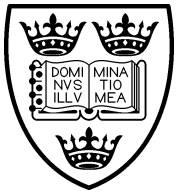
Delay-Tolerant Data Collection in Sensor Networks with Mobile Sinks



Felix Ricklef Scriven Wohlers

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2012

Abstract

Collecting data from sensor nodes to designated sinks is a common and challenging task in a wide variety of wireless sensor network (WSN) applications, ranging from animal monitoring to security surveillance. A number of approaches exploiting sink mobility have been proposed in recent years: some are proactive, in that sensor nodes push their readings to storage nodes from where they are collected by roaming mobile sinks, whereas others are reactive, in that mobile sinks pull readings from nearby sensor nodes as they traverse the sensor network.

In this thesis, we point out that deciding which data collection approach is more energy-efficient depends on application characteristics, including the mobility patterns of sinks and the desired latency of collected data. We introduce novel adaptive data collection schemes that are able to automatically adjust to changing sink visiting patterns or data requirements, thereby significantly easing the deployment of a WSN.

We illustrate cases where combining proactive and reactive modes of data collection is particularly beneficial. This motivates the design of TwinRoute, a novel hybrid algorithm that can flexibly mix the two collection modes at appropriate levels depending on the application scenario. Our extensive experimental evaluation, which uses synthetic and real-world sink traces, allows us to identify scenario characteristics that suit proactive, reactive or hybrid data collection schemes. It shows that TwinRoute outperforms the pure approaches in most scenarios, achieving desirable tradeoffs between communication cost and timely delivery of sensor data.

Acknowledgements

There are a few people that I'd like to thank for their continued support throughout my DPhil, without whom this thesis would have not been completed. The briefness in doing so is unrepresentative of the gratitude I have towards them.

First and foremost, I'd like to thank my supervisor Niki Trigoni for teaching me how to think in a structured manner.

From my research group, I'd like to thank Andrew Markham, Muzammil Hussain and Julian de Hoog for all the professional and personal support.

The bunch of Keble MCR folk who made my life ever more interesting and enjoyable.

Finally, I'd like to thank my parents, Jeannie and Wohlert, for ensuring that I prevailed in completing the thesis.

Thank you all.

Contents

1 Introduction **9**

1.1 Problem Definition 13
1.1.1 Assumptions 13
1.1.2 Objectives 14
1.2 Research Questions 16
1.3 Main Contributions 16
1.4 Publications 17
1.5 Summary 18

2 Literature Review **20**

2.1 MANET Routing Algorithms 20
2.1.1 Synchronous MANET Routing Algorithms 21
2.1.2 Asynchronous MANET Routing Algorithms 22
2.2 WSN Routing Algorithms 25
2.2.1 WSN Routing Algorithms with Fixed Nodes 25
2.2.2 WSN Routing Algorithms for Fixed Nodes and Mobile Sinks . 27
2.3 Mobility Models 33
2.3.1 Random Models 34
2.3.2 Models with Temporal Dependency 35
2.3.3 Models with Spatial Dependency and Geographic Restrictions 36
2.4 Summary 37

3 Static Routing Protocols **39**

3.1 Proactive Algorithms 40
3.1.1 P-scheme Algorithm 40
3.1.2 Algorithm Behaviour 41
3.1.2.1 Experimental Setup 42

3.1.2.2 Parameter Sensitivity Analysis . . . 43
3.1.2.3 Algorithm Trade-off . . . 44
3.2 Reactive Algorithms . . . 46
3.2.1 R-scheme Algorithm . . . 47
3.2.2 Algorithm Behaviour . . . 49
3.2.2.1 Parameter Sensitivity Analysis . . . 49
3.2.2.2 Algorithm Trade-off . . . 51
3.3 Summary . . . 55

4 Adaptive Routing Protocols **57**
4.1 Adaptive P-scheme . . . 57
4.1.1 Algorithm . . . 59
4.1.2 Parameter Sensitivity Analysis . . . 64
4.2 Adaptive R-scheme . . . 67
4.2.1 Algorithm . . . 68
4.2.2 Parameter Sensitivity Analysis . . . 74
4.2.3 Implementation Details . . . 80
4.3 Summary . . . 81

5 Hybrid Routing Protocol (TwinRoute) **83**
5.1 TwinRoute Algorithm . . . 84
5.2 Algorithm Parameter Sensitivity Analysis . . . 87
5.3 Summary . . . 90

6 Implementation and Setup **92**
6.1 WSN Operating System . . . 92
6.1.1 WSN Simulator . . . 93
6.1.2 Contiki's uIP IPv6 Network Stack . . . 94
6.1.3 MAC-Layer Protocols . . . 96
6.2 Experimental Setup . . . 97
6.2.1 Cooja Setup . . . 97
6.2.2 Static Node Setup . . . 98
6.2.3 Mobile Sink Setup . . . 99
6.3 Routing Algorithms' Implementation Architecture . . . 99
6.3.1 Routing Module . . . 100
6.3.2 Data Transfer Module . . . 102
6.4 Summary . . . 103

7 Performance Evaluation **104**
7.1 Adaptive and Static Algorithms 104
7.1.1 P-scheme 104
7.1.1.1 Deployment Procedures 105
7.1.1.2 Performance Evaluation 106
7.1.1.3 Summary 110
7.1.2 R-scheme 110
7.1.2.1 Deployment Procedures 110
7.1.2.2 Performance Evaluation 111
7.1.2.3 Summary 114
7.1.3 Summary 114
7.2 Performance of Adaptive Algorithms in Different Scenarios 115
7.2.1 Methodology 116
7.2.1.1 Sink Mobility Analysis 116
7.2.1.2 Algorithm Behaviour 117
7.2.1.3 Experimental Results 119
7.2.2 Farm Maintenance Model 120
7.2.2.1 Sink Mobility Analysis 120
7.2.2.2 Algorithm Behaviour 121
7.2.2.3 Experimental Results 123
7.2.2.4 Summary 125
7.2.3 Town Suburb 125
7.2.3.1 Sink Mobility Analysis 125
7.2.3.2 Algorithm Behaviour 128
7.2.3.3 Experimental Results 129
7.2.3.4 Summary 131
7.2.4 Town Centre 131
7.2.4.1 Scenario Analysis 132
7.2.4.2 Algorithm Behaviour 133
7.2.4.3 Experimental Results 134
7.2.4.4 Summary 136
7.2.5 Ocean Monitoring 137
7.2.5.1 Scenario Analysis 137
7.2.5.2 Algorithm Behaviour 138
7.2.5.3 Experimental Results 139
7.2.5.4 Summary 141

7.2.6 National Reserve 141
7.2.6.1 Scenario Analysis 141
7.2.6.2 Algorithm Behaviour 143
7.2.6.3 Experimental Results 144
7.2.6.4 Summary 146
7.2.7 Wytham Woods - National Park 146
7.2.7.1 Scenario Analysis 147
7.2.7.2 Algorithm Behaviour 148
7.2.7.3 Experimental Results 149
7.2.7.4 Summary 151
7.2.8 Scenario Performance Classification 151
7.3 Summary 154

8 Conclusions **156**
8.1 Contributions 157
8.1.1 Adaptive Data Collection 158
8.1.2 Hybrid Data Collection 159
8.1.3 Scenario Classification and Adaptive Algorithm Performance 159
8.2 Future Work 160
8.3 Summary 161

Chapter 1

Introduction

Current technological advances have allowed wireless sensor networks (WSNs) to be employed in a large range of applications, ranging from wildlife monitoring to security and surveillance systems. Typically, numerous battery-powered sensor nodes are deployed in an area to monitor ambient conditions and transfer data in a wireless multi-hop manner to designated sink nodes. In many WSN scenarios, sensor nodes are static nodes placed near points of interest and left to operate unattended for a long period of time, whereas sink nodes are rechargeable mobile devices held by application users, vehicles or animals, which move within the borders of the monitored area (see Figure 1.1).

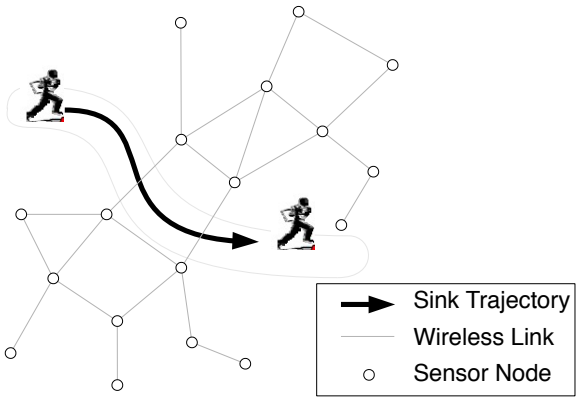


Figure 1.1: Example WSN configuration in which a mobile sink roams the network to collect data from fixed sensor nodes.

Sensor nodes have restricted processing, storage and communication capabilities and are majorly restricted by the energy available from their battery. Since a significant proportion of their energy is consumed in wireless communication, it is vital to minimise the number of packet transmissions involved in delivering data from fixed sensors to mobile sinks. Mobile sinks, on the other hand, are assumed to be rechargeable nodes with abundant energy, storage and processing resources. They can wirelessly communicate with static sensor nodes within range to collect data from the network. Their movement is not controlled, and they often do not know their location within the network.

A plethora of data collection schemes has been proposed in the literature to enable energy-efficient delivery of data from fixed sensors to mobile sinks. Typically, these schemes are divided into two main classes: proactive and reactive. In proactive schemes, sensor nodes proactively forward their data to carefully selected storage nodes, from where data is collected opportunistically by roaming mobile sinks. Proactively pushing the data towards the destinations incurs a small routing overhead, however it is less flexible and adaptable to changing data needs or sink visiting patterns. In reactive schemes, sensor nodes react to sink-initiated probes for data collection. Sinks announce their presence by sending query messages to nearby sensors, which react by returning sensor data pertinent to the query. Reactive routing can deal well with changing sink visiting patterns due to its on-demand data forwarding approach.

