Fxd1	2	406	3.0 ± 1.7	41.4 ± 0.1	98 ± 0.4	124 ± 1	225 ± 1	36 ± 1
2.0	Fxd1	1	437	0.9 ± 0.9	41.4 ± 0.1	98 ± 0.3	132 ± 1	227 ± 1	41 ± 1
2.0	Fxd2	5	469	1.7 ± 1.2	41.5 ± 0.1	97 ± 0.3	147 ± 2	227 ± 2	39 ± 1
2.0	Fxd2	4	425	0.9 ± 0.9	41.5 ± 0.1	97 ± 0.4	149 ± 2	229 ± 2	44 ± 1
2.0	Fxd2	3	423	0.2 ± 0.5	41.5 ± 0.0	96 ± 0.5	153 ± 2	229 ± 2	45 ± 1
2.0	Fxd2	2	418	0.2 ± 0.5	41.5 ± 0.0	96 ± 0.5	157 ± 2	226 ± 2	47 ± 1
2.0	Fxd2	1	413	0.2 ± 0.5	41.5 ± 0.0	95 ± 0.5	163 ± 1	225 ± 1	53 ± 1
4.0	Fxd1	5	449	4.5 ± 1.9	78.2 ± 0.3	98 ± 0.4	61 ± 0	232 ± 3	31 ± 1
4.0	Fxd1	4	445	5.4 ± 2.1	77.9 ± 0.3	98 ± 0.3	63 ± 0	228 ± 2	33 ± 1
4.0	Fxd1	3	444	3.8 ± 1.8	78.2 ± 0.3	98 ± 0.3	63 ± 0	228 ± 2	35 ± 1
4.0	Fxd1	2	403	5.7 ± 2.3	78.0 ± 0.3	98 ± 0.4	65 ± 0	226 ± 2	35 ± 1
4.0	Fxd1	1	473	1.1 ± 0.9	78.6 ± 0.1	98 ± 0.3	69 ± 0	227 ± 2	40 ± 1
4.0	Fxd2	5	418	1.4 ± 1.1	78.4 ± 0.2	98 ± 0.3	80 ± 1	228 ± 2	38 ± 1
4.0	Fxd2	4	456	0.9 ± 0.9	78.5 ± 0.2	98 ± 0.3	81 ± 1	231 ± 4	42 ± 1
4.0	Fxd2	3	413	0.0 ± 0.0	78.6 ± 0.0	98 ± 0.3	84 ± 1	231 ± 3	45 ± 1
4.0	Fxd2	2	412	1.0 ± 1.0	78.5 ± 0.2	98 ± 0.3	85 ± 1	225 ± 2	45 ± 1

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
4.0	Fxd2	1	408	0.2 ± 0.5	78.6 ± 0.0	98 ± 0.4	88 ± 1	227 ± 2	48 ± 1
8.0	Fxd1	5	435	1.6 ± 1.2	148.8 ± 0.4	97 ± 0.4	32 ± 0	232 ± 3	36 ± 1
8.0	Fxd1	4	436	1.8 ± 1.3	149.2 ± 0.4	97 ± 0.4	32 ± 0	232 ± 4	43 ± 1
8.0	Fxd1	3	471	1.1 ± 0.9	149.4 ± 0.2	98 ± 0.3	33 ± 0	235 ± 4	45 ± 1
8.0	Fxd1	2	392	1.8 ± 1.3	149.4 ± 0.2	98 ± 0.3	34 ± 0	233 ± 4	41 ± 1
8.0	Fxd1	1	415	0.2 ± 0.5	149.8 ± 0.2	98 ± 0.3	38 ± 0	234 ± 4	47 ± 1
8.0	Fxd2	5	417	0.2 ± 0.5	150.7 ± 0.2	97 ± 0.4	40 ± 1	231 ± 4	46 ± 1
8.0	Fxd2	4	407	0.0 ± 0.0	150.9 ± 0.0	97 ± 0.4	43 ± 0	232 ± 4	53 ± 1
8.0	Fxd2	3	399	0.0 ± 0.0	151.0 ± 0.0	98 ± 0.4	46 ± 0	234 ± 4	54 ± 1
8.0	Fxd2	2	361	0.0 ± 0.0	151.0 ± 0.0	98 ± 0.3	47 ± 0	232 ± 5	55 ± 1
8.0	Fxd2	1	385	0.0 ± 0.0	151.1 ± 0.0	98 ± 0.3	51 ± 0	232 ± 4	56 ± 1

E.2.4 DynamicSPR with TinyOS LPL

Table E.30: Confidence Intervals for DynamicSPR with TinyOS LPL on Cooja Network Size 7

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	Fxd1	1	501	22.8 ± 3.7	9.5 ± 0.1	64 ± 1.6	99 ± 1	369 ± 4	19 ± 1
0.5	Fxd1	2	502	6.6 ± 2.2	9.8 ± 0.1	66 ± 1.3	115 ± 1	286 ± 3	26 ± 1
0.5	Fxd1	3	501	37.3 ± 4.2	9.2 ± 0.1	62 ± 1.7	86 ± 1	438 ± 6	14 ± 1
0.5	Fxd1	4	501	41.7 ± 4.3	9.4 ± 0.1	61 ± 1.7	85 ± 1	478 ± 7	14 ± 1
0.5	Fxd1	5	502	14.5 ± 3.1	9.7 ± 0.1	61 ± 1.3	100 ± 1	341 ± 4	22 ± 1
0.5	Fxd2	1	498	4.4 ± 1.8	9.8 ± 0.1	45 ± 1.5	110 ± 1	379 ± 5	30 ± 1
0.5	Fxd2	2	502	0.8 ± 0.8	9.9 ± 0.0	55 ± 1.3	133 ± 1	279 ± 3	31 ± 1
0.5	Fxd2	3	500	18.2 ± 3.4	9.6 ± 0.1	48 ± 1.8	94 ± 1	448 ± 7	23 ± 1
0.5	Fxd2	4	496	16.3 ± 3.3	9.8 ± 0.1	48 ± 1.8	92 ± 1	489 ± 8	23 ± 1
0.5	Fxd2	5	502	2.6 ± 1.4	9.9 ± 0.0	50 ± 1.3	117 ± 1	335 ± 5	30 ± 1
0.5	RND	1	500	11.2 ± 2.8	9.7 ± 0.1	54 ± 1.6	106 ± 1	376 ± 4	26 ± 1
0.5	RND	2	502	1.6 ± 1.1	9.9 ± 0.0	59 ± 1.2	129 ± 1	280 ± 3	30 ± 1
0.5	RND	3	500	27.4 ± 3.9	9.4 ± 0.1	52 ± 1.9	91 ± 1	440 ± 6	18 ± 1
0.5	RND	4	500	28.4 ± 4.0	9.6 ± 0.1	53 ± 1.8	89 ± 1	475 ± 8	19 ± 1
0.5	RND	5	502	4.4 ± 1.8	9.8 ± 0.1	52 ± 1.4	111 ± 1	336 ± 4	28 ± 1
1.0	Fxd1	1	502	1.0 ± 0.9	15.1 ± 0.0	73 ± 1.4	84 ± 1	361 ± 4	30 ± 1
1.0	Fxd1	2	502	1.4 ± 1.0	15.1 ± 0.0	77 ± 1.2	87 ± 1	288 ± 3	31 ± 1
1.0	Fxd1	3	502	3.6 ± 1.6	15.0 ± 0.1	68 ± 1.5	76 ± 1	432 ± 5	28 ± 1
1.0	Fxd1	4	502	3.8 ± 1.7	15.0 ± 0.1	70 ± 1.5	74 ± 0	473 ± 5	28 ± 1
1.0	Fxd1	5	502	1.0 ± 0.9	15.1 ± 0.0	71 ± 1.4	80 ± 1	363 ± 4	29 ± 1
1.0	Fxd2	1	502	0.4 ± 0.6	15.1 ± 0.0	64 ± 1.4	97 ± 1	377 ± 5	32 ± 1
1.0	Fxd2	2	502	0.2 ± 0.4	15.1 ± 0.0	75 ± 1.3	105 ± 1	292 ± 3	33 ± 0
1.0	Fxd2	3	500	1.0 ± 0.9	15.1 ± 0.1	57 ± 1.4	88 ± 1	454 ± 6	32 ± 1
1.0	Fxd2	4	502	1.0 ± 0.9	15.1 ± 0.1	54 ± 1.5	86 ± 1	506 ± 7	32 ± 1
1.0	Fxd2	5	502	1.0 ± 0.9	15.1 ± 0.1	69 ± 1.4	93 ± 1	364 ± 4	31 ± 1
1.0	RND	1	502	0.4 ± 0.6	15.1 ± 0.0	66 ± 1.3	94 ± 1	370 ± 4	33 ± 1