Although a lot of previous work has focused on the design of energy-efficient routing algorithms for WSNs, little attention has been paid to the problem of optimising algorithms according to the characteristics and requirements of a specific application. This is an important problem since WSN applications differ in several respects, as described below:

First, different applications typically feature very different sink mobility patterns.

Some scenarios involve only mobile sinks, whereas others feature a mixture of mobile and fixed sinks; mobile sinks may follow completely random trajectories, or repeatedly take predictable paths. Furthermore, mobile sink visiting patterns vary greatly in the frequency and distribution of visits, with some trajectories visiting few areas very frequently while others spread their visiting pattern over a greater area with less frequent visits.

Second, applications vary significantly in their requirements for data latency - i.e. the time between data generation at a sensor node and delivery to a user. Emergency or intrusion detection applications require sensor events to be delivered within a few seconds from their detection, whereas traffic monitoring applications can tolerate delivery delays of a few minutes. Variations in desirable data latency are often observed within the scope of the same application. For example, in a wildlife monitoring application, an animal movement event may need to be reported immediately in order to activate a nearby camera and capture its behaviour; delays of several minutes can be tolerated to activate environmental sensors to capture micro-climatic conditions in the animals vicinity; delays of hours are tolerated for monitoring long-term animal behaviour.

Third, applications may vary in their delivery ratio requirements, that is, the percentage of data that is actually collected within the specified latency threshold. For example, in a wildlife monitoring application, 90% of animal sightings must be reported within seconds to activate nearby cameras, whereas in an intrusion detection system, 100% of people detection events must be collected within seconds.

It becomes apparent that application scenarios vary significantly and it is far from trivial to select and configure a data collection algorithm that fits a particular scenario. Assuming the main deployment variables are known, how does a network engineer choose the right type of data collection algorithm? Do proactive or reactive approaches suit a given scenario such that the application requirements

are met? Furthermore, how should the data collection scheme be tuned in order to maximise the lifetime of the sensor network? These are the main questions this thesis is going to address.

More specifically, the high-level goal of this work is to propose adaptive algorithms that can self-configure to fit the characteristics of an application scenario (i.e. sink mobility and latency/delivery ratio requirements), and can adapt to dynamic changes in these characteristics. This problem is particularly challenging because of the following reasons:

- WSN applications vary widely in sink mobility patterns and their requirements for timely data collection.
- Some WSN applications may benefit from combining more than one approach for data collection, thus suggesting the need for designing hybrid proactive-reactive algorithms.
- Application characteristics such as sink mobility or time constraints for data delivery may change during the lifetime of a WSN.

This work is part of the WildSensing project, which is a collaborative effort between researchers at the University of Oxford and the University of Cambridge. Within the scope of WildSensing, a wireless sensor network has been deployed at Oxford University's research reserve in Wytham Woods to monitor environmental conditions. This deployment provides the unique opportunity to extract real sink mobility traces that we could use in our study, for example traces of foresters roaming through the woods for their daily tasks.

The remainder of this section is organised as follows: Section 1.1 presents the assumptions and objectives of the thesis. Section 1.2 presents the research questions and Section 1.3 outlines the main contributions of the thesis. Finally, Section 1.4 describes the publications related to the thesis.



Figure 1.2: A TmoteSky sensor node with USB-connection.

1.1 Problem Definition

In this section, we define the problem under investigation, starting with the model assumptions and continuing with the specific objectives for efficient data collection.

1.1.1 Assumptions

In this section, we discuss the main assumptions about the WSN, which consists of fixed sensor nodes and mobile sinks. We also discuss the application requirements for the data delivery from the sensors to the mobile sinks.

Fixed sensors: Static sensor nodes are typically battery-powered and therefore energy-constrained. They sense the environment and generate data that needs to be collected by one of the mobile sinks. A popular platform of fixed sensor nodes is the Tmote Sky platform [36] (see Figure 1.2). It features a 8MHz TI MSP 430 micro-controller (10kb RAM, 48kb of Flash), a 250kbps IEEE 802.15.4 wireless transceiver that has a stated range of 50m (indoors) to 125m (outdoors). The Tmote Sky uses two AA batteries as power supply and supports TinyOS or contiki as off-the-shelf operating system. We assume that the storage restrictions of the sensor nodes are of secondary importance as the Tmote Sky sensor boards can easily be extended with cheap and energy efficient flash storage that can provide gigabytes of storage memory.

Mobile Sinks: We assume that mobile sinks are not energy-constrained as

their batteries are regularly recharged by mobile users, example devices being PDAs, laptops or mobile phones. Once data arrives at a mobile sink, it becomes available to all application users through a backbone network, e.g. through a 3G or 4G connection from the mobile sinks to the end-user application. Mobile sinks can wirelessly communicate with nearby fixed sensors, within a distance determined by their communication range. We do not assume control over the mobile sinks' movement. As a result, our work is applicable to a potentially much wider set of applications than those targeted by previous work [31, 14, 29, 47]. The various mobility patterns of sinks also pose a challenge as the sink mobility may be a highly predictable bus route or less predictable animal movement.

Application Requirements: We consider WSN applications in which mobile sinks collect data from fixed sensor nodes. An important non-functional requirement of WSN applications is to extend the network's lifetime. In addition, application time constraints for timely data delivery are captured by two distinct metrics. Firstly, a data latency threshold during which the data is of value to the end-user (e.g. $latency$=50 seconds). Secondly, a delivery ratio DR of data collected within the latency bounds, e.g. $DR = 90\%$ suggests that 90% of every node's data readings need to be collected within the latency requirement. Applications such as security monitoring are inclined to have stringent requirements for both data latency as well as delivery ratio whereas a farm monitoring application may impose more relaxed constraints on data latency and delivery ratio.

1.1.2 Objectives

The overall aim of the D.Phil thesis is to relieve network engineers from having to carefully select a data collection algorithm and tune it for a particular application. The main objective is to automate this task by designing energy-efficient data collection algorithms that can adjust their behaviour to the needs and characteristics

of the application scenario. We measure energy-efficiency of data collection algorithms using the radio duty cycle, namely the percentage of time the radio is turned on for communication during which it consumes energy. The radio duty cycle is a hardware independent way of measuring the communication cost, in contrast to directly measuring the radio energy usage that tie results to a particular hardware platform.

Since the communication cost typically dominates the energy expenditure, our goal is to identify distributed algorithms that minimise the number of packet transmissions within the fixed sensor network. Furthermore, we want to maximise the time during which the entire wireless sensor network (WSN) remains operational and does not suffer from any sensor node failures due to energy depletion. We plan to use two metrics to measure communication efficiency that reflect the previous two observations, respectively:

Average radio duty cycle of the fixed sensors. It reflects the overall energy expenditure of the algorithm.

Maximum radio duty cycle of any fixed sensor node. Spreading the communication load prevents the so-called “hotspot problem”, in which a small number of sensor nodes are burdened with a large amount of data routing. A high routing load can lead to premature energy depletion and potentially to a partial failure of the WSN.

In short, our objective is to design data collection algorithms that adhere to application requirements for latency and delivery ratio, while minimising the average (or maximum) energy consumption in the fixed sensor network.

1.2 Research Questions

Identifying energy-efficient algorithms that suit a particular WSN application is far from trivial, and it involves addressing three distinct research questions:

- How do we design adaptive routing algorithms that can minimise energy usage for a large range of scenarios, e.g. with differing sink mobility or data latency requirements?
- Proactive and reactive algorithms have strengths and weaknesses for given scenarios. Can we save energy by combining aspects of proactive and reactive algorithms?
- What WSN scenarios suit proactive or reactive data collection schemes? Are there scenarios that would particularly benefit from a hybrid data collection scheme?

The thesis explores these three research questions in depth, and contributes novel approaches to data collection as described in the next subsection.

1.3 Main Contributions

The main contributions of this thesis are threefold.

Firstly, we describe the design, implementation and configuration of *adaptive data collection algorithms*, which can self-configure to fit the characteristics of a particular application. Autonomous configuration of data collection algorithms relieves network engineers from discovering sink mobility patterns and carefully tuning the parameters of the data collection algorithm to satisfy latency and delivery ratio requirements. We show that the adaptive data collection algorithms

are comparable to optimal manual configurations in static network conditions, and outperform manual configurations in dynamic network conditions.

Secondly, we describe the design, implementation and configuration of a *hybrid data collection algorithm* that combines aspects of both reactive and proactive schemes. We highlight cases where the hybrid algorithm can outperform both proactive and reactive schemes in terms of communication cost as well as achievable latency.

Finally, we analyse a range of different scenarios, and identify which adaptive (pure and hybrid) algorithm best fits which scenario characteristics. Furthermore, we show how hybrid schemes can adapt to a much larger range of scenarios while performing nearly as well as or better than the pure algorithms.

1.4 Publications

During the course of working on this thesis three peer reviewed publications were produced. The contributions of the author to each of these as well as their connection to the thesis will be described in more detail.

”TwinRoute: Energy-Efficient Data Collection in Fixed Sensor Networks with Mobile Sinks” [48]: In this paper we describe the novel notion and advantage of a hybrid data collection scheme which laid the foundation of this thesis. The hybrid TwinRoute data collection algorithm combines elements of existing proactive and reactive algorithms. We show that the hybrid TwinRoute algorithm outperforms or performs at least as well as the pure data collection algorithms. This is achieved by the flexibility of the hybrid algorithm that can act like a pure data collection scheme when beneficial, or take advantage of the hybrid option when the application scenario permits. Contributions include the key idea of combining proactive and reactive data collection into a single algorithm, all the experiments of the performance

evaluation, and the writing of the main results.

”Evolution and Sustainability of a Wildlife Monitoring Sensor Network” [46]: The system design, implementation and iteration of the WildSensing system are described in this paper. The personal contributions lie in the design, implementation and deployment of a novel proactive data collection scheme that resembles an early version of the proactive routing algorithm used in this thesis. Unlike other data collection schemes, the proactive scheme allowed data to be assigned a priority that resulted in high-priority data using more energetically expensive routes (e.g. through a 3G gateway) while low priority data utilised energy efficient routes when possible. The deployment showed that the novel approach increased the time-to-failure by 57%. Due to time constraints, the priority based routing was not incorporated into the thesis, though the proactive routing algorithm and use of the IPv6 networking stack laid the technical groundwork for the implementation used for this thesis.

”Evolution and Sustainability of a Wildlife Monitoring Sensor Network: the WildSensing System.” [45]: This journal paper is an extended version of the previous paper that was voted as one of the top 3 papers of the SenSys conference. The paper describes in more detail the usage and implementation details of the IPv6/6LoWPAN standard, while highlighting the current shortcomings thereof for a WSN environment.

1.5 Summary

In this chapter we outlined the problems faced by network engineers when deploying a delay-tolerant WSN with fixed sensors and mobile sinks. We outlined the application requirements that we assume are applied to the WSN as well as our objectives of finding a suitable data collection scheme that minimises the average

and maximum radio duty cycle of the network. This is not a trivial question for multiple reasons, such as making the fundamental choice between proactive and reactive routing algorithms. Furthermore, can we combine aspects of proactive and reactive data collection schemes in order to save energy within the WSN?

The contributions of this thesis lie, firstly, in the description of adaptive data collection schemes, secondly, the design of hybrid data collection algorithms and, finally, the experimental evaluation and comparison of data collection schemes.

The thesis is organised as follows. Chapter 2 describes the existing literature for delay tolerant data collection within WSNs. In Chapter 3 we present two example implementations of existing proactive and reactive data collection schemes, that we refer to as P- and R-scheme. Chapter 4 outlines and analyses novel adaptive versions of the P- and R-scheme. Chapter 5 describes a novel hybrid routing algorithm that can adapt to a large range of scenarios. Following this, Chapter 6 details the software implementation. The experimental analysis is conducted in Chapter 7, followed by a conclusion and ideas for future work in Chapter 8.

Chapter 2

Literature Review

The aim of the thesis is to devise energy efficient routing protocols for data collection in fixed WSNs with mobile sinks. To do so, we draw inspiration from existing routing protocols for mobile ad-hoc networks (MANETs) and WSNs. In this section, we summarise these protocols, and point out how they differ from the protocols proposed in the thesis.

The literature review is split into three broad sub-sections. Firstly, routing algorithms for MANETs are described in Section 2.1. Secondly, routing algorithms for wireless sensor networks (WSNs) are described in Section 2.2. Lastly, mobility models and mobility metrics are discussed in Section 2.3.

2.1 MANET Routing Algorithms

Wireless sensor networks (WSNs) differ in many aspects from the closely related mobile ad-hoc networks (MANETs). The main difference lies within the less resource constrained nature of MANETs, MANETs often being based on highly capable computing devices such as laptops. WSNs on the other hand often have severe processing, communication, storage and energy constraints and require a

delicate allocation of their limited resources. The mode of communication also differs substantially: MANETs often concentrate on one-to-one communication between individual nodes, WSNs typically incur many-to-one communication in which the sensed data is relayed to a single sink. Furthermore, the data interests differ conceptually: MANETs usually focus on address-centric data transfer similar to point-to-point internet communication, WSNs transfer data on a data-centric basis, that is, they transfer data that satisfy specific user interests. In the remainder of this section, we outline the most popular routing algorithms for MANETs, which later on inspired the design of routing algorithms for WSNs.

2.1.1 Synchronous MANET Routing Algorithms

The Destination-Sequenced Distance Vector routing algorithm (DSDV) [38] is a basic Bellman-Ford routing algorithm modified for ad-hoc mobility. A detailed routing table is maintained at every node that is updated by exchanging routing information with neighbouring nodes. Routing table updates are conducted at a higher frequency when movement is detected in the form of link breakage whilst sequence numbers of routing table updates are used to prevent loops after topological changes.

Perkins et al. formulated the Ad-hoc On-Demand Distance Vector Routing protocol (AODV) [39] that also maintains detailed routing tables at nodes, however unlike DSDV the routing tables are only updated upon demand. When a data demand is initiated, routes are established by flooding the network with route requests until a route is found.

Unlike the previous two routing algorithms that use routing tables, the Dynamic Source Routing (DSR) [20] approach sets the exact routing path at the source node before data is sent, the path including all the individual hops. The source node floods the network with route requests to determine the path to which the

destination responds with a route reply using the reverse path, thus not requiring a detailed routing table at individual sensor nodes. DSR minimises the route request flooding by making use of routing caches and by overhearing by-passing packages. Faulty routes incurred due to caching errors or topological changes are removed using explicit route error packages.

Johansson et al. [19] compare the performance of DSDV, AODV and DSR in different employment scenarios and find that the reactive protocols, namely AODV and DSR, perform significantly better than the proactive DSDV routing algorithm that suffers from the large overhead incurred by the routing table exchanges. AODV is found to perform robustly in scenarios with high node mobility, whilst the DSR algorithm incurs a smaller overhead. Generally however, both AODV and DSR are found to perform similarly in the investigated scenarios.

Bai et al. [6] also analyse the performance of DSDV, AODV and DSR under differing mobility models (more details in Section 2.3.3). Similarly to Johansson et al., they find that the reactive protocols (AODV and DSR) deal better with increased mobility with DSR requiring a smaller overhead. However, for stable connections between the sensor nodes DSDV can achieve a higher throughput than the reactive protocols due to the proactive behaviour requiring a smaller routing overhead. In addition to this, Bai et al. state that a single performance favourite cannot be identified given different mobility models, thus implying the notion of dependability between algorithm performance and mobility model.

2.1.2 Asynchronous MANET Routing Algorithms

The previously described popular routing algorithms use connected end-to-end routes to transfer a single data copy from the source to the destination. Unlike the routing algorithms in the previous section, the herein presented routing algorithms accept temporary disconnections in the routing path and utilise mobility amongst

the nodes to reach the destination. More specifically, the data collection algorithms distribute multiple copies of the packages amongst their moving neighbours. The multiple copies in turn increase the likelihood of encountering the data destination. The routing algorithms that use this strategy, from hereon referred to as replicate strategy, also do not assume highly restricted resources and therefore also represent a class of MANET routing algorithms. All the presented algorithms assume high mobility for all nodes, again making the routing algorithms not directly applicable to our WSN scenario.

Vahdat et al. [44] investigate epidemic routing for partially connected ad-hoc networks. The protocol simply distributes multiple copies of the data packages to every newly encountered sensor node, relying on transitive asynchronous distribution to reach the data destination. The resource consumption of the algorithm can be restricted by a maximum hop-count per message or by imposing buffer space limitations. When a sensor node runs out of buffer space, data is deleted by a FIFO selection process. The routing protocol performs excellently (100% message delivery) when the resources are hardly or not at all restricted. With restricted resources the algorithm is able to reduce the energy load, however the algorithm performance also takes a substantial loss. Despite being able to guarantee a high message delivery rate, the epidemic routing protocol does not effectively control the number of data transmissions and therefore resource usage. Furthermore, successfully delivered messages are not removed from the network as the hash table used to exchange message information does not contain data removal instructions. The resource wastage can be assumed to be high, which in energy restricted wireless sensor networks is crucial for a successful employment.

Whilst the epidemic routing algorithm naively distributes data copies, the prophet routing algorithm [26] uses probabilistic information based on the history of node encounters to distribute the copies. Every node computes its own forwarding prob-

ability for a destination, the forwarding probability being high if the data destination was encountered frequently or other nodes were met that have a high transitive destination encountering probability. Ageing is used to re-adjust the forwarding probability to changing mobility patterns. Similarly to the epidemic routing algorithm, a FIFO data deletion process is used to free up space when the buffer is full. prophet outperforms the epidemic routing algorithm in most of the tested scenarios, the simulations using only very small buffer sizes. Restricting the buffer size in the simulation gives prophet a great advantage as its data forwarding policy is more selective than in epidemic routing, thereby requiring less buffer space. The major restriction of prophet is that it has no effective control over the number of copies and therefore in effect the resources used within the wireless sensor network.

Unlike the previous two replicate routing approaches, the Spray and Wait routing algorithm [42] investigates controlled flooding techniques such that the number of copies within the WSN is restricted. The routing algorithm consists of two phases. In the initial spray phase a number of data copies are naively distributed amongst encountered sensor nodes, early encountered sensor nodes being allocated a larger share of copies for further distribution. Once all copies are distributed, the wait phase is used to transmit data to the final destination. Spray and Wait is found to outperform all compared schemes with respect to the number of transmissions and delivery delay as the controlled flooding contains potential contention within the WSN as found in other flooding based approaches. However, by only using the random waypoint and random direction mobility models it inherently only exposes and compares its algorithm to mobility with high encounter likelihood.