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	RND	2	502	0.2 ± 0.4	15.1 ± 0.0	74 ± 1.2	100 ± 1	291 ± 3	33 ± 1
1.0	RND	3	502	0.6 ± 0.7	15.1 ± 0.0	61 ± 1.4	84 ± 1	446 ± 6	31 ± 1
1.0	RND	4	502	2.4 ± 1.3	15.0 ± 0.1	60 ± 1.5	82 ± 1	493 ± 6	31 ± 1
1.0	RND	5	502	0.0 ± 0.0	15.1 ± 0.0	69 ± 1.4	90 ± 1	357 ± 4	31 ± 1
2.0	Fxd1	1	485	0.4 ± 0.6	25.8 ± 0.1	92 ± 0.9	56 ± 0	312 ± 2	31 ± 1
2.0	Fxd1	2	449	0.4 ± 0.6	25.8 ± 0.0	93 ± 0.8	57 ± 1	295 ± 2	31 ± 1
2.0	Fxd1	3	502	0.2 ± 0.4	25.9 ± 0.0	81 ± 1.2	54 ± 0	382 ± 3	32 ± 1
2.0	Fxd1	4	439	0.5 ± 0.6	25.8 ± 0.1	76 ± 1.3	53 ± 0	426 ± 4	32 ± 1
2.0	Fxd1	5	478	0.2 ± 0.4	25.9 ± 0.0	87 ± 1.0	54 ± 0	360 ± 3	31 ± 1
2.0	Fxd2	1	464	0.4 ± 0.6	25.8 ± 0.1	78 ± 1.3	69 ± 1	350 ± 3	34 ± 0
2.0	Fxd2	2	456	0.0 ± 0.0	25.9 ± 0.0	82 ± 1.2	68 ± 1	292 ± 3	34 ± 0
2.0	Fxd2	3	502	0.0 ± 0.0	25.9 ± 0.0	72 ± 1.3	63 ± 1	415 ± 5	34 ± 0
2.0	Fxd2	4	423	0.0 ± 0.0	25.9 ± 0.0	69 ± 1.5	63 ± 1	452 ± 6	35 ± 0
2.0	Fxd2	5	502	0.0 ± 0.0	25.9 ± 0.0	77 ± 1.2	66 ± 1	364 ± 4	33 ± 0
2.0	RND	1	469	0.0 ± 0.0	25.9 ± 0.0	82 ± 1.1	67 ± 1	338 ± 3	34 ± 0
2.0	RND	2	502	0.0 ± 0.0	25.9 ± 0.0	85 ± 1.1	69 ± 1	296 ± 2	34 ± 0
2.0	RND	3	502	0.4 ± 0.6	25.8 ± 0.1	74 ± 1.3	62 ± 1	409 ± 4	34 ± 0
2.0	RND	4	411	0.0 ± 0.0	25.9 ± 0.0	71 ± 1.4	62 ± 1	445 ± 6	35 ± 0
2.0	RND	5	502	0.4 ± 0.6	25.8 ± 0.1	81 ± 1.2	64 ± 1	366 ± 3	33 ± 1
4.0	Fxd1	1	309	0.0 ± 0.0	47.4 ± 0.0	98 ± 0.5	32 ± 0	305 ± 2	31 ± 1
4.0	Fxd1	2	475	0.2 ± 0.4	47.3 ± 0.1	95 ± 0.6	32 ± 0	299 ± 2	31 ± 1
4.0	Fxd1	3	500	0.4 ± 0.6	47.3 ± 0.1	97 ± 0.5	31 ± 0	362 ± 2	30 ± 1
4.0	Fxd1	4	271	0.0 ± 0.0	47.4 ± 0.0	98 ± 0.5	32 ± 0	380 ± 4	31 ± 1
4.0	Fxd1	5	476	0.0 ± 0.0	47.4 ± 0.0	95 ± 0.7	31 ± 0	369 ± 3	31 ± 1
4.0	Fxd2	1	314	0.0 ± 0.0	47.3 ± 0.0	93 ± 1.0	40 ± 0	314 ± 2	35 ± 0
4.0	Fxd2	2	447	0.2 ± 0.4	47.3 ± 0.0	93 ± 0.8	39 ± 0	296 ± 2	34 ± 0
4.0	Fxd2	3	500	0.0 ± 0.0	47.4 ± 0.0	85 ± 1.0	39 ± 0	377 ± 3	35 ± 0
4.0	Fxd2	4	264	0.0 ± 0.0	47.4 ± 0.0	81 ± 1.5	39 ± 0	406 ± 5	35 ± 1
4.0	Fxd2	5	497	0.2 ± 0.4	47.4 ± 0.0	89 ± 0.9	38 ± 0	363 ± 3	34 ± 0
4.0	RND	1	299	0.0 ± 0.0	47.4 ± 0.0	88 ± 1.2	40 ± 1	323 ± 3	35 ± 0
4.0	RND	2	460	0.0 ± 0.0	47.3 ± 0.0	91 ± 0.8	40 ± 0	296 ± 2	34 ± 0
4.0	RND	3	500	0.2 ± 0.4	47.3 ± 0.1	85 ± 1.0	38 ± 0	381 ± 3	35 ± 0
4.0	RND	4	264	0.0 ± 0.0	47.4 ± 0.0	84 ± 1.3	39 ± 0	411 ± 5	35 ± 0
4.0	RND	5	498	0.2 ± 0.4	47.3 ± 0.1	88 ± 0.9	37 ± 0	365 ± 3	34 ± 1
8.0	Fxd1	1	186	0.5 ± 1.1	88.9 ± 0.2	98 ± 0.6	17 ± 0	309 ± 3	31 ± 1
8.0	Fxd1	2	319	0.0 ± 0.0	89.0 ± 0.0	96 ± 0.7	17 ± 0	298 ± 2	31 ± 1
8.0	Fxd1	3	66	0.0 ± 0.0	89.0 ± 0.0	98 ± 1.0	17 ± 0	365 ± 6	31 ± 2
8.0	Fxd1	4	162	0.0 ± 0.0	89.0 ± 0.0	98 ± 0.6	17 ± 0	383 ± 4	32 ± 1
8.0	Fxd1	5	388	0.0 ± 0.0	89.0 ± 0.0	96 ± 0.6	17 ± 0	369 ± 3	31 ± 1
8.0	Fxd2	1	183	0.0 ± 0.0	89.0 ± 0.0	97 ± 0.7	22 ± 0	307 ± 3	35 ± 1
8.0	Fxd2	2	306	0.0 ± 0.0	89.0 ± 0.0	96 ± 0.7	21 ± 0	299 ± 3	34 ± 1
8.0	Fxd2	3	28	0.0 ± 0.0	89.0 ± 0.0	98 ± 1.9	21 ± 1	356 ± 12	35 ± 1
8.0	Fxd2	4	159	0.0 ± 0.0	89.0 ± 0.0	97 ± 0.7	22 ± 0	374 ± 5	35 ± 1
8.0	Fxd2	5	364	0.0 ± 0.0	89.0 ± 0.0	95 ± 0.7	21 ± 0	373 ± 3	34 ± 1
8.0	RND	1	183	0.0 ± 0.0	89.0 ± 0.0	98 ± 0.7	22 ± 0	308 ± 3	35 ± 1

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
8.0	RND	2	306	0.0 ± 0.0	89.0 ± 0.0	96 ± 0.7	21 ± 0	299 ± 2	34 ± 1
8.0	RND	3	29	0.0 ± 0.0	89.0 ± 0.0	96 ± 2.1	21 ± 1	361 ± 14	34 ± 2
8.0	RND	4	170	0.0 ± 0.0	89.0 ± 0.0	97 ± 0.7	22 ± 0	376 ± 4	35 ± 1
8.0	RND	5	365	0.5 ± 0.8	88.9 ± 0.2	95 ± 0.7	21 ± 0	367 ± 3	34 ± 1

Table E.31: Confidence Intervals for DynamicSPR with TinyOS LPL on Cooja Network Size 9

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	Fxd1	1	502	15.9 ± 3.2	11.6 ± 0.1	60 ± 1.4	174 ± 1	496 ± 5	29 ± 1
0.5	Fxd1	2	502	2.4 ± 1.3	11.9 ± 0.0	64 ± 1.1	206 ± 2	383 ± 3	39 ± 1
0.5	Fxd1	3	502	32.3 ± 4.1	11.2 ± 0.1	61 ± 1.6	151 ± 1	579 ± 6	20 ± 2
0.5	Fxd1	4	502	43.6 ± 4.4	11.1 ± 0.1	62 ± 1.6	145 ± 1	630 ± 7	17 ± 2
0.5	Fxd1	5	502	5.6 ± 2.0	11.8 ± 0.1	57 ± 1.2	182 ± 2	462 ± 4	35 ± 1
0.5	Fxd2	1	502	3.2 ± 1.5	11.9 ± 0.1	44 ± 1.4	192 ± 1	520 ± 6	42 ± 1
0.5	Fxd2	2	502	0.2 ± 0.4	12.0 ± 0.0	53 ± 1.2	238 ± 2	388 ± 4	44 ± 1
0.5	Fxd2	3	501	11.4 ± 2.8	11.7 ± 0.1	45 ± 1.5	164 ± 1	593 ± 6	34 ± 1
0.5	Fxd2	4	501	17.0 ± 3.3	11.8 ± 0.1	49 ± 1.6	157 ± 1	657 ± 8	31 ± 2
0.5	Fxd2	5	501	2.2 ± 1.3	12.0 ± 0.0	47 ± 1.2	209 ± 2	458 ± 5	42 ± 1
0.5	RND	1	502	4.8 ± 1.9	11.8 ± 0.1	50 ± 1.4	187 ± 1	511 ± 5	38 ± 1
0.5	RND	2	468	0.2 ± 0.4	11.9 ± 0.0	56 ± 1.1	228 ± 2	387 ± 4	44 ± 1
0.5	RND	3	502	21.1 ± 3.6	11.5 ± 0.1	52 ± 1.6	159 ± 1	588 ± 6	28 ± 2
0.5	RND	4	502	26.9 ± 3.9	11.5 ± 0.1	55 ± 1.7	152 ± 1	646 ± 8	24 ± 2
0.5	RND	5	502	2.6 ± 1.4	11.9 ± 0.1	50 ± 1.2	198 ± 2	452 ± 5	41 ± 1
1.0	Fxd1	1	439	0.0 ± 0.0	19.1 ± 0.0	65 ± 1.1	146 ± 1	491 ± 4	44 ± 1
1.0	Fxd1	2	502	0.0 ± 0.0	19.1 ± 0.0	70 ± 1.0	150 ± 1	393 ± 3	42 ± 1
1.0	Fxd1	3	502	1.2 ± 1.0	19.1 ± 0.1	64 ± 1.1	130 ± 1	585 ± 5	40 ± 1
1.0	Fxd1	4	410	1.7 ± 1.3	19.1 ± 0.1	65 ± 1.2	129 ± 1	643 ± 7	41 ± 1
1.0	Fxd1	5	490	1.0 ± 0.9	19.1 ± 0.0	69 ± 1.1	140 ± 1	485 ± 4	40 ± 1
1.0	Fxd2	1	426	0.2 ± 0.5	19.1 ± 0.1	55 ± 1.3	165 ± 1	510 ± 5	45 ± 1
1.0	Fxd2	2	500	0.0 ± 0.0	19.1 ± 0.0	67 ± 1.0	179 ± 2	391 ± 3	45 ± 1
1.0	Fxd2	3	500	0.0 ± 0.0	19.1 ± 0.0	49 ± 1.2	149 ± 1	625 ± 7	45 ± 1
1.0	Fxd2	4	439	0.2 ± 0.4	19.1 ± 0.0	48 ± 1.4	147 ± 1	685 ± 9	47 ± 1
1.0	Fxd2	5	502	0.0 ± 0.0	19.1 ± 0.0	62 ± 1.1	160 ± 1	483 ± 4	43 ± 1
1.0	RND	1	452	0.0 ± 0.0	19.1 ± 0.0	57 ± 1.2	159 ± 1	506 ± 5	46 ± 1
1.0	RND	2	502	0.0 ± 0.0	19.1 ± 0.0	67 ± 0.9	171 ± 1	393 ± 3	45 ± 1
1.0	RND	3	502	0.4 ± 0.6	19.1 ± 0.0	55 ± 1.2	143 ± 1	602 ± 6	44 ± 1
1.0	RND	4	441	0.2 ± 0.4	19.1 ± 0.0	54 ± 1.4	140 ± 1	671 ± 8	45 ± 1
1.0	RND	5	445	0.4 ± 0.6	19.1 ± 0.0	64 ± 1.1	155 ± 1	490 ± 5	43 ± 1
2.0	Fxd1	1	360	0.3 ± 0.5	33.7 ± 0.1	81 ± 1.1	90 ± 1	435 ± 3	45 ± 1
2.0	Fxd1	2	424	0.0 ± 0.0	33.7 ± 0.0	89 ± 0.8	94 ± 1	384 ± 2	43 ± 1
2.0	Fxd1	3	500	0.0 ± 0.0	33.7 ± 0.0	74 ± 1.1	88 ± 1	537 ± 4	46 ± 1
2.0	Fxd1	4	180	0.0 ± 0.0	33.8 ± 0.0	74 ± 1.9	88 ± 1	584 ± 7	46 ± 1
2.0	Fxd1	5	502	0.0 ± 0.0	33.7 ± 0.0	83 ± 0.9	89 ± 1	478 ± 3	43 ± 1