Instead of naively distributing the replicate copies as in Spray and Wait, the SCAR routing protocol [35] exploits movement and resource predictions to smartly forward a restricted number of data copies towards the destination. Initially, multiple copies are forwarded to neighbouring nodes with the highest data delivery

likelihoods to increase the reliability of reaching the data destination. The data delivery likelihood is based on relevant factors such as connectivity change, sink collocation or sensor node survivability (or battery level). Kalman filters are then used to smooth the predictions of the data delivery likelihood functions. Out of the number of copies created, one designated master copy is given higher priority in order to increase the delivery likelihood of at least one copy when sensor node failures are common. Replicate copies are removed from the network through neighbourhood notifications of successful data delivery, reducing the storage load within the WSN.

In this section we described MANET routing algorithm approaches that assume low resource restrictions and address centric communication. MANET data collection algorithms are of interest as they also focus on delay tolerant data collection, however unlike this thesis they assume that all nodes are mobile. The following sections will give an insight into routing algorithms for WSNs in which all nodes are resource-constrained.

2.2 WSN Routing Algorithms

Routing algorithms in WSNs are designed with highly resource constrained sensor nodes in mind. In Section 2.2.1, we describe WSN routing algorithms in which all sensor nodes are assumed to be stationary, such that data is sent from static sensor nodes to static sinks. In Section 2.2.2, we discuss WSN routing algorithms that assume that data is collected from static sensor nodes by mobile sinks.

2.2.1 WSN Routing Algorithms with Fixed Nodes

The first two algorithms are tree-based routing algorithms that upon request reactively establish a routing tree along which data is transmitted. Tree-based routing

algorithms use beacon broadcasts that are initiated by the sink to establish routing trees. Nodes neighbouring the sink register the beacon and set the beacon sender as their parent in the routing tree. Conversely, the neighbouring nodes themselves send another beacon to their neighbouring nodes, who in turn initiate another beacon broadcast until eventually the entire network is reached by broadcasts. Loops within the tree can easily be prevented by including the tree's hop-count in the beacon, thus only accepting a node as parent if the hop-count is smaller.

One such tree-based routing algorithm is the Directed Diffusion routing algorithm [16] in which sinks announce data-centric interests within the WSN. A sink propagates a data interest by flooding or restrictively flooding the network (as described previously), the flooding also being used to establish tree-based routes between the source and the sink. If a sensor node satisfies the data needs, it starts propagating its information along all the established paths at a low data rate. Efficient paths are then reinforced by the sink by requesting the same data at a sufficiently high data rate.

The Tiny AGgregation Service (TAG) [33] also floods a network with SQL like requests in order to build tree-based routing paths. The provided SQL statements allow accessing information from the WSN in a similar fashion to traditional relational databases. Instead of using a server-based approach in which all data is naively centrally gathered, TAG utilises in-network processing along the routing-tree to reduce the communication required to answer the SQL requests. By partially processing information in the routing tree from leaf to root node, the number of messages is reduced to one per node for aggregate SQL operators (e.g. average, sum, min, max etc.).

To avoid flooding the network as in the previous tree-based approaches, GPSR [21] uses location awareness (e.g. GPS) to reach its target. Two data forwarding approaches are used to reach a destination: greedy and perimeter forwarding.

The greedy forwarding approach sends data to neighbours that are geographically closest to the destination, thereby reducing the number of hops required. If a region is reached in which the greedy forwarding fails to find a closer neighbour, the perimeter forwarding approach will circumnavigate the disjoint area until greedy forwarding can be re-applied. GPSR's advantages are its simplicity and reliance on local data knowledge only. However, location aware technology, such as GPS, is energetically expensive and not assumed to be available in our WSN setup.

A major problem with using only stationary nodes is that areas close to points of interests are likely to have to burden a higher routing load due to the passing traffic. As the nodes are static, the higher load will lead to an imbalanced use of resources, potentially leading to premature resource depletion at these so called hotspots. A relatively novel way of balancing the routing load is by using mobile sinks that spread the resource usage throughout the entire network. The following section describes this setup in which static sensor nodes generate data that is collected by mobile sinks.

2.2.2 WSN Routing Algorithms for Fixed Nodes and Mobile Sinks

Utilising the movement of sinks was seen as an additional obstacle in routing algorithms. However, it has become apparent that the load-balancing feature of mobile sinks can be used to elongate the network lifetime and survivability [30, 17, 8]. This section highlights algorithms that were designed for WSNs with static sensor nodes and mobile sinks. The algorithms are presented in two sub-sections according to their behaviour. At first, reactive algorithms are presented. Reactive algorithms only transmit data "in reaction" to an advertised data interest by the sink. Secondly, proactive routing algorithms are presented. Proactive routing algorithms forward data such that each node proactively pushes data towards the best path available.

Reactive Data Collection Algorithms Most tree-based algorithms are reactive algorithms as data is reactively sent along the routing tree after a data interest has been flooded across the network. In fact, the tree-based algorithms for static sinks (Directed Diffusion, TAG) can be easily adapted to WSNs with mobile sinks by simply updating the tree-routes more frequently according to the mobile sink movement.

Instead of frequently updating the tree-routes, Akkaya et al. [3] describe techniques to extend such tree-based routes such that no extensive routing updates are required. A cost metric based on hop-count is used to decide whether the existing path should be extended or a new route should be found. The routing path can be extended in two ways: either the wireless range is extended by using higher power radio levels or the routing path is simply extended using more forwarding nodes. If both wireless range or routing path extension are deemed too expensive, the node topology and routing paths are re-established using the underlying routing protocol. Unlike Akkaya et al., we assume no control over the radio range of the nodes, rendering half of the route extending approach inapplicable to our scenario. Furthermore, the underlying tree-based routing algorithm does not make direct use of the sink mobility, therefore being a sub-optimal solution to our WSN scenario.

Unlike the many-to-one communication in the previous reactive algorithms, the TTDD [28] algorithm uses two-tier hierarchical routing to connect a source to multiple mobile sinks. For every source, the sensor network is split into individual square cells that each select a dissemination point that is located closest to the centre of the cell. The dissemination points are in charge of routing data as well as gathering information on data availability within the network. As the dissemination points are aware of the available data sources and their location, data requests of mobile sinks are only forwarded to the appropriate data sources along the grid's intermediate dissemination points. Data is then reactively sent on the reverse path

to the initially contacted dissemination point located closest to the mobile sink, from which it is forwarded to the mobile sink. If a mobile sink leaves the old cell, the new cell's intermediate dissemination point is contacted that in turn establishes a new data route for the mobile sink. As a grid of dissemination points is used for routing, the routing paths are bound to rectangular paths. A rectangular path can be $\sqrt{2}$ times longer than a direct route, however the authors assume that the increased scalability due to ease of routing out weights the sub-optimal routing path. Hierarchical routing is well suited to one-to-many communication as it does not require flooding to determine the routing path. However, as we assume many-to-one communication, the dissemination nodes would regularly fail prematurely due to their heavy routing load.

The SEAD protocol [23] is also designed for reactive synchronous data dissemination from static sensor nodes to mobile sinks. Unlike the hierarchical TTDD approach, SEAD uses a tree-routing approach that tries to minimise energy consumption in both tree construction and data dissemination to the mobile sink. A routing tree is constructed and maintained for every data source, mobile sinks having to register with the data source to then reactively receive data. The mobile sink registers and communicates with the routing tree through an *access node* to be found in the direct vicinity of the mobile sink. A new *access node* is selected to represent the mobile sink whenever the hop-count from the mobile sink to the old *access node* becomes too large, thereby reducing the need for tree reconstructions to whenever a new *access node* is selected. Intermediate replica nodes are selected to store data for intermediate storage and to regulate the data rate according to the sink's needs. Kim et al. compare the reactive SEAD protocol [23] against the directed diffusion as well as TTDD [28] algorithm and find that it performs competitively but does not outperform either in all cases. However, SEAD requires substantially less energy due to the energy saving measures in form of re-

duced tree construction frequency and shared routing paths for multiple sinks. Both SEAD and TTDD are designed to route data from a source to multiple mobile sinks (one-to-many communication) and consequently do not address our assumption of routing data to any mobile sink.

Gandham et al. [14] also assume reactive synchronous data communication between the sensor nodes and the mobile sinks. Similar to our assumptions, once data arrives at a mobile sink it is assumed to be available to the end-user, thus requiring many-to-one communication. Integer linear program formulations (ILP) are used to determine the placement of mobile sinks within the WSN that then reactively request data from the surrounding nodes. Two ILP formulations that determine the mobile sink placements are proposed: firstly, an ILP formulation that minimises the total energy consumed by all sensor nodes during a round (Min-Energy), and secondly an ILP that minimises the maximum energy spent by any sensor node in a round (min-max). The placement is conducted in a round-based system such that the sink placement occurs at the beginning of any round, therefore not taking a movement phase into account. As expected the min-max ILP achieves a longer time until the first node dies and uses less energy per round, whilst the min-energy ILP survives longer until a fraction of the nodes die and receives a larger total number of messages. The findings of Gandham et al. are of theoretical nature as their simulations ignore realistic aspects of movement in between rounds as well as the energetically often unfeasible global knowledge required for the ILP formulations. However, it does show the possibility of reducing overall energy usage by using ILP formulations to control the mobile sink's positions. As we assume uncontrolled sink mobility, this approach is also not applicable to our WSN scenario.

Proactive Data Collection Algorithms Instead of having a reactive synchronous connection between the sensor nodes and the mobile sinks as in the previous routing algorithms, the following algorithms forward data in a proactive asynchronous manner. Jain et al. [17] let data MULEs physically carry data from sensor nodes to the basestations, reducing the communication need to two hops between the sensor nodes and the MULEs as well as the MULEs and the base stations. The authors propose the usage of a replicate strategy amongst the MULEs to increase the likelihood of reaching the base station. Relying on the physical movement only restricts the delivery to a random basis, consequently not being able to guarantee a data delivery. We do not assume the mobile sinks to be intermediate data transporters, as the MULEs are, but to be the final destination.

Unlike the single hop communication and random movement as used by the data MULEs, Somasundara et al. [41] use multi-hop routing and controlled mobility to guarantee data collection from the static sensor nodes (we make no assumptions about controlled mobility). Data collection trees are created along the fixed path of the mobile sink such that data is proactively sent to nodes that are in direct contact range with the mobile sink, directed diffusion being used as the underlying routing protocol. The paper investigates the use of variable speeds during the path traversal to increase the network efficiency, in particular by spending more time at congested network areas.

Similarly to the previous routing algorithm, Ekici et al. [13] also assume controlled mobile sinks, however, unlike the previous paper, control of movement and speed of the mobile sink is assumed. The PBS algorithm computes a trajectory for the mobile sink such that all sensor nodes are individually visited before their storage buffer overflows. To do so, the data generation rate and consequently the predicted buffer overflow time at the sensor nodes is assumed to be constant and known to the mobile sink. The sensor nodes are categorised according to their

predicted buffer overflow time and multiple sub-trajectories are computed for each buffer overflow time category. The trajectories are then concatenated to a "super-cycle" such that sensor nodes with a low buffer overflow time are visited more frequently, preventing any sensor node from failing due to buffer overflow.

The second proposed data collection algorithm in [13] is the MRME algorithm that is based on the PBS algorithm but unlike the previous algorithms considers a data latency threshold σ. Sensor nodes with a buffer overflow rate greater than σ relay data to their neighbouring nodes until all nodes' buffer overflow times are below the threshold. The updated buffer overflow times of the sensor nodes are then used to recompute a PBS "super-cycle" that will meet the given latency threshold.

Lee et. al. [25] predict the mobility patterns of mobile sinks in order to determine highly frequented data-stashing nodes to which data can be forwarded for collection. Data-stashing nodes are determined offline by analysing similarities between multiple sink trajectories using a heuristic alignment method, typically employed in bioinformatics for DNA protein alignments. Once the mobility pattern is predicted, the visiting information is broadcast to all nodes within the network and utilised by the nodes to determine the closest data stashing node in the vicinity. The authors show that the energy used by the data-stashing algorithm outperforms a simple reactive algorithm when requiring the same data reliability, however fail to analyse whether delay requirements will be met by their approach.

We have described many popular routing concepts used in WSNs and MANETs. Most routing algorithms work under different assumptions about node mobility, assuming that either all nodes are stationary or mobile, or that control over the mobile sink movement is available. To our knowledge, there is a lack of adaptive routing algorithms that try to monitor sink movement and adjust their behaviour accordingly. Furthermore, application scenarios are likely to exhibit different sink mobility models, which means that, unless adaptive algorithms are designed, it is unlikely

to find an algorithm that fits all applications. Before further investigating adaptive algorithms, we will take a closer look at some of the common mobility models that have been studied in literature.

2.3 Mobility Models

Mobility models are used in simulations to guide mobile sinks and are therefore crucial to evaluate the routing algorithm performance. Bai et al. [5] analyse a wide range of techniques to generate synthetic mobility models, which can be used to simulate the movement of mobile sinks. Four main categories are described that vary in their restrictions (see Figure 2.1).

Random Models are completely unrestricted such that nodes can move freely and randomly to any point in the space.

Models with Temporal Dependency restrain mobility such that new actions have to take previous conditions into consideration, e.g. requiring a smooth acceleration from the old to the new node speed.

Models with Spatial Dependency restrict node movement spatially, e.g. respecting the distance to the next vehicle on a road.

Models with Geographic Restrictions disallow nodes to move through boundaries within the area, such as restraining the movement of vehicles to a road.

Examples and more detailed discussions of the four classes will be presented in the following subsections.

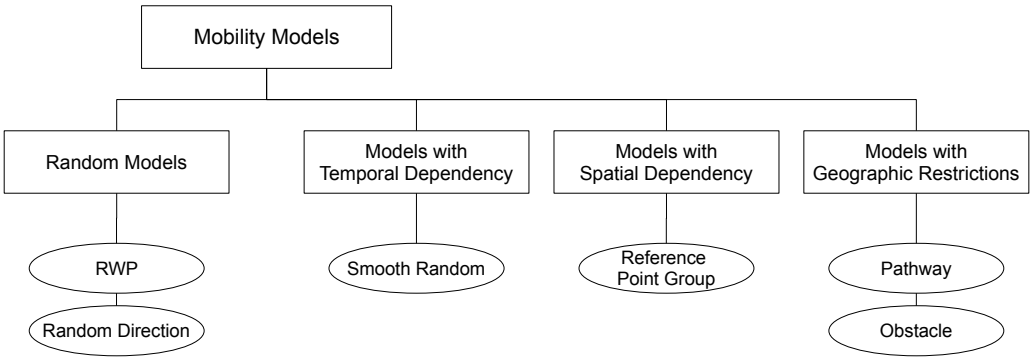


Figure 2.1: A simplified version of the mobility model classification by Bai et al. [5]

2.3.1 Random Models

The random waypoint mobility model (RWP) is one of the most widely used mobility models in MANET and WSN simulations. Nodes select their path by randomly selecting their next destination and speed, thus being very simple and straightforward to implement. The movement may be paused for a random amount of time whenever a new destination is selected. This is a simplified movement model that takes neither temporal or spatial dependencies or geographic restrictions into account, as often found with realistic mobility models. The RWP model has a *non-uniform spatial distribution* [5] such that nodes are more likely to be encountered towards the centre of the simulation, due to an increased likelihood of nodes moving towards the centre or crossing the centre on their path to a new destination.

The random direction mobility model [40] uses a similar setup to the RWP model but chooses its next destination by selecting a direction to move to. Unlike the RWP model, it overcomes the non-uniform spatial distribution [5].

Le Boudec et al. [24] give a proof of the existence of a time-stationary regime for random models and show for the RWP model that a long transition period is required to reach it. This suggests that simulations using the RWP model need to be of substantial length to fully expose the characteristics of the mobility model.

Khelil et al. [22] analyse mobility metrics for the random waypoint mobility

model, assuming all nodes are mobile. New metrics are proposed for delay-tolerant sensor networks including average contact rate ($ACR_n = Contacts/Time$), average encounter frequency ($AEF_n = |Encounters_n|/|Contacts_n|$) and average contact duration (ACD). The metrics are found to respond as expected, e.g. the average encounter frequency (AEF) is unaffected by the node density but scales linearly with a higher node mobility. Furthermore, a theory is established that connects node encounters with wireless range, node speed and node density that is shown to be correct both theoretically as well as in the simulated environment.

2.3.2 Models with Temporal Dependency

After having briefly discussed the random mobility models, a mobility model with temporal dependency is introduced. The *Smooth Random Mobility Model* [7] models a smoother, thus more realistic movement. Unlike the RWP model, changes in both speed and direction are adjusted smoothly such that the new speed is reached using a (de)acceleration phase and the change of direction is smoothly conducted using a movement radius. Furthermore, the speed is not set at random but according to a probability distribution that favours a set of preferred speeds, similar to the set of road speed limits in real-life.

Bettstetter [7] also investigates the effect of border behaviour on the spatial distribution of nodes for the smooth movement model. The bounce border behaviour in which a mobile node moves at the same outgoing angle as the incoming angle leads to the sides being less frequented than the central simulation area. The delete and replace behaviour effectively moves a node instantaneously into a new random position upon reaching the boundary area of the simulation field. This also leads to a non-uniform node distribution as the sides are less frequented than the centre. Only the wrap-around border behaviour in which a node continues moving on the opposite side of the simulated area at the same angle offers an equal

visiting probability.

2.3.3 Models with Spatial Dependency and Geographic Restrictions

Models with spatial dependency have to take surrounding objects into account, such as vehicles that are sharing the same road and have to stay at a safe distance to vehicles ahead.

Bai et al. [6] suggest three new mobility models that underlie spatial dependencies and geographic restrictions. The *reference point group model* (*RPGM*) is a spatial dependency model in which nodes have to stay within the vicinity of a roaming leader node, the leader node moving with a RWP model. A Military offensive is an example of the *RPGM* model in which soldiers have to follow their officers but may take independent initiative at times. The model can potentially encompass multiple groups, each having their own leaders. Unlike the RPGM model in which nodes have to stay within the vicinity of a leader, the *freeway mobility* model simulates a motorway traffic environment in which nodes have to keep a security distance to cars ahead. The cars are bound to simple freeway models, the mobility model therefore being both spatially dependent on other nodes as well as geographically restricted by the freeways. The last model suggested is similar to the *freeway model* in that it also simulates road traffic, however in a city centre environment, it is often referred to as the *Manhattan* mobility model due to its grid road layout. Cars are restricted to the roads and have to make direction decisions at road intersections, which suggests it is a geographically restricted mobility model. The grid layout and the random decisions made at road intersections allow for predictable randomness with an increased visiting likelihood towards the centre of the grid. After proposing the three mobility models (RPGM, freeway and Man-

hattan), Bai et al. use a multitude of mobility and connectivity metrics to analyse the mobility models, with a particular emphasis on using the metrics to distinguish between the mobility models. The conducted experiments show that out of the investigated mobility metrics the geographic restrictions, average relative speed as well as average degree of spatiality metrics were able to differentiate between the individual mobility models. Amongst the connectivity metrics the average link duration as well as average path duration are found to be able to differentiate between the models.