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	Fxd2	1	311	0.0 ± 0.0	33.8 ± 0.0	76 ± 1.3	115 ± 1	471 ± 5	48 ± 1
2.0	Fxd2	2	414	0.2 ± 0.5	33.7 ± 0.1	82 ± 1.1	114 ± 1	391 ± 3	46 ± 1
2.0	Fxd2	3	500	0.2 ± 0.4	33.7 ± 0.1	72 ± 1.1	107 ± 1	564 ± 5	47 ± 1
2.0	Fxd2	4	281	0.4 ± 0.7	33.7 ± 0.1	70 ± 1.5	106 ± 1	609 ± 7	47 ± 1
2.0	Fxd2	5	450	0.0 ± 0.0	33.7 ± 0.0	78 ± 1.1	108 ± 1	492 ± 4	45 ± 1
2.0	RND	1	324	0.0 ± 0.0	33.8 ± 0.0	76 ± 1.3	112 ± 1	465 ± 4	48 ± 1
2.0	RND	2	416	0.0 ± 0.0	33.8 ± 0.0	84 ± 1.0	113 ± 1	392 ± 2	46 ± 1
2.0	RND	3	500	0.0 ± 0.0	33.7 ± 0.0	72 ± 1.0	104 ± 1	563 ± 5	47 ± 0
2.0	RND	4	283	0.4 ± 0.7	33.7 ± 0.1	68 ± 1.5	103 ± 1	601 ± 7	47 ± 1
2.0	RND	5	436	0.2 ± 0.5	33.7 ± 0.1	80 ± 1.0	105 ± 1	486 ± 4	45 ± 1
4.0	Fxd1	1	218	0.0 ± 0.0	63.2 ± 0.0	91 ± 0.5	52 ± 0	411 ± 3	42 ± 1
4.0	Fxd1	2	338	0.3 ± 0.6	63.1 ± 0.2	89 ± 0.5	52 ± 1	391 ± 2	43 ± 1
4.0	Fxd1	3	500	0.2 ± 0.4	63.1 ± 0.1	91 ± 0.4	51 ± 0	482 ± 2	41 ± 1
4.0	Fxd1	4	207	0.0 ± 0.0	63.1 ± 0.0	91 ± 0.6	53 ± 0	501 ± 4	43 ± 1
4.0	Fxd1	5	417	0.2 ± 0.5	63.1 ± 0.0	88 ± 0.5	50 ± 0	480 ± 3	41 ± 1
4.0	Fxd2	1	211	0.0 ± 0.0	63.2 ± 0.0	79 ± 1.2	64 ± 1	426 ± 3	48 ± 1
4.0	Fxd2	2	351	0.0 ± 0.0	63.2 ± 0.0	85 ± 0.8	63 ± 1	386 ± 2	46 ± 1
4.0	Fxd2	3	89	0.0 ± 0.0	63.2 ± 0.1	76 ± 2.1	63 ± 1	523 ± 8	48 ± 1
4.0	Fxd2	4	184	0.0 ± 0.0	63.1 ± 0.0	74 ± 1.4	63 ± 1	554 ± 7	49 ± 1
4.0	Fxd2	5	368	0.0 ± 0.0	63.2 ± 0.0	81 ± 0.8	62 ± 1	481 ± 3	45 ± 1
4.0	RND	1	230	0.0 ± 0.0	63.2 ± 0.0	81 ± 1.1	67 ± 1	437 ± 3	48 ± 1
4.0	RND	2	322	0.0 ± 0.0	63.1 ± 0.0	85 ± 0.8	64 ± 1	390 ± 2	46 ± 1
4.0	RND	3	500	0.0 ± 0.0	63.2 ± 0.0	81 ± 0.8	63 ± 1	519 ± 3	47 ± 1
4.0	RND	4	201	0.0 ± 0.0	63.2 ± 0.0	80 ± 1.2	64 ± 1	545 ± 6	49 ± 1
4.0	RND	5	388	0.0 ± 0.0	63.2 ± 0.0	82 ± 0.8	61 ± 1	482 ± 3	44 ± 1
8.0	Fxd1	1	133	0.0 ± 0.0	120.9 ± 0.0	98 ± 0.5	27 ± 0	415 ± 3	41 ± 1
8.0	Fxd1	2	232	0.0 ± 0.0	120.9 ± 0.0	96 ± 0.7	28 ± 0	391 ± 2	43 ± 1
8.0	Fxd1	3	146	0.0 ± 0.0	120.9 ± 0.0	97 ± 0.8	26 ± 0	483 ± 4	41 ± 1
8.0	Fxd1	4	114	0.0 ± 0.0	120.9 ± 0.0	98 ± 0.6	28 ± 0	496 ± 5	44 ± 1
8.0	Fxd1	5	283	0.4 ± 0.7	120.8 ± 0.2	95 ± 0.7	27 ± 0	482 ± 3	43 ± 1
8.0	Fxd2	1	142	0.0 ± 0.0	120.9 ± 0.0	98 ± 0.6	35 ± 1	410 ± 3	47 ± 1
8.0	Fxd2	2	230	0.0 ± 0.0	120.9 ± 0.0	96 ± 0.7	34 ± 1	393 ± 3	46 ± 1
8.0	Fxd2	3	143	0.0 ± 0.0	120.9 ± 0.0	96 ± 0.8	34 ± 1	485 ± 5	46 ± 1
8.0	Fxd2	4	111	0.0 ± 0.0	120.9 ± 0.0	97 ± 0.9	35 ± 1	495 ± 6	48 ± 1
8.0	Fxd2	5	269	0.0 ± 0.0	120.9 ± 0.0	95 ± 0.7	33 ± 1	480 ± 3	44 ± 1
8.0	RND	1	130	0.8 ± 1.5	120.5 ± 0.8	97 ± 0.9	36 ± 1	413 ± 4	48 ± 1
8.0	RND	2	223	0.0 ± 0.0	120.9 ± 0.0	96 ± 0.6	35 ± 1	393 ± 2	47 ± 1
8.0	RND	3	158	0.6 ± 1.3	120.6 ± 0.7	94 ± 1.0	35 ± 1	487 ± 4	48 ± 1
8.0	RND	4	121	0.0 ± 0.0	120.9 ± 0.0	92 ± 1.2	36 ± 1	502 ± 5	48 ± 1
8.0	RND	5	270	0.0 ± 0.0	120.9 ± 0.0	94 ± 0.8	33 ± 1	482 ± 3	44 ± 1

Table E.32: Confidence Intervals for DynamicSPR with TinyOS LPL on Cooja Network Size 11

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	Fxd1	1	267	13.5 ± 4.1	13.5 ± 0.1	58 ± 1.8	276 ± 2	623 ± 7	39 ± 2
0.5	Fxd1	2	267	0.0 ± 0.0	13.9 ± 0.0	63 ± 1.4	328 ± 3	482 ± 4	52 ± 1
0.5	Fxd1	3	500	34.8 ± 4.2	12.9 ± 0.1	59 ± 1.5	234 ± 2	727 ± 7	26 ± 2
0.5	Fxd1	4	267	47.9 ± 6.0	12.6 ± 0.2	63 ± 2.0	221 ± 2	782 ± 12	19 ± 3
0.5	Fxd1	5	268	3.0 ± 2.1	13.8 ± 0.1	54 ± 1.4	292 ± 4	575 ± 6	47 ± 2
0.5	Fxd2	1	241	1.7 ± 1.6	13.8 ± 0.1	42 ± 1.8	297 ± 2	642 ± 10	54 ± 2
0.5	Fxd2	2	253	0.4 ± 0.8	13.9 ± 0.0	53 ± 1.5	372 ± 3	491 ± 5	57 ± 1
0.5	Fxd2	3	500	9.2 ± 2.5	13.7 ± 0.1	46 ± 1.5	252 ± 1	754 ± 8	44 ± 2
0.5	Fxd2	4	235	21.3 ± 5.3	13.6 ± 0.2	52 ± 2.3	240 ± 2	810 ± 12	36 ± 3
0.5	Fxd2	5	252	0.0 ± 0.0	13.9 ± 0.0	43 ± 1.6	328 ± 4	577 ± 8	55 ± 1
0.5	RND	1	262	4.2 ± 2.4	13.7 ± 0.1	48 ± 1.7	290 ± 2	646 ± 9	49 ± 2
0.5	RND	2	273	0.0 ± 0.0	13.9 ± 0.0	56 ± 1.4	357 ± 4	484 ± 5	55 ± 1
0.5	RND	3	500	21.0 ± 3.6	13.3 ± 0.1	51 ± 1.5	245 ± 1	745 ± 7	35 ± 2
0.5	RND	4	237	27.4 ± 5.7	13.5 ± 0.2	56 ± 2.4	233 ± 2	797 ± 11	31 ± 3
0.5	RND	5	259	0.8 ± 1.1	13.9 ± 0.1	47 ± 1.6	313 ± 4	569 ± 7	53 ± 1
1.0	Fxd1	1	191	0.5 ± 1.0	22.9 ± 0.0	74 ± 1.6	223 ± 2	617 ± 7	56 ± 2
1.0	Fxd1	2	238	0.0 ± 0.0	22.9 ± 0.0	79 ± 1.4	231 ± 2	489 ± 4	54 ± 1
1.0	Fxd1	3	62	1.6 ± 3.2	22.8 ± 0.3	65 ± 3.5	198 ± 5	736 ± 18	51 ± 3
1.0	Fxd1	4	189	2.1 ± 2.1	22.9 ± 0.1	66 ± 1.8	196 ± 2	808 ± 11	52 ± 2
1.0	Fxd1	5	241	0.0 ± 0.0	22.9 ± 0.0	75 ± 1.4	214 ± 2	602 ± 6	50 ± 2
1.0	Fxd2	1	184	0.0 ± 0.0	23.0 ± 0.0	60 ± 2.0	252 ± 3	655 ± 9	57 ± 1
1.0	Fxd2	2	234	0.0 ± 0.0	23.0 ± 0.0	70 ± 1.5	275 ± 4	491 ± 4	56 ± 1
1.0	Fxd2	3	53	0.0 ± 0.0	23.0 ± 0.1	51 ± 3.3	228 ± 4	790 ± 28	57 ± 2
1.0	Fxd2	4	169	0.0 ± 0.0	23.0 ± 0.0	45 ± 2.2	224 ± 2	870 ± 16	60 ± 1
1.0	Fxd2	5	226	0.0 ± 0.0	23.0 ± 0.0	66 ± 1.5	240 ± 3	613 ± 7	54 ± 1
1.0	RND	1	200	0.0 ± 0.0	23.0 ± 0.0	63 ± 1.8	245 ± 2	641 ± 7	58 ± 1
1.0	RND	2	240	0.0 ± 0.0	23.0 ± 0.0	73 ± 1.4	261 ± 3	491 ± 4	55 ± 1
1.0	RND	3	78	0.0 ± 0.0	22.9 ± 0.1	58 ± 3.1	216 ± 3	755 ± 14	57 ± 2
1.0	RND	4	172	0.0 ± 0.0	23.0 ± 0.0	53 ± 2.0	213 ± 2	842 ± 13	59 ± 1
1.0	RND	5	243	0.0 ± 0.0	22.9 ± 0.0	68 ± 1.5	237 ± 2	599 ± 6	54 ± 1
2.0	Fxd1	1	141	0.0 ± 0.0	41.5 ± 0.0	80 ± 1.6	137 ± 2	560 ± 6	58 ± 1
2.0	Fxd1	2	211	0.0 ± 0.0	41.5 ± 0.0	85 ± 1.1	142 ± 2	480 ± 3	56 ± 1
2.0	Fxd1	3	152	0.7 ± 1.3	41.4 ± 0.2	77 ± 1.5	133 ± 2	676 ± 7	57 ± 1
2.0	Fxd1	4	123	0.0 ± 0.0	41.5 ± 0.1	77 ± 1.8	135 ± 2	740 ± 10	58 ± 1
2.0	Fxd1	5	209	0.0 ± 0.0	41.5 ± 0.0	83 ± 1.2	132 ± 2	593 ± 5	53 ± 1
2.0	Fxd2	1	133	0.8 ± 1.5	41.3 ± 0.3	76 ± 1.7	172 ± 3	585 ± 7	59 ± 2
2.0	Fxd2	2	196	0.0 ± 0.0	41.5 ± 0.0	84 ± 1.3	171 ± 3	489 ± 4	57 ± 1
2.0	Fxd2	3	160	0.6 ± 1.2	41.4 ± 0.2	72 ± 1.8	160 ± 2	702 ± 9	58 ± 1
2.0	Fxd2	4	119	0.8 ± 1.7	41.4 ± 0.3	67 ± 2.2	158 ± 3	757 ± 11	59 ± 2
2.0	Fxd2	5	192	0.0 ± 0.0	41.5 ± 0.0	79 ± 1.5	160 ± 3	604 ± 6	55 ± 1
2.0	RND	1	134	0.0 ± 0.0	41.5 ± 0.0	76 ± 1.7	166 ± 3	586 ± 7	59 ± 1
2.0	RND	2	194	0.0 ± 0.0	41.5 ± 0.0	83 ± 1.3	170 ± 3	489 ± 4	56 ± 1