Jardosh et al. [18] describe an interesting way of implementing an obstacle pathway model and compare the model to the RWP model. Pre-defined polygon obstacles are placed within the area for which pathways are computed using Voronoi diagrams. In effect, the paths created are set in between all the obstacles such that the obstacles can be circumnavigated. A reachability matrix is used to determine whether any radio signal is absorbed by the obstacles, assuming that the obstacles completely absorb any signal. The comparison to the standard RWP model, using the AODV routing protocol, finds that the obstacle model performs differently mainly due to the reachability matrix reducing the potential communication and therefore also reducing contention within the model.

2.4 Summary

In this chapter we described the literature related to the thesis.

We provided an overview of the most popular routing protocols designed for MANETs and WSNs. We pointed out that the majority of WSN routing protocols either assume no sink mobility, or fixed/predictable/controlled mobility. To our knowledge there is currently no existing work on adaptive routing algorithms that can learn and adjust to potentially changing mobility patterns. In the next chapter,

we analyse the behaviour of two existing proactive and reactive algorithms in the context of fixed WSNs with mobile sinks.

Chapter 3

Static Routing Protocols

In this chapter, we revisit existing data collection schemes in more detail, and show how they can be applied in a scenario with fixed sensor nodes and mobile sinks. Recall that they can be classified into two broad categories: *proactive* and *reactive* schemes. *Proactive* schemes proactively push data to carefully selected storage nodes, from where data is collected by roaming sinks. In *reactive* schemes, mobile sinks continually advertise their presence to nearby sensors, and the latter react by forwarding their data back to the sinks.

Reactive and proactive schemes exhibit different strengths and weaknesses in particular with regards to sink mobility and routing maintenance costs [43]. To investigate this behaviour, two representative implementations of pure schemes will be presented that will be used to later analyse the characteristics of each. A detailed description of the two pure schemes and a description of their behaviour are provided in the following sections.

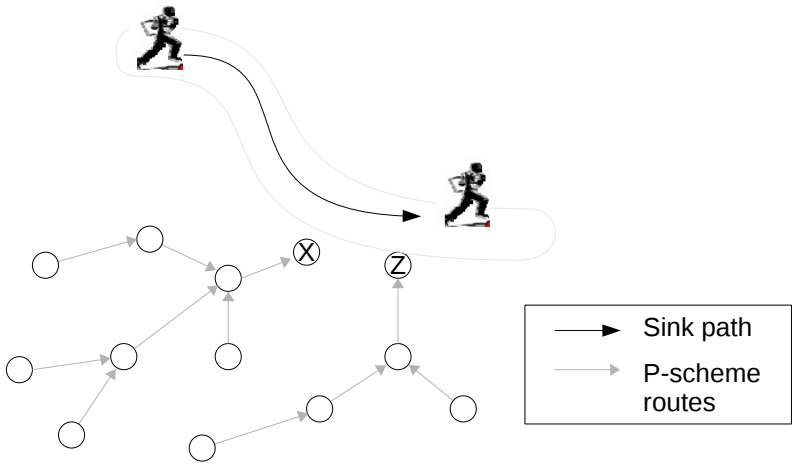


Figure 3.1: P-scheme in action in an example sensor network where node X and Z have been selected as storage nodes, leading to the construction of data collection trees.

3.1 Proactive Algorithms

Proactive schemes proactively push data to carefully selected storage nodes, from where data is collected by roaming sinks. Typically, frequently visited sensors are selected as storage nodes according to the long-term visiting behaviour of mobile sinks [41, 25]. A small routing maintenance overhead is typically required as changes in storage node selection are assumed to be infrequent. Due to the reliance on long-term trends, proactive schemes are less able to deal with random or unexpected changes in sink mobility.

3.1.1 P-scheme Algorithm

In this section we present an exemplary proactive scheme, named P-scheme, inspired by existing algorithms that build and maintain data dissemination trees rooted in pre-defined storage nodes [41, 31, 32, 25].

The P-scheme algorithm selects storage nodes at locations that are frequented by mobile sinks, and builds dissemination trees rooted at them, thus enabling the remaining sensors to proactively forward their data to the closest storage node

(see Figure 3.1). More specifically, P-scheme is a localised algorithm that involves three tasks, as shown in the pseudocode depicted in Table 3.1.

The first task (P1) sets the local variables according to whether a node has been selected as a storage node. The P-scheme algorithm assumes prior knowledge of nodes that are sufficiently frequented by mobile sinks to allow for timely data collection. The algorithm therefore assumes that the storage nodes are selected prior to the deployment at frequented locations such as roads or social hotspots at which mobile sinks can reliably collect the data in a timely fashion. Once a node is selected as a storage node ($SN = True$), it informs its surrounding neighbours by transmitting a beacon which initiates the creation of a data collection tree.

The second task (P2) describes how a sensor processes a tree construction beacon upon reception, so that it associates itself to a route leading to its closest (in terms of hops) storage node. The third task (P3) presents a zero-wait, push data reporting mechanism that sends data along the established routes to the closest storage node, or directly to the sink if the latter is in range.

Figure 3.1 illustrates the key ideas behind the algorithm using a simple network. The mobile sink (a human carrying a mobile device) repeatedly travels by node X and Y. Therefore, node X and Y are selected as storage nodes and the routing trees direct traffic towards them for data collection by the mobile sinks.

3.1.2 Algorithm Behaviour

We are now in a position to look at the behaviour of P-scheme, in a simple scenario. We start by describing the experimental setup used to simulate its behaviour in Section 3.1.2.1, followed by a parameter sensitivity analysis in Section 3.1.2.2. Finally we discuss how the algorithm trades communication cost for data latency in Section 3.1.2.3.

INPUT VARIABLES:
SN - boolean determining storage node selection;
ID - node identity;
LOCAL VARIABLES:
$distP$ - number of hops to storage node;
$parentP$ - parent node ID in routing tree;

MAIN ALGORITHM:
$distP = \infty; parentP = ID; visits = \emptyset;$
WHILE true:

[P1] IF $SN == True$:
$distP = 0$; $b.dist = 0$; $b.source = ID$;
broadcast b;

[P2] IF a P beacon b is received:
IF $b.dist < distP$**:**
$distP = b.dist$; $parentP = b.source$;
$b.dist = b.dist + 1$; $b.source = ID$;
broadcast b;

[P3] IF any data d:
set $d.src = ID$;
IF sink in range:
send *all* data d to sink;
ELSE IF $parentP \neq ID$:
send *all* data d to $parentP$;

Table 3.1: Pseudocode of P-scheme.

3.1.2.1 Experimental Setup

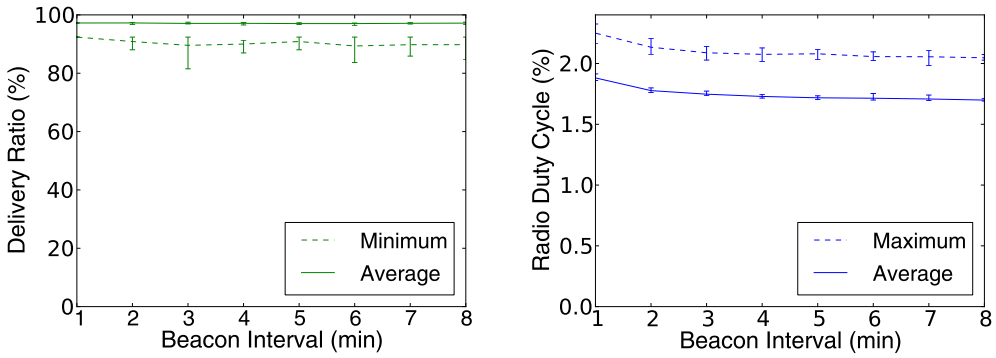
The test environment was simulated in Contiki's cooja simulator using 64 static sensor nodes distributed in a 8x8 grid with each sensor being spaced 60 meters apart, the radio range being 75m. The data latency requirement was set to 200 seconds with a required data delivery ratio of 80%, a data package being generated at every node once a minute. Each test simulated a period of 180 minutes, from which only the last 90 minutes were used to gather data in order to allow the

network to start-up and settle. Three mobile sinks roam the network using a random waypoint (RWP) model with a speed of 5m/s, this being similar to a standard cycling pace. Each individual test is repeated 4 times in order to increase the accuracy of results, since a real network stack is subject to variance in performance due to interference, transmission timings and data loads (see Section 6.2 for more detailed information). This setup will be used in the following sections to describe the behaviour of the algorithms.

3.1.2.2 Parameter Sensitivity Analysis

The P-scheme algorithm only contains one parameter that needs tuning, namely the interval at which beacons are transmitted ($BeaconInterval$). The shorter the beacon interval, the quicker trees are constructed and deconstructed while also incurring a higher communication cost. Conversely, the longer the beacon interval, the lower the communication cost and the more likely the network will contain stale routing information, leading to data being delivered in a less timely manner. Figure 3.2 shows the sensitivity analysis of the $PBeaconPeriod$ in which a longer beacon interval results in a minimally lower delivery ratio and a lower communication cost.

Figure 3.2 shows that If we do not take into account network dynamics (i.e. node / link failures), $BeaconInterval$ has little effect on delivery ratio. However, in a real sensor network, such as the one deployed in Wytham Woods for badger monitoring [46], the quality of links changes over time [4]. In practice, setting $BeaconInterval$ to 5 mins allowed us to successfully recover from short-term changes in the network topology. In the following sections, we use 5 minutes as the default value of $BeaconInterval$.



(a) Delivery ratio

(b) Communication cost

Figure 3.2: Tuning of the $BeaconInterval$ paramater for the P-scheme algorithm. When the beacon interval is small, the communication cost increases due to the frequent beacons. Likewise, the communication cost is reduced when using a larger beacon interval. The practical value, also used during the real-life deployment of the project's WSN, was $BeaconInterval = 5min$.

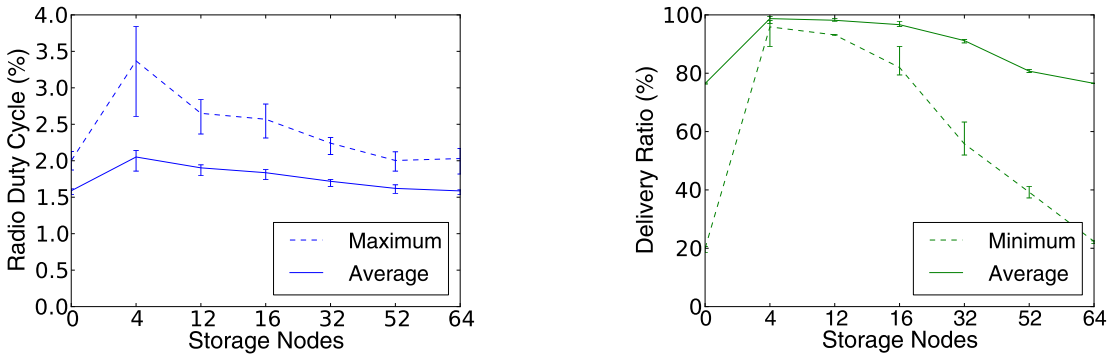
3.1.2.3 Algorithm Trade-off

After having described the P-scheme algorithm and the experimental setup, this section illustrates the behaviour of the algorithm, namely its ability to gracefully trade communication cost for data freshness. The storage node boolean SN (see Table 3.1) defines whether a node acts as a storage node. If SN is set to true, the node initiates the creation of a data collection tree, allowing a mobile sink to collect data from the storage node. Static P-scheme requires the network engineer to know the frequented locations of mobile sinks and configure nearby sensors to act as storage nodes ($SN = True$). Depending on the sink mobility, a network engineer can control how many storage nodes exist in the network by expanding the number of storage nodes in less frequented areas. The less frequently a storage node is visited, the higher the data latency is likely to be. On the other hand, the more nodes are selected as storage nodes, the smaller the communication trees formed around them. By carefully selecting the number and distribution of P-scheme's storage nodes, a network engineer can gracefully trade off communi-

cation cost for data latency.

This trade-off is illustrated in Figure 3.3 in which the number of storage nodes has been varied for the P-scheme algorithm. Figure 3.3a shows how the communication cost increases when fewer storage nodes are selected. Zero selected storage nodes result in mobile sinks only collecting data locally, thereby reducing the communication cost to a minimum. Conversely, Figure 3.3b shows how fewer and better placed storage nodes allow for more data being delivered within the latency requirement of 200 seconds.

In a deployment setting, the network engineer would have to decide on this trade-off in order to reliably achieve given application requirements. If a network engineer had experimentally determined this trade-off for a data latency of 200 seconds, as done in simulation for Figure 3.3, he could optimise the network accordingly. In order to achieve a data delivery ratio of 80% the network engineer would choose to select 16 optimally placed storage nodes, as this setting is the one with the lowest communication cost that achieves the data delivery ratio (see Figure 3.3b). The highest radio duty cycle used by a node when selecting 16 storage nodes is a radio duty cycle of 2.7%, as can be seen in Figure 3.3a. It is therefore evident that the network engineer has to optimise the storage node selection such that the WSN meets or exceeds the data requirements while minimising the communication costs.



(a) Communication Cost

(b) Delivery Ratio

Figure 3.3: P-scheme algorithm performance when varying the number of optimally selected storage nodes for a) the communication cost and b) the delivery ratio for a given data latency requirement of 200s. By carefully selecting the number of storage nodes, we can trade-off data delivery ratio for communication cost.

3.2 Reactive Algorithms

An implicit assumption behind the proactive scheme presented in the previous subsection is that there is some predictable long-term trend in the collective behaviour of sink trajectories. In particular, the P-scheme exploits the fact that the frequency of mobile sink visits at a sensor can be forecast. As a result, a constantly changing network, e.g. one with alternating periods of high and low sink traffic, can significantly undermine the effectiveness of this scheme. Worse, some sensor networks may experience periods of random sink trajectories (e.g. visitors to a national park) followed by periods of highly repeatable sink trajectories (e.g. scientists studying animal species in a national park). In this case, it may be difficult to discover a consistent visit history to make accurate predictions for storage node placements.

Wary of this, we present R-scheme, a reactive algorithm that does not presume long-term statistics and reacts to the movement of individual sinks in real-time, it being inspired by [27]. In reactive schemes, mobile sinks continually advertise their presence to nearby sensors, and the latter react by forwarding their data back to

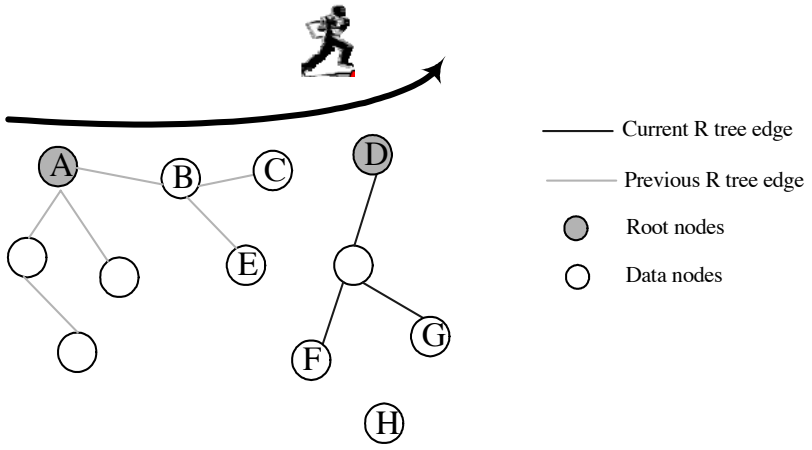


Figure 3.4: R-scheme in action with $TD = 2$ in an example sensor network.

the sinks. By continually reacting to the movement of the mobile sinks the routing protocols deal well with random or unpredictable movement, however, continually advertising the presence of mobile sinks also incurs a high routing maintenance cost.

The strength of reactive schemes is particularly apparent when mobile sinks are subject to unexpected movement, as would occur when a monitored car would have to take a detour due to road closure or a monitored animal behaved differently due to the presence of predators. As mobile sinks continually update their presence, the data is always forwarded to the current location of the mobile sink. The reactive schemes independence of predictable or unpredictable sink mobility is however offset by the energetically expensive procedure of having to continually advertise their presence and maintain the data collection tree.

In the following section, we will describe our example implementation of a reactive scheme that we call R-scheme.

3.2.1 R-scheme Algorithm

As described in the pseudocode in Table 3.2, the R-scheme is a localised algorithm consisting of six tasks. In the first task (R1), a roaming mobile sink initiates

a routing-tree construction process within its radio range. In the second task (R2), static nodes handle routing-tree construction beacons received from their neighbours. Both (R1) and (R2) utilise the third task (R3) which ensures that no tree construction is initiated if 1) a beacon exceeds a maximum tree depth of TD hops or 2) the node has been visited very recently. The purpose of the first restriction is to constrain the depth of reactive trees and thus the energy consumed in constructing them and in delivering data along their paths. The second restriction follows the intuition that it is not worth extending the reactive tree to an area where there is little data left to report.

The fourth task (R4) automatically destroys the current tree when after a short interval the mobile sink moves out of range of the tree's root[1]. The fifth task (R5) allows nodes to participate again in tree construction after a sufficient time interval has elapsed during which new data has been generated. In the sixth task (R6), nodes forward buffered data to their parent in the reactive tree or to a mobile sink if within range.

Figure 3.4 depicts how reactive routing works in an example sensor network, where a mobile sink traverses node A, B, C and D. In particular, the figure depicts the moment when a new tree rooted at D is being constructed. Nodes B, C and E do not participate in the new tree rooted at D, as they were very recently part of the tree rooted at A ($justCoveredTimer$). For the same reason, B and C did not become root nodes when visited by the sink. In addition, tree construction beacons are propagated up to nodes F and G, and they are not forwarded to H, as the maximum tree depth ($TD = 2$) has been reached.

[1]For simplicity, we assume that this interval is the same for all nodes in the tree, and it suffices to empty their buffers and forward any data to the sink before it moves out of range. This is realistic in applications where the sink does not move very fast, and routing trees are relatively shallow.

3.2.2 Algorithm Behaviour

After having described the P-scheme algorithm, we are now in a position to look at the behaviour of the algorithm. At first, we conduct a parameter sensitivity analysis, followed by a description of the trade-off that can be made between communication cost and data latency. All simulations described within this section use the previously described experimental setup (see Section 3.1.2.1).

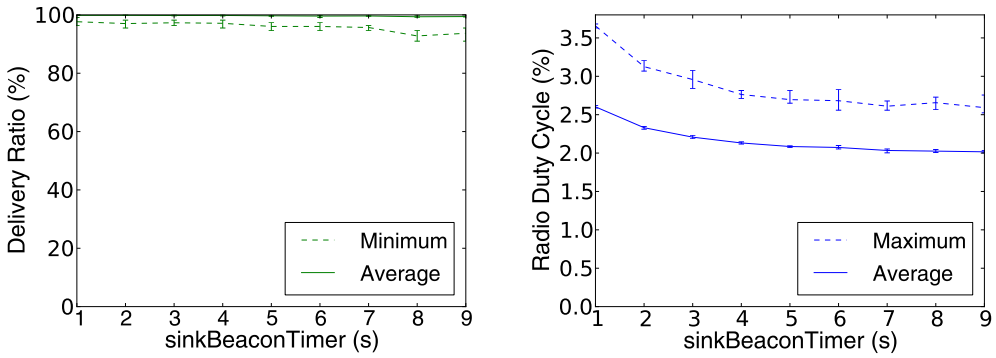
3.2.2.1 Parameter Sensitivity Analysis

In this section, we describe the impact of varying the following three parameters of R-scheme on the performance of the algorithm.