P_{src} (sec)	App.	TOS LPL	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	RND	3	161	0.0 ± 0.0	41.5 ± 0.0	72 ± 1.8	157 ± 2	697 ± 9	58 ± 1
2.0	RND	4	119	0.0 ± 0.0	41.5 ± 0.1	67 ± 2.1	155 ± 2	739 ± 13	60 ± 1
2.0	RND	5	197	0.0 ± 0.0	41.5 ± 0.0	81 ± 1.3	153 ± 2	605 ± 6	54 ± 2
4.0	Fxd1	1	93	0.0 ± 0.0	78.6 ± 0.1	97 ± 0.8	78 ± 1	513 ± 4	53 ± 2
4.0	Fxd1	2	151	0.0 ± 0.0	78.6 ± 0.0	95 ± 0.8	78 ± 1	480 ± 3	54 ± 2
4.0	Fxd1	3	121	0.0 ± 0.0	78.6 ± 0.1	94 ± 1.1	76 ± 1	600 ± 5	54 ± 2
4.0	Fxd1	4	79	0.0 ± 0.0	78.6 ± 0.1	93 ± 1.4	77 ± 1	623 ± 7	57 ± 2
4.0	Fxd1	5	183	0.0 ± 0.0	78.6 ± 0.0	92 ± 1.0	76 ± 1	593 ± 4	53 ± 1
4.0	Fxd2	1	98	0.0 ± 0.0	78.6 ± 0.1	85 ± 1.7	96 ± 1	545 ± 6	61 ± 1
4.0	Fxd2	2	143	0.0 ± 0.0	78.6 ± 0.1	89 ± 1.3	94 ± 2	483 ± 4	58 ± 1
4.0	Fxd2	3	124	0.0 ± 0.0	78.6 ± 0.1	83 ± 1.6	95 ± 2	653 ± 7	60 ± 1
4.0	Fxd2	4	77	0.0 ± 0.0	78.6 ± 0.1	81 ± 2.2	95 ± 2	685 ± 11	60 ± 2
4.0	Fxd2	5	173	0.0 ± 0.0	78.6 ± 0.0	87 ± 1.1	89 ± 2	596 ± 5	55 ± 2
4.0	RND	1	90	2.2 ± 3.1	77.8 ± 1.1	88 ± 1.8	98 ± 3	543 ± 6	58 ± 2
4.0	RND	2	155	0.0 ± 0.0	78.6 ± 0.0	91 ± 1.0	93 ± 2	487 ± 3	56 ± 1
4.0	RND	3	124	0.0 ± 0.0	78.7 ± 0.1	88 ± 1.4	96 ± 1	641 ± 8	61 ± 1
4.0	RND	4	77	1.3 ± 2.6	78.2 ± 0.9	85 ± 2.0	95 ± 2	672 ± 11	60 ± 2
4.0	RND	5	172	0.0 ± 0.0	78.6 ± 0.0	89 ± 1.1	90 ± 2	598 ± 4	56 ± 1
8.0	Fxd1	1	61	0.0 ± 0.0	151.5 ± 0.0	98 ± 0.9	41 ± 1	510 ± 4	55 ± 2
8.0	Fxd1	2	104	0.0 ± 0.0	151.5 ± 0.0	96 ± 0.8	41 ± 1	486 ± 3	54 ± 2
8.0	Fxd1	3	78	0.0 ± 0.0	151.5 ± 0.0	97 ± 0.8	40 ± 1	595 ± 6	53 ± 2
8.0	Fxd1	4	47	0.0 ± 0.0	151.5 ± 0.0	98 ± 1.0	41 ± 1	619 ± 7	54 ± 2
8.0	Fxd1	5	129	0.0 ± 0.0	151.5 ± 0.0	94 ± 0.9	40 ± 1	595 ± 4	52 ± 2
8.0	Fxd2	1	60	0.0 ± 0.0	151.5 ± 0.0	98 ± 0.9	52 ± 1	515 ± 4	61 ± 1
8.0	Fxd2	2	100	0.0 ± 0.0	151.5 ± 0.0	96 ± 0.9	50 ± 1	485 ± 3	55 ± 2
8.0	Fxd2	3	75	0.0 ± 0.0	151.5 ± 0.0	95 ± 1.0	51 ± 1	601 ± 6	59 ± 2
8.0	Fxd2	4	46	2.2 ± 4.4	150.0 ± 3.0	94 ± 1.9	51 ± 2	616 ± 9	59 ± 3
8.0	Fxd2	5	132	0.0 ± 0.0	151.5 ± 0.0	94 ± 0.9	48 ± 1	599 ± 4	55 ± 1
8.0	RND	1	54	0.0 ± 0.0	151.5 ± 0.0	95 ± 1.4	53 ± 2	518 ± 5	59 ± 2
8.0	RND	2	101	0.0 ± 0.0	151.5 ± 0.0	95 ± 1.2	50 ± 2	486 ± 3	57 ± 2
8.0	RND	3	76	0.0 ± 0.0	151.5 ± 0.0	92 ± 1.5	51 ± 1	610 ± 6	59 ± 2
8.0	RND	4	45	0.0 ± 0.0	151.5 ± 0.0	89 ± 2.1	53 ± 2	635 ± 10	61 ± 2
8.0	RND	5	123	0.0 ± 0.0	151.5 ± 0.0	94 ± 1.1	48 ± 1	591 ± 5	53 ± 2

E.3 FlockLab

E.3.1 Protectionless

Table E.33: Confidence Intervals for Protectionless on FlockLab

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	14	100.0 ± 0.0	2.8 ± 0.7	94 ± 8.5	36 ± 3	143 ± 17	0 ± 0
1.0	14	100.0 ± 0.0	5.4 ± 0.7	92 ± 6.8	18 ± 1	144 ± 14	0 ± 0

P_{src} (sec)	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
2.0	14	100.0 ± 0.0	10.5 ± 2.1	98 ± 3.9	10 ± 1	144 ± 16	0 ± 0
8.0	16	100.0 ± 0.0	46.0 ± 5.5	89 ± 7.5	2 ± 0	151 ± 10	0 ± 0

E.3.2 DynamicSPR

Table E.34: Confidence Intervals for DynamicSPR on FlockLab

P_{src} (sec)	App.	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
0.5	Fxd1	5	20.0 ± 55.5	4.3 ± 1.9	78 ± 25.8	127 ± 20	207 ± 30	264 ± 262
0.5	Fxd2	5	0.0 ± 0.0	5.0 ± 0.0	70 ± 13.1	151 ± 14	273 ± 77	536 ± 139
1.0	Fxd1	5	0.0 ± 0.0	9.9 ± 0.0	76 ± 26.5	68 ± 7	160 ± 20	360 ± 165
1.0	Fxd2	5	0.0 ± 0.0	9.9 ± 0.0	77 ± 9.9	89 ± 21	189 ± 32	489 ± 188
2.0	Fxd1	5	0.0 ± 0.0	19.6 ± 0.0	89 ± 13.8	35 ± 2	159 ± 35	460 ± 150
2.0	Fxd2	5	0.0 ± 0.0	19.2 ± 0.2	87 ± 11.5	49 ± 7	154 ± 30	583 ± 108
8.0	Fxd1	9	0.0 ± 0.0	86.0 ± 0.0	93 ± 5.4	8 ± 0	145 ± 11	407 ± 105
8.0	Fxd2	7	0.0 ± 0.0	85.8 ± 0.3	97 ± 4.5	12 ± 2	147 ± 15	539 ± 56