- $sinkBeaconTimer$: Describes the time interval at which a sink sends a beacon to notify surrounding sensor nodes of its presence.
- $treeDestructTimer$: The period for which a sink is assumed to be within communication range of the data collection tree.
- $justCoveredTimer$: The period during which a sensor node will not reparticipate in a new multi-hop R-scheme tree as its data has already been recently collected.

The **sinkBeaconTimer** as well as the $treeDestructTimer$ are primarily dependent on the duration of a sink visit. The faster a mobile sink moves or the shorter its communication range, the shorter its visit duration will be. Conversely, the slower a mobile sink moves and the longer its communication range, the longer its visit durations will be.

The aim of configuring the $sinkBeaconTimer$ is to choose a sufficiently small interval to minimise the number of beacons, thereby reducing the communication



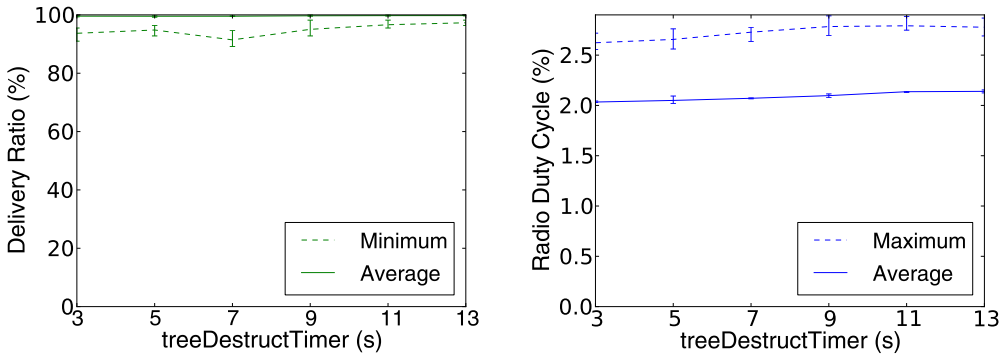
(a) Delivery ratio

(b) Communication cost

Figure 3.5: Sensitivity analysis of R-scheme's $sinkBeaconTimer$ parameter. b) When increasing the $sinkBeaconTimer$, the radio duty cycle decreases due to fewer beacons being transmitted. a) The increase of the $sinkBeaconTimer$ only has a negligible impact on the delivery ratio. A good trade-off is achieved when setting $sinkBeaconTimer = 5s$.

cost, while allowing for timely data collection. Figure 3.5 shows how the performance varies as we change the beacon interval configuration. Up until a beacon interval of 8 seconds it has a negligible impact on the data delivery ratio, while clearly showing that an increased beacon interval results in a lower radio duty cycle. The $sinkBeaconTimer$ was chosen to be 5 seconds as the radio duty cycle savings are minimal when increasing the beacon interval further.

The **treeDestructTimer** determines the time that an R-scheme tree remains active. Once this time elapses the tree is destroyed without explicitly sending destruction messages. The smaller the $treeDestructTimer$ is, the less likely data will be sent to a potentially out-of-range sink, thereby decreasing the incurred communication cost. However, the smaller the $treeDestructTimer$, the less data can be sent to the mobile sink, thereby potentially reducing the achieved data delivery ratio. These trends can be seen in Figure 3.6, though a change has a marginal impact on the radio duty cycle, as the used network stack (Contiki's uIP) detects and if necessary aborts data transactions if the destination is busy or unavailable.



(a) Delivery ratio

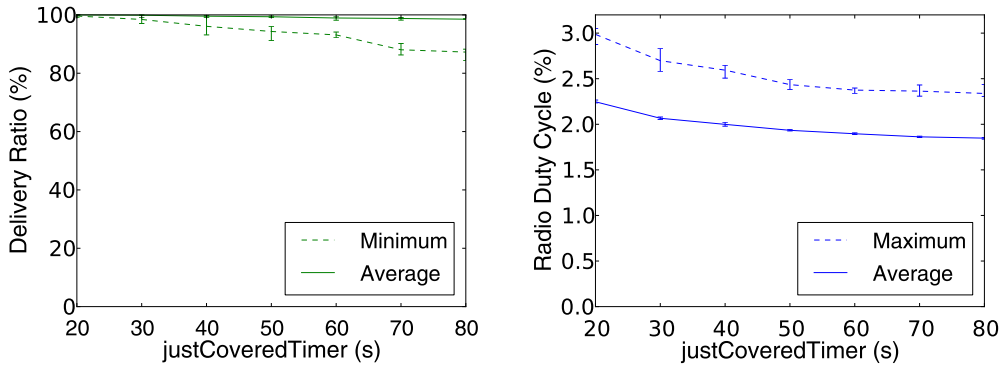
(b) Communication cost

Figure 3.6: Parameter tuning of R-scheme's $treeDestructTimer$. The sooner the R-scheme trees are destructed, the less likely data will be sent to an out-of-range sink, decreasing the radio duty cycle. However, a smaller $treeDestructTimer$ also decreases the throughput, potentially decreasing the data delivery ratio. The configuration was set to $treeDestructTimer = 9s$.

The primary role of the **justCoveredTimer** is to prevent R-scheme from trying to collect data multiple times within a short period of time in the same area, when potentially few or no data packages are available. The longer the $justCoveredTimer$ is set, the fewer trees will be constructed leading to a reduced radio duty cycle, as can be seen in Figure 3.7. As the maximum assumed visit duration is 20 seconds, a sufficiently larger value is advisable, though of course too large a value could potentially harm the effectiveness of the R-scheme algorithm. The $justCoveredTimer$ was therefore set to 30 seconds, as Figure 3.7 shows that the radio duty cycle reduction thereafter follows the reduction in the achieved data delivery ratio.

3.2.2.2 Algorithm Trade-off

We are now in a position to discuss the behaviour of the R-scheme algorithm when varying its main tuneable parameter, namely the tree-depth (TD). The tree-depth (TD) value must be decided based on application data requirements. The higher the TD, the better the data latency for collected data will be, at the expense of a



(a) Delivery ratio

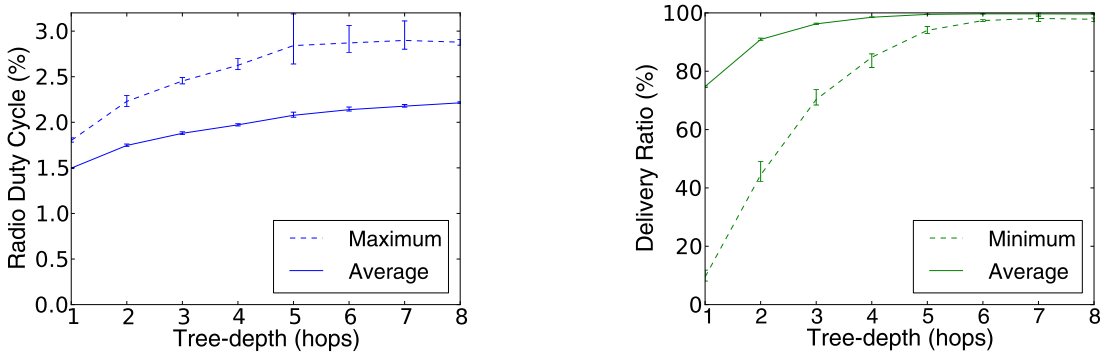
(b) Communication cost

Figure 3.7: Tuning of R-scheme's $justCoveredTimer$. The larger the $justCoveredTimer$, the less frequently R-scheme trees will be constructed in the same area, leading to a lower radio duty cycle as well as a potentially lower data delivery ratio. The configuration was set to $justCoveredTimer = 30s$.

higher communication cost. Conversely, the lower the TD the lower the communication cost at the expense of a lower data latency. Thus, by carefully varying TD, the R-scheme gracefully trades communication cost for data latency.

This can be seen in Figure 3.8 in which the communication cost (radio duty cycle) as well as the data delivery ratio are shown for different tree depths TD. Recall that the delivery ratio is the percentage of packages delivered within the data latency requirements. The simulation setup used remains the same as described for the P-scheme behaviour (see Section 3.1.2.1). The communication cost (Figure 3.8a) is minimal when the tree depth TD is limited to 1, allowing only data collection from nodes within direct communication range of the mobile sink. However, a minimal energy usage at tree depth TD of 1 also results in the lowest data delivery ratio for a given data latency, as can be seen in Figure 3.8b. With an increase in tree depth TD, both the communication cost as well as the delivery ratio increase allowing for a trade off between the two to optimally achieve a predefined application data requirement.

The optimal trade-off of the WSN performance is conducted by a network en-



(a) Communication Cost

(b) Delivery Ratio

Figure 3.8: R-scheme algorithm performance when varying the maximum tree-depth (TD) and the resulting a) radio duty cycle as well as b) the delivery ratio for a latency requirement of 200 seconds. Increasing the tree-depth also increases the resulting delivery ratio as well as the required radio duty cycle. A balance can therefore be struck that allows us to achieve data latency and delivery ratio requirements at the lowest possible communication cost.

gineer. Assuming a network engineer needs to achieve a delivery ratio of 80% for the given scenario