E.3.3 DynamicSPR with Duty Cycling

Table E.35: Confidence Intervals for DynamicSPR with Duty Cycling on FlockLab

P_{src} (sec)	App.	DC	R	Captured (%)	$\mathcal{TT}$ (sec)	Received (%)	Sent per sec	Latency (ms)	Attacker Distance
1.0	Fxd1	5	9	22.2 ± 33.9	9.3 ± 1.0	87 ± 11.0	50 ± 3	186 ± 14	325 ± 164
1.0	Fxd1	2	11	9.1 ± 20.3	9.6 ± 0.6	91 ± 7.3	55 ± 3	188 ± 11	374 ± 110
1.0	Fxd1	6	13	0.0 ± 0.0	9.9 ± 0.0	88 ± 8.4	57 ± 3	193 ± 13	426 ± 43
1.0	Fxd2	5	7	28.6 ± 45.1	8.9 ± 1.6	44 ± 26.3	50 ± 5	217 ± 37	304 ± 198
1.0	Fxd2	2	8	0.0 ± 0.0	9.9 ± 0.0	56 ± 12.2	59 ± 4	231 ± 38	455 ± 95
1.0	Fxd2	6	8	0.0 ± 0.0	9.9 ± 0.0	60 ± 17.2	64 ± 5	258 ± 35	455 ± 45
2.0	Fxd1	5	8	0.0 ± 0.0	19.5 ± 0.1	75 ± 22.6	23 ± 4	204 ± 23	299 ± 102
2.0	Fxd1	2	9	0.0 ± 0.0	19.5 ± 0.1	88 ± 10.8	25 ± 3	198 ± 14	411 ± 112
2.0	Fxd1	6	10	10.0 ± 22.6	18.4 ± 2.2	84 ± 16.0	26 ± 2	213 ± 33	408 ± 125
2.0	Fxd2	5	8	25.0 ± 38.7	17.5 ± 2.8	82 ± 12.0	29 ± 5	207 ± 13	299 ± 185
2.0	Fxd2	2	10	0.0 ± 0.0	19.1 ± 0.0	81 ± 15.5	33 ± 3	215 ± 27	397 ± 76
2.0	Fxd2	6	9	0.0 ± 0.0	19.1 ± 0.0	80 ± 16.4	36 ± 3	204 ± 12	446 ± 79
8.0	Fxd1	5	6	16.7 ± 42.8	80.8 ± 13.3	81 ± 14.9	6 ± 1	186 ± 19	336 ± 181
8.0	Fxd1	2	8	0.0 ± 0.0	85.9 ± 0.1	95 ± 4.5	7 ± 0	196 ± 18	349 ± 88
8.0	Fxd1	6	8	0.0 ± 0.0	85.8 ± 0.1	95 ± 4.5	7 ± 0	196 ± 11	431 ± 47
8.0	Fxd2	5	15	13.3 ± 19.5	84.4 ± 2.4	85 ± 9.8	8 ± 1	196 ± 10	354 ± 127
8.0	Fxd2	2	11	0.0 ± 0.0	85.9 ± 0.0	95 ± 5.5	9 ± 1	201 ± 11	401 ± 64
8.0	Fxd2	6	13	0.0 ± 0.0	85.5 ± 0.4	95 ± 5.3	9 ± 1	204 ± 15	447 ± 24

Bibliography

[1] IEEE standard for low-rate wireless networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pages 1–709, April 2016. doi: 10.1109/IEEESTD.2016.7460875.

[2] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. FIT IoT-LAB: A large scale open experimental IoT testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464, Dec 2015. doi: 10.1109/WF-IoT.2015.7389098.

[3] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 247–255. ACM, 2001. ISBN 1-58113-361-8. doi: 10.1145/375551.375602.

[4] R. Anastácio, S. Cardoso, and M. J. Pereira. Spy out to protect: Sensing devices for wildlife virtual fencing. *Open Journal of Ecology*, 8(3):192–208, March 2018. ISSN 2162-1993. doi: 10.4236/oje.2018.83013.

[5] T. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724, June 2005. doi: 10.1109/.2005.1467103.

[6] S. Armenia, G. Morabito, and S. Palazzo. Analysis of location privacy/energy efficiency tradeoffs in wireless sensor networks. In *Proceedings of the 6^{th} international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet*, NETWORKING'07, pages 215–226, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72606-7. doi: 10.1007/978-3-540-72606-7_19.

[7] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Comput. Netw.*, 46(5):605–634, December 2004. ISSN 1389-1286. doi: 10.1016/j.comnet.2004.06.007.

[8] R. Asorey-Cacheda, A.-J. Garcia-Sanchez, F. Garcia-Sánchez, and J. Garcia-Haro. A survey on non-linear optimization problems in wireless sensor networks. *Journal of*

Network and Computer Applications, 82:1–20, 2017. ISSN 1084-8045. doi: 10.1016/j.jnca.2017.01.001.

[9] E. Baccelli, O. Hahm, M. Wählisch, M. Günes, and T. Schmidt. RIOT: One OS to Rule Them All in the IoT. Research Report RR-8176, INRIA, December 2012. URL `https://hal.inria.fr/hal-00768685`.

[10] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt. RIOT OS: Towards an OS for the Internet of Things. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 79–80, April 2013. doi: 10.1109/INFCOMW.2013.6970748.

[11] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*, 8(4):34:1–34:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240123.

[12] A.-M. Badescu and L. Cotofana. A wireless sensor network to monitor and protect tigers in the wild. *Ecological Indicators*, 57:447–451, 2015. ISSN 1470-160X. doi: 10.1016/j.ecolind.2015.05.022.

[13] Z. Benenson, P. M. Cholewinski, and F. C. Freiling. *Wireless Sensors Networks Security*, chapter Vulnerabilities and Attacks in Wireless Sensor Networks, pages 22–43. IOS Press, 2008.

[14] L. Benini, D. Brunelli, C. Petrioli, and S. Silvestri. GENESI: Green sEnsor NEtworks for Structural monItoring. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1–3, June 2010. doi: 10.1109/SECON.2010.5508230.

[15] E. Bertino, I. N. Fovino, and L. P. Provenza. A framework for evaluating privacy preserving data mining algorithms*. *Data Min. Knowl. Discov.*, 11(2):121–154, September 2005. ISSN 1384-5810. doi: 10.1007/s10618-005-0006-6.

[16] E. Bertino, D. Lin, and W. Jiang. *A Survey of Quantification of Privacy Preserving Data Mining Algorithms*, pages 183–205. Springer US, Boston, MA, 2008. ISBN 978-0-387-70992-5. doi: 10.1007/978-0-387-70992-5_8.

[17] E. S. Biagioni and K. W. Bridges. The application of remote sensor technology to assist the recovery of rare and endangered species. *International Journal of High Performance Computing Applications*, 16:2002, 2002.

[18] J. Blobel and F. Dressler. Sender-Triggered Selective Wake-Up Receiver for Low-Power Sensor Networks. In *36th IEEE Conference on Computer Communications (INFOCOM 2017), Demo Session*, Atlanta, GA, May 2017. IEEE.

[19] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on*

Discrete Algorithms and Methods for Mobile Computing and Communications, DIALM '99, pages 48–55. ACM, 1999. ISBN 1-58113-174-7. doi: 10.1145/313239.313282.

[20] A. Boubrima, W. Bechkit, and H. Rivano. Optimal WSN deployment models for air pollution monitoring. *IEEE Transactions on Wireless Communications*, PP(99):1–1, 2017. ISSN 1536-1276. doi: 10.1109/TWC.2017.2658601.

[21] M. Bradbury and A. Jhumka. A near-optimal source location privacy scheme for wireless sensor networks. In *16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 409–416, August 2017. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.265.

[22] M. Bradbury and A. Jhumka. Understanding source location privacy protocols in sensor networks via perturbation of time series. In *INFOCOM, 2017 Proceedings IEEE*, pages 1611–1619, May 2017. doi: 10.1109/INFOCOM.2017.8057122.

[23] M. Bradbury, A. Jhumka, and C. Maple. A novel source location privacy-aware duty cycle for wireless sensor networks. Manuscript submitted for publication.

[24] M. Bradbury, M. Leeke, and A. Jhumka. A dynamic fake source algorithm for source location privacy in wireless sensor networks. In *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 531–538, August 2015. doi: 10.1109/Trustcom.2015.416.

[25] M. Bradbury, A. Jhumka, and M. Leeke. Hybrid online protocols for source location privacy in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 115:67–81, May 2018. ISSN 0743-7315. doi: 10.1016/j.jpdc.2018.01.006.

[26] M. Brownfield, Y. 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.

[27] A. Bushang and A. Mahmood. A specialized event-driven network simulator for security and anonymity applications of wireless sensor networks. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–6, January 2017. doi: 10.1109/CCWC.2017.7868420.

[28] L. Buttyán, T. Holczer, A. Weimerskirch, and W. Whyte. SLOW: A Practical pseudonym changing scheme for location privacy in VANETs. In *2009 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, Oct 2009. doi: 10.1109/VNC.2009.5416380.

[29] F. Cadger, K. Curran, J. Santos, and S. Moffett. A survey of geographical routing in wireless ad-hoc networks. *IEEE Communications Surveys Tutorials*, 15(2):621–653, Second 2013. ISSN 1553-877X. doi: 10.1109/SURV.2012.062612.00109.

[30] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque. Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(1):181–194, First 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.052213.00116.

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

[32] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev.*, 31(2 supplement):20–41, April 2001. ISSN 0146-4833. doi: 10.1145/844193.844196.

[33] M. Chae, H. Yoo, J. Kim, and M. Cho. Development of a wireless sensor network system for suspension bridge health monitoring. *Automation in Construction*, 21(0): 237–252, 2012. ISSN 0926-5805. doi: 10.1016/j.autcon.2011.06.008.

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

[35] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981. ISSN 0001-0782. doi: 10.1145/358549.358563.

[36] H. Chen and W. Lou. From nowhere to somewhere: Protecting end-to-end location privacy in wireless sensor networks. In *Performance Computing and Communications Conference (IPCCC), 2010 IEEE 29th International*, pages 1–8, December 2010. doi: 10.1109/PCCC.2010.5682341.

[37] J. Chen, Z. Lin, D. Wu, and B. Wang. Enhancing source location protection in wireless sensor networks. In *Sixth International Conference on Electronics and Information Engineering*, volume 9794, pages 0–5, dec 2015. doi: 10.1117/12.2205835.

[38] W. Chen, M. Zhang, G. Hu, X. Tang, and A. K. Sangaiah. Constrained random routing mechanism for source privacy protection in wsns. *IEEE Access*, PP(99):1–1, 2017. doi: 10.1109/ACCESS.2017.2752179.

[39] X. Chen, K. Makki, K. Yen, and N. Pissinou. Sensor network security: a survey. *IEEE Communications Surveys Tutorials*, 11(2):52–73, Second 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090205.

[40] M. Conti, J. Willemsen, and B. Crispo. Providing source location privacy in wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 15(3): 1238–1280, 2013. ISSN 1553-877X. doi: 10.1109/SURV.2013.011413.00118.

[41] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005*, pages 113–126, 2005. doi: 10.1109/SECURECOMM.2005.16.

[42] J. Deng, R. Han, and S. Mishra. Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks. *Pervasive and Mobile Computing*, 2(2):159–186, 2006. ISSN 1574-1192. doi: 10.1016/j.pmcj.2005.12.003. Security in Wireless Mobile Computing Systems.

[43] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. *Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed*, pages 302–316. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-29273-6. doi: 10.1007/978-3-642-29273-6_23.

[44] M. Dong, K. Ota, and A. Liu. Preserving source-location privacy through redundant fog loop for wireless sensor networks. In *13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 1835–1842, Liverpool, UK, October 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.274.

[45] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, 2004. doi: 10.1109/LCN.2004.38.

[46] N. Dutta, A. Saxena, and S. Chellappan. Defending wireless sensor networks against adversarial localization. In *2010 Eleventh International Conference on Mobile Data Management (MDM)*, pages 336–341, May 2010. doi: 10.1109/MDM.2010.75.

[47] V. Dyo, S. A. Ellwood, D. W. Macdonald, A. Markham, N. Trigoni, R. Wohlers, C. Mascolo, B. Pásztor, S. Scellato, and K. Yousef. Wildsensing: Design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Trans. Sen. Netw.*, 8(4): 29:1–29:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240118.

[48] R. El-Badry, A. Sultan, and M. Youssef. Hyberloc: Providing physical layer location privacy in hybrid sensor networks. In *2010 IEEE International Conference on Communications (ICC)*, pages 1–5, May 2010. doi: 10.1109/ICC.2010.5502104.

[49] K. Emara. Safety-aware location privacy in VANET: Evaluation and comparison. *IEEE Transactions on Vehicular Technology*, 66(12):10718–10731, Dec 2017. ISSN 0018-9545. doi: 10.1109/TVT.2017.2736885.

[50] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón. Cooja/MSPSim: interoperability testing for wireless sensor networks. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 27:1–27:7, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-45-5. doi: 10.4108/ICST.SIMUTOOLS2009.5637.

[51] Y. Fan, Y. Jiang, H. Zhu, and X. Shen. An efficient privacy-preserving scheme against traffic analysis attacks in network coding. In *INFOCOM, 2009 Proceedings IEEE*, pages 2213–2221, April 2009. doi: 10.1109/INFCOM.2009.5062146.

[52] Y. Fan, J. Chen, X. Lin, and X. Shen. Preventing traffic explosion and achieving source unobservability in multi-hop wireless networks using network coding. In *Global*

Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 1–5, December 2010. doi: 10.1109/GLOCOM.2010.5683317.

[53] D. Fotue, G. Tanonkou, and T. Engel. An ad-hoc wireless sensor networks with application to air pollution detection. In G. K. Lee, editor, *Proceedings of the ISCA First International Conference on Sensor Networks and Applications (SNA-2009), November 4-6, 2009, Hilton San Francisco Fisherman s Wharf, San Francisco, California, USA*, pages 48–53. ISCA, 2009.

[54] J. Freudiger, M. H. Manshaei, J.-P. Hubaux, and D. C. Parkes. Non-cooperative location privacy. *IEEE Transactions on Dependable and Secure Computing*, 10(2): 84–98, March 2013. ISSN 1545-5971. doi: 10.1109/TDSC.2012.85.

[55] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[56] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, PLDI '03, pages 1–11. ACM, 2003. ISBN 1-58113-662-5. doi: 10.1145/781131.781133.

[57] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis. Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM Trans. Sen. Netw.*, 10(1):16:1–16:49, December 2013. ISSN 1550-4859. doi: 10.1145/2529988.

[58] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, feb 1999. doi: 10.1145/293411.293443.

[59] C. Gu, M. Bradbury, A. Jhumka, and M. Leeke. Assessing the performance of phantom routing on source location privacy in wireless sensor networks. In *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 99–108, November 2015. doi: 10.1109/PRDC.2015.9.

[60] C. Gu, M. Bradbury, and A. Jhumka. Phantom walkabouts in wireless sensor networks. In *Proceedings of the Symposium on Applied Computing*, SAC'17, pages 609–616. ACM, April 2017. ISBN 978-1-4503-4486-9. doi: 10.1145/3019612.3019732.

[61] C. Gu, M. Bradbury, J. Kirton, and A. Jhumka. A decision theoretic framework for selecting source location privacy aware routing protocols in wireless sensor networks. *Future Generation Computing Systems*, 87:514–526, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2018.01.046.

[62] L. Guo, L. Wang, J. Liu, and W. Zhou. A survey on motion detection using wifi signals. In *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 202–206, Dec 2016. doi: 10.1109/MSN.2016.040.

[63] G. Han, L. Zhou, H. Wang, W. Zhang, and S. Chan. A source location protection protocol based on dynamic routing in {WSNs} for the social internet of things. *Future Generation Computer Systems*, 82:689 – 697, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2017.08.044. URL `http://www.sciencedirect.com/science/article/pii/S0167739X17310348`.

[64] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the 2Nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, REALMAN '06, pages 63–70, New York, NY, USA, 2006. ACM. ISBN 1-59593-360-3. doi: 10.1145/1132983.1132995.

[65] J. Hao, B. Zhang, and H. T. Mouftah. Routing protocols for duty cycled wireless sensor networks: A survey. *IEEE Communications Magazine*, 50(12):116–123, December 2012. ISSN 0163-6804. doi: 10.1109/MCOM.2012.6384460.

[66] C. Hartung, R. Han, C. Seielstad, and S. Holbrook. FireWxNet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the 4^{th} international conference on Mobile systems, applications and services*, MobiSys '06, pages 28–41. ACM, 2006. ISBN 1-59593-195-3. doi: 10.1145/1134680.1134685.

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

[68] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, October 2002. ISSN 1536-1276. doi: 10.1109/TWC.2002.804190.

[69] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205, September 2005. doi: 10.1109/SECURECOMM.2005.33.

[70] X. Hong, P. Wang, J. Kong, Q. Zheng, and J. Liu. Effective probabilistic approach protecting sensor traffic. In *Military Communications Conference (MILCOM), 2005 IEEE*, pages 169–1751, October 2005. doi: 10.1109/MILCOM.2005.1605681.

[71] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. MQTT-S — a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference On Communication Systems Software And Middleware And Workshops (comsware '08)*, pages 791–798, January 2008. doi: 10.1109/COMSWA.2008.4554519.

[72] J. Hwang, T. He, and Y. Kim. Secure localization with phantom node detection. *Ad Hoc Networks*, 6(7):1031–1050, 2008. ISSN 1570-8705. doi: 10.1016/j.adhoc.2007.10.005.

[73] Intel. Anti-poaching technology: Wearables are helping save rhinos. Online, 2014. URL http://web.archive.org/web/20160324001414/iq.intel.com/how-wearable-technology-is-helping-saving-rhinos-from-poachers. Accessed: 2016-06-29.

[74] A. Jhumka and M. Bradbury. Deconstructing source location privacy-aware routing protocols. In *Proceedings of the Symposium on Applied Computing*, SAC'17, pages 431–436. ACM, April 2017. ISBN 978-1-4503-4486-9. doi: 10.1145/3019612.3019655.

[75] A. Jhumka, M. Leeke, and S. 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.

[76] A. Jhumka, M. Bradbury, and M. Leeke. Towards understanding source location privacy in wireless sensor networks through fake sources. In *11^{th} IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 760–768, June 2012. doi: 10.1109/TrustCom.2012.281.

[77] A. Jhumka, M. Bradbury, and S. Saginbekov. Efficient fault-tolerant collision-free data aggregation scheduling for wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1789–1801, 2014. ISSN 0743-7315. doi: 10.1016/j.jpdc.2013.09.011.

[78] A. Jhumka, M. Bradbury, and M. Leeke. Fake source-based source location privacy in wireless sensor networks. *Concurrency and Computation: Practice and Experience*, 27 (12):2999–3020, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3242.

[79] Z. Jia, X. Wei, H. Guo, W. Peng, and C. Song. A privacy protection strategy for source location in wsn based on angle and dynamical adjustment of node emission radius. *Chinese Journal of Electronics*, 26(5):1064–1072, 2017. ISSN 1022-4653. doi: 10.1049/cje.2016.08.022.

[80] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, October 2002. ISSN 0163-5980. doi: 10.1145/635508.605408.

[81] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *25^{th} IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 599–608, June 2005. doi: 10.1109/ICDCS.2005.31.

[82] P. Kamat, W. Xu, W. Trappe, and Y. Zhang. Temporal privacy in wireless sensor networks. In *Distributed Computing Systems, 2007. ICDCS '07. 27^{th} International Conference on*, pages 23–23, June 2007. doi: 10.1109/ICDCS.2007.146.

[83] P. Kamat, W. Xu, W. Trappe, and Y. Zhang. Temporal privacy in wireless sensor networks: Theory and practice. *ACM Trans. Sen. Netw.*, 5(4):28:1–28:24, November 2009. ISSN 1550-4859. doi: 10.1145/1614379.1614380.

[84] D. Kar, B. Ford, S. Gholami, F. Fang, A. Plumptre, M. Tambe, M. Driciru, F. Wanyama, A. Rwetsiba, M. Nsubaga, and J. Mabonga. Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 159–167, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[85] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings Of The 6th Annual International Conference On Mobile Computing And Networking*, MobiCom '00, pages 243–254. ACM, 2000. ISBN 1-58113-197-6. doi: 10.1145/345910.345953.

[86] F. Kazemeyni, E. B. Johnsen, O. Owe, and I. Balasingham. Mule-based wireless sensor networks: Probabilistic modeling and quantitative analysis. In J. Derrick, S. Gnesi, D. Latella, and H. Treharne, editors, *Integrated Formal Methods*, pages 143–157, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30729-4.

[87] K. K. Khedo, R. Perseedoss, and A. 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.

[88] J. Kirton, M. Bradbury, and A. Jhumka. Source location privacy-aware data aggregation scheduling for wireless sensor networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2200–2205, June 2017. doi: 10.1109/ICDCS.2017.171.

[89] J. Kirton, M. Bradbury, and A. Jhumka. Towards optimal source location privacy-aware TDMA schedules in wireless sensor networks. *Computer Networks*, 146:125–137, 2018. ISSN 1389-1286. doi: 10.1016/j.comnet.2018.09.010.

[90] T. Koch, T. Ralphs, and Y. Shinano. Could we use a million cores to solve an integer program? *Mathematical Methods of Operations Research*, 76(1):67–93, Aug 2012. ISSN 1432-5217. doi: 10.1007/s00186-012-0390-9.

[91] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22 (1):79–86, 03 1951. doi: 10.1214/aoms/1177729694.

[92] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. In *2003 Ieee Wireless Communications And Networking, 2003. Wcnc 2003.*, volume 3, pages 1954–1961 vol.3, March 2003. doi: 10.1109/WCNC.2003.1200686.

[93] J. F. Laikin, M. Bradbury, C. Gu, and M. Leeke. Towards fake sources for source location privacy in wireless sensor networks with multiple sources. In *15th IEEE International Conference on Communication Systems (ICCS'16)*, pages 1–6, December 2016. doi: 10.1109/ICCS.2016.7833572.

[94] L. Lazos and R. Poovendran. HiRLoc: High-resolution robust localization for wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 24(2):233–246, February 2006. ISSN 0733-8716. doi: 10.1109/JSAC.2005.861381.

[95] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 126–137. ACM, 2003. ISBN 1-58113-707-9. doi: 10.1145/958491.958506.

[96] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-23867-6. doi: 10.1007/3-540-27139-2_7.

[97] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 7(8):1501–1514, 2009. ISSN 1570-8705. doi: 10.1016/j.adhoc.2009.04.009. Privacy and Security in Wireless Sensor and Ad Hoc Networks.

[98] N. Li, M. Raj, D. Liu, M. Wright, and S. K. Das. Using data mules to preserve source location privacy in wireless sensor networks. In *Proceedings of the 13th International Conference on Distributed Computing and Networking*, ICDCN'12, pages 309–324, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-25958-6. doi: 10.1007/978-3-642-25959-3_23.

[99] X. Li, X. Wang, N. Zheng, Z. Wan, and M. Gu. Enhanced location privacy protection of base station in wireless sensor networks. In *5th International Conference on Mobile Ad-hoc and Sensor Networks, 2009. MSN '09.*, pages 457–464, 2009. doi: 10.1109/MSN.2009.19.

[100] Y. Li and J. Ren. Mixing ring-based source-location privacy in wireless sensor networks. In *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009. ICCCN 2009*, pages 1–6, August 2009. doi: 10.1109/ICCCN.2009.5235216.

[101] Y. Li and J. Ren. Preserving source-location privacy in wireless sensor networks. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09.*, pages 1–9, 2009. ISBN 978-1-4244-2907-3. doi: 10.1109/SAHCN.2009.5168962.

[102] Y. Li and J. Ren. Providing source-location privacy in wireless sensor networks. In B. Liu, A. Bestavros, D.-Z. Du, and J. Wang, editors, *Wireless Algorithms, Systems, and Applications*, volume 5682 of *Lecture Notes in Computer Science*, pages 338–347. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03416-9. doi: 10.1007/978-3-642-03417-6_33.

[103] Y. Li, L. Lightfoot, and J. Ren. Routing-based source-location privacy protection in wireless sensor networks. In *IEEE International Conference on Electro/Information Technology, 2009. eit '09.*, pages 29–34, June 2009. doi: 10.1109/EIT.2009.5189579.

[104] Y. Li, J. Ren, and J. Wu. Quantitative measurement and design of source-location privacy schemes for wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(7):1302–1311, July 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2011.260.

[105] Libelium. Waspmote Power Programming Guide, 2012. URL `https://www.libelium.com/v11-files/documentation/waspmote/waspmote-power-programming_guide.pdf`. Accessed: 2018-10-07.

[106] Libelium. Waspmote Datasheet, 2016. URL `http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf`. Accessed: 2018-09-17.

[107] Libelium. Waspmote LoRaWAN Networking Guide, 2018. URL `http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf`. Accessed: 2018-09-17.

[108] L. Lightfoot, Y. Li, and J. Ren. Star: design and quantitative measurement of source-location privacy for wireless sensor networks. *Security and Communication Networks*, 9(3):220–228, 2016. ISSN 1939-0122. doi: 10.1002/sec.527.

[109] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 153–165, April 2013. doi: 10.1145/2461381.2461402.

[110] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, Jan 1991. ISSN 0018-9448. doi: 10.1109/18.61115.

[111] Linear Technology Corporation. LTC5800-IPM - SmartMesh IP Node 2.4GHz 802.15.4e Wireless Mote-on-Chip, December 2015. URL `http://www.analog.com/media/en/technical-documentation/data-sheets/5800ipmfa.pdf`. Accessed: 2018-09-17.

[112] B. Liu, W. Zhou, T. Zhu, L. Gao, and Y. Xiang. Location privacy and its applications: A systematic study. *IEEE Access*, 6:17606–17624, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2822260.

[113] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 52–61. ACM, 2003. ISBN 1-58113-738-9. doi: 10.1145/948109.948119.

[114] K. Liu, Q. Ma, H. Liu, Z. Cao, and Y. Liu. End-to-end delay measurement in wireless sensor networks without synchronization. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 583–591, October 2013. doi: 10.1109/MASS.2013.71.

[115] J. Long, M. Dong, K. Ota, and A. Liu. Achieving source location privacy and network lifetime maximization through tree-based diversionary routing in wireless sensor networks. *IEEE Access*, 2:633–651, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2332817.

[116] R. Lu, X. Lin, T. H. Luan, X. Liang, and X. Shen. Pseudonym changing at social spots: An effective strategy for location privacy in vanets. *IEEE Transactions on Vehicular Technology*, 61(1):86–96, Jan 2012. ISSN 0018-9545. doi: 10.1109/TVT.2011.2162864.

[117] M. Mahmoud and X. Shen. A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(10):1805–1818, October 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2011.302.

[118] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 88–97. ACM, 2002. ISBN 1-58113-589-0. doi: 10.1145/570738.570751.

[119] R. Manjula and R. Datta. An energy-efficient routing technique for privacy preservation of assets monitored with WSN. In *Students' Technology Symposium (TechSym), 2014 IEEE*, pages 325–330, February 2014. doi: 10.1109/TechSym.2014.6808069.

[120] R. Manjula and R. Datta. A novel source location privacy preservation technique to achieve enhanced privacy and network lifetime in WSNs. *Pervasive and Mobile Computing*, 44:58 – 73, 2018. ISSN 1574-1192. doi: 10.1016/j.pmcj.2018.01.006.

[121] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 39–49. ACM, 2004. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031501.

[122] K. Mehta, D. Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *2007 IEEE International Conference on Network Protocols*, pages 314–323, October 2007. doi: 10.1109/ICNP.2007.4375862.

[123] K. Mehta, D. Liu, and M. Wright. Protecting location privacy in sensor networks against a global eavesdropper. *IEEE Trans. on Mobile Computing*, 11(2):320–336, February 2012. ISSN 1536-1233. doi: 10.1109/TMC.2011.32.

[124] Memsic. MICAz, 2011. URL `www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf`. Accessed: 2018-09-17.

[125] D. Moss, J. Hui, and K. Klues. Low power listening. Technical report, TinyOS Core Working Group, 2007. URL `http://www.btnode.ethz.ch/static_docs/tinyos-2.x/pdf/tep105.pdf`. TEP 105.

[126] moteiv. tmote sky, 2007. URL `https://insense.cs.st-andrews.ac.uk/files/2013/04/tmote-sky-datasheet.pdf`. Accessed: 2018-09-17.

[127] A. Nassiri, M. A. Razzaque, and A. H. Abdullah. Isolated adversary zone for source location privacy in wireless sensor networks. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 108–113, September 2016. doi: 10.1109/IWCMC.2016.7577042.

[128] A. A. Nezhad, A. Miri, and D. Makrakis. Location privacy and anonymity preserving routing for wireless sensor networks. *Computer Networks*, 52(18):3433–3452, 2008. ISSN 1389-1286. doi: 10.1016/j.comnet.2008.09.005.

[129] E.-H. Ngai and I. Rodhe. On providing location privacy for mobile sinks in wireless sensor networks. *Wireless Networks*, 19(1):115–130, 2013. ISSN 1022-0038. doi: 10.1007/s11276-012-0454-z.

[130] X. Niu, C. Wei, W. Feng, and Q. Chen. Osap: Optimal-cluster-based source anonymity protocol in delay-sensitive wireless sensor networks. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2880–2885, April 2014. doi: 10.1109/WCNC.2014.6952906.

[131] X. Niu, Y. Yao, C. Wei, Y. Liu, J. Liu, and X. Chen. A novel source-location anonymity protocol in surveillance systems. In *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*, pages 100–104, October 2015. doi: 10.1109/IIKI.2015.30.

[132] X. Niu, Y. Zhang, Y. Yao, X. Chen, J. M. Jornet, and J. Liu. An energy-efficient source-anonymity protocol in surveillance systems. *Personal and Ubiquitous Computing*, 20(5):771–783, 2016. ISSN 1617-4917. doi: 10.1007/s00779-016-0949-1.

[133] S. Oh and M. Gruteser. Multi-node coordinated jamming for location privacy protection. In *Military Communications Conference (MILCOM), 2011 IEEE*, pages 1243–1249, November 2011. doi: 10.1109/MILCOM.2011.6127471.

[134] J. M. O'Kane. *On the Value of Ignorance: Balancing Tracking and Privacy Using a Two-Bit Sensor*, pages 235–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-00312-7. doi: 10.1007/978-3-642-00312-7_15.

[135] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011. ISSN 0140-3664. doi: 10.1016/j.comcom.2010.05.013. Special Issue of Computer Communications on Information and Future Communication Security.

[136] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, November 2006. doi: 10.1109/LCN.2006.322172.

[137] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon. Entrapping adversaries for source protection in sensor networks. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006*, pages 10–34, 2006. doi: 10.1109/WOWMOM.2006.40.

[138] C. Ozturk, Y. Zhang, and W. 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. ACM, 2004. ISBN 1-58113-972-1. doi: 10.1145/1029102.1029117.

[139] panStamp. panStamp NRG3, 2018. URL `http://www.panstamp.com/product/panstamp-nrg3`. Accessed: 2018-09-17.

[140] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados. Energy-efficient routing protocols in wireless sensor networks: A survey. *IEEE Communications Surveys Tutorials*, 15 (2):551–591, Second 2013. ISSN 1553-877X. doi: 10.1109/SURV.2012.062612.00084.

[141] H. Park, S. Song, B. Y. Choi, and C. T. Huang. Passages: Preserving anonymity of sources and sinks against global eavesdroppers. In *INFOCOM, 2013 Proceedings IEEE*, pages 210–214, April 2013. doi: 10.1109/INFCOM.2013.6566765.

[142] Particle. Electron Datasheet (v005), 2018. URL `https://docs.particle.io/datasheets/electron-(cellular)/electron-datasheet/#recommended-operating-conditions-`. Accessed: 2018-09-17.

[143] Particle. Photon datasheet (v016), 2018. URL `https://docs.particle.io/datasheets/photon-(wifi)/photon-datasheet/#recommended-operating-conditions`. Accessed: 2018-09-17.

[144] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. RFC 3561, July 2003. URL `http://www.rfc-editor.org/rfc/rfc3561.txt`.

[145] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, June 2004. ISSN 0001-0782. doi: 10.1145/990680.990707.

[146] S. Peter, D. Westhoff, and C. Castelluccia. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Transactions on Dependable and Secure Computing*, 7(1):20–34, Jan 2010. ISSN 1545-5971. doi: 10.1109/TDSC.2008.23.

[147] D. Pfammatter, D. Giustiniano, and V. Lenders. A software-defined sensor architecture for large-scale wideband spectrum monitoring. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, pages 71–82. ACM, 2015. ISBN 978-1-4503-3475-4. doi: 10.1145/2737095.2737119.

[148] K. Pongaliur and L. Xiao. Maintaining source privacy under eavesdropping and node compromise attacks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1656–1664, April 2011. doi: 10.1109/INFCOM.2011.5934959.

[149] A. Proaño, L. Lazos, and M. Krunz. Traffic decorrelation techniques for countering a global eavesdropper in WSNs. *IEEE Trans. on Mobile Computing*, 16(3):857–871, March 2017. ISSN 1536-1233. doi: 10.1109/TMC.2016.2573304.

[150] M. Raj, N. Li, D. Liu, M. Wright, and S. K. Das. Using data mules to preserve source location privacy in wireless sensor networks. *Pervasive and Mobile Computing*, 11: 244–260, 2014. ISSN 1574-1192. doi: 10.1016/j.pmcj.2012.10.002.

[151] A. RayChaudhuri, U. K. Chinthala, and A. Bhattacharya. Obfuscating temporal context of sensor data by coalescing at source. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11(2):41–42, April 2007. ISSN 1559-1662. doi: 10.1145/1282221.1282226.

[152] M. A. Razzaque and S. Dobson. Energy-efficient sensing in wireless sensor networks using compressed sensing. *Sensors*, 14(2):2822–2859, 2014. ISSN 1424-8220. doi: 10.3390/s140202822.

[153] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998. ISSN 0733-8716. doi: 10.1109/49.668972.

[154] J. Ren, Y. Li, and T. Li. Routing-based source-location privacy in wireless sensor networks. In *IEEE International Conference on Communications, 2009. ICC '09*, pages 1–5, June 2009. doi: 10.1109/ICC.2009.5199430.

[155] R. Rios and J. Lopez. Analysis of location privacy solutions in wireless sensor networks. *Communications, IET*, 5(17):2518–2532, November 2011. ISSN 1751-8628. doi: 10.1049/iet-com.2010.0825.

[156] R. Rios and J. Lopez. Exploiting context-awareness to enhance source-location privacy in wireless sensor networks. *The Computer Journal*, 54(10):1603–1615, June 2011. doi: 10.1093/comjnl/bxr055.

[157] R. Rios, J. Lopez, and J. Cuellar. *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, chapter Location Privacy in WSNs: Solutions, Challenges, and Future Trends, pages 244–282. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10082-1. doi: 10.1007/978-3-319-10082-1_9.

[158] K. Roussel, Y.-Q. Song, and O. Zendra. Using Cooja for WSN Simulations: Some New Uses and Limits. In K. Roemer, editor, *EWSN 2016 - NextMote workshop*, EWSN 2016 - NextMote workshop, pages 319–324, Graz, Austria, February 2016. ACM, Junction Publishing. URL `https://hal.inria.fr/hal-01240986`.

[159] P. K. Roy, J. P. Singh, P. Kumar, and M. Singh. Source location privacy using fake source and phantom routing (fsapr) technique in wireless sensor networks. *Procedia Computer Science*, 57:936–941, 2015. ISSN 1877-0509. doi: 10.1016/j.procs.2015.07.486. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

[160] K. Sampigethaya, M. Li, L. Huang, and R. Poovendran. Amoeba: Robust location privacy scheme for vanet. *IEEE Journal on Selected Areas in Communications*, 25(8): 1569–1589, Oct 2007. ISSN 0733-8716. doi: 10.1109/JSAC.2007.071007.

[161] A. A. A. Shabaneh, A. M. Ali, C. K. Ng, N. K. Noordin, A. Sali, and M. H. Yaacob. Review of energy conservation using duty cycling schemes for ieee 802.15.4 wireless sensor network (wsn). *Wireless Personal Communications*, 77(1):589–604, Jul 2014. ISSN 1572-834X. doi: 10.1007/s11277-013-1524-y.

[162] H. Shafiei, A. Khonsari, H. Derakhshi, and P. Mousavi. Rate-privacy in wireless sensor networks. *CoRR*, abs/1302.1506, 2013. URL `http://arxiv.org/abs/1302.1506`.

[163] M. Shao, Y. Yang, S. Zhu, and G. Cao. Towards statistically strong source anonymity for sensor networks. In *INFOCOM, 2008 Proceedings IEEE*, April 2008. doi: 10.1109/INFOCOM.2008.19.

[164] M. Shao, W. Hu, S. Zhu, G. Cao, S. Krishnamurth, and T. La Porta. Cross-layer enhanced source location privacy in sensor networks. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '09)*, pages 1–9, 2009. doi: 10.1109/SAHCN.2009.5168923.

[165] R. Shi, M. Goswami, J. Gao, and X. Gu. Is random walk truly memoryless — traffic analysis and source location privacy under random walks. In *INFOCOM, 2013 Proceedings IEEE*, pages 3021–3029, April 2013. doi: 10.1109/INFCOM.2013.6567114.

[166] A. Sieber, R. Karnapke, J. Nolte, and T. Martschei. Using sensor technology to protect an endangered species: A case study. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 1044–1047, Oct 2011. doi: 10.1109/LCN.2011.6115161.

[167] J. P. Singh, P. K. Roy, S. K. Singh, and P. Kumar. Source location privacy using data mules in wireless sensor networks. In *2016 IEEE Region 10 Conference (TENCON)*, pages 2743–2747, November 2016. doi: 10.1109/TENCON.2016.7848539.

[168] P. Spachos, D. Toumpakaris, and D. Hatzinakos. Angle-based dynamic routing scheme for source location privacy in wireless sensor networks. In *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, pages 1–5, May 2014. doi: 10.1109/VTCSpring.2014.7022833.

[169] J. A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, Feb 2014. ISSN 2327-4662. doi: 10.1109/JIOT.2014.2312291.

[170] G. Suarez-Tangil, E. Palomar, B. Ramos, and A. Ribagorda. An experimental comparison of source location privacy methods for power optimization in WSNs. In *Proceedings of the 3rd WSEAS International Conference on Advances in Sensors, Signals and Materials*, SENSIG '10/MATERIALS'10, pages 79–84, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-474-248-6.

[171] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 1–14. ACM, 2008. ISBN 978-1-59593-990-6. doi: 10.1145/1460412.1460414.

[172] H. Sundani, H. Li, V. K. Devabhaktuni, M. Alam, and P. Bhattacharya. Wireless sensor network simulators: A survey and comparisons. *International Journal of Computer Networks*, 2(6):249–265, 2011. ISSN 19854129. URL `http://cscjournals.org/csc/manuscript/Journals/IJCN/volume2/Issue6/IJCN-72.pdf`.

[173] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In H. Karl, A. Wolisz, and A. Willig, editors, *Wireless Sensor Networks*, volume 2920 of *Lecture Notes in Computer Science*, pages 307–322. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-20825-9. doi: 10.1007/978-3-540-24606-0_21.

[174] W. Tan, K. Xu, and D. Wang. An anti-tracking source-location privacy protection protocol in WSNs based on path extension. *Internet of Things Journal, IEEE*, 1(5): 461–471, October 2014. ISSN 2327-4662. doi: 10.1109/JIOT.2014.2346813.

[175] B. Tavli, M. Ozciloglu, and K. Bicakci. Mitigation of compromising privacy by transmission range control in wireless sensor networks. *Communications Letters, IEEE*, 14(12):1104–1106, December 2010. ISSN 1089-7798. doi: 10.1109/LCOMM.2010.101210.100731.

[176] M. Teimouri, U. G. Indahl, H. Sickel, and H. Tveite. Deriving animal movement behaviors using movement parameters extracted from location data. *ISPRS International Journal of Geo-Information*, 7(2), 2018. ISSN 2220-9964. doi: 10.3390/ijgi7020078.

[177] Texas Instruments. CC2420 datasheet. Online. URL `ti.com/lit/ds/symlink/cc2420.pdf`. Accessed: 21/09/2017.

[178] Texas Instruments. MSP430 SoC With RF Core, 2016. URL `https://www1.iodparts.com/datasheets/texas-rf-transceiver-ics-cc430f5137.pdf`. Accessed: 2018-09-17.

[179] A. Thomason, M. Leeke, M. Bradbury, and A. Jhumka. Evaluating the impact of broadcast rates and collisions on fake source protocols for source location privacy. In *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 667–674, July 2013. doi: 10.1109/TrustCom.2013.81.

[180] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 477–482, April 2005. doi: 10.1109/IPSN.2005.1440978.

[181] Y. Tscha. Routing for enhancing source-location privacy in wireless sensor networks of multiple assets. *Communications and Networks, Journal of*, 11(6):589–598, December 2009. ISSN 1229-2370. doi: 10.1109/JCN.2009.6388412.

[182] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-20-2.

[183] H. Wang, B. Sheng, and Q. Li. Privacy-aware routing in sensor networks. *Computer Networks*, 53(9):1512–1529, 2009. ISSN 1389-1286. doi: 10.1016/j.comnet.2009.02.002.

[184] J. R. Ward and M. Younis. On the use of distributed relays to increase base station anonymity in wireless sensor networks. In *Military Communications Conference (MILCOM), 2012 IEEE*, pages 1–6, October 2012. doi: 10.1109/MILCOM.2012.6415575.

[185] J. R. Ward and M. Younis. On the effect of imperfect range estimates on base station anonymity in wireless sensor networks. In *Military Communications Conference (MILCOM), 2013 IEEE*, pages 605–610, November 2013. doi: 10.1109/MILCOM.2013.109.

[186] J. R. Ward and M. Younis. Examining the effect of wireless sensor network synchronization on base station anonymity. In *Military Communications Conference (MILCOM), 2014 IEEE*, pages 204–209, October 2014. doi: 10.1109/MILCOM.2014.39.

[187] J. R. Ward and M. Younis. A metric for evaluating base station anonymity in acknowledgement-based wireless sensor networks. In *Military Communications Conference (MILCOM), 2014 IEEE*, pages 216–221, October 2014. doi: 10.1109/MILCOM.2014.41.

[188] W. Wei-Ping, C. Liang, and W. Jian-xin. A source-location privacy protocol in WSN based on locational angle. In *IEEE International Conference on Communications (ICC)*, pages 1630–1634, May 2008. doi: 10.1109/ICC.2008.315.

[189] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7^{th} symposium on Operating systems design and implementation*, OSDI '06, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association. ISBN 1-931971-47-1.

[190] Y.-C. Wu, Q. Chaudhari, and E. Serpedin. Clock synchronization of wireless sensor networks. *Signal Processing Magazine, IEEE*, 28(1):124–138, January 2011. ISSN 1053-5888. doi: 10.1109/MSP.2010.938757.

[191] WWF. Wildlife crime technology project. Online, 2012–2016. URL `http://worldwildlife.org/projects/wildlife-crime-technology-project`. Accessed: 2016-06-03.

[192] Y. Xi, L. Schwiebert, and W. Shi. Preserving source location privacy in monitoring-based wireless sensor networks. In *20th International Parallel and Distributed Processing Symposium*, pages 1–8, April 2006. doi: 10.1109/IPDPS.2006.1639682.

[193] Y. Xu, G. Chen, J. Ford, and F. Makedon. Detecting wormhole attacks in wireless sensor networks. In E. Goetz and S. Shenoi, editors, *Critical Infrastructure Protection*, pages 267–279, Boston, MA, 2008. Springer US. ISBN 978-0-387-75462-8.

[194] W. Yang and W. Zhu. Protecting source location privacy in wireless sensor networks with data aggregation. In Z. Yu, R. Liscano, G. Chen, D. Zhang, and X. Zhou, editors, *Ubiquitous Intelligence and Computing*, volume 6406 of *Lecture Notes in Computer Science*, pages 252–266. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-16354-8. doi: 10.1007/978-3-642-16355-5_22.

[195] Y. Yang, M. Shao, S. Zhu, and G. Cao. Towards statistically strong source anonymity for sensor networks. *ACM Trans. Sen. Netw.*, 9(3):34:1–34:23, June 2013. ISSN 1550-4859. doi: 10.1145/2480730.2480737.

[196] L. Yao, L. Kang, F. Deng, J. Deng, and G. Wu. Protecting source–location privacy based on multirings in wireless sensor networks. *Concurrency and Computation: Practice and Experience*, 27(15):3863–3876, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3075.

[197] Zolertia. Z1 Datasheet, 2010. URL `http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf`. Accessed: 2018-09-17.

[198] M. Zuniga. Building a network topology for tossim. Online, 2006. URL `https://htmlpreview.github.io/?https://github.com/tinyos/tinyos-main/blob/master/doc/html/tutorial/usc-topologies.html`. Accessed: 28/09/2017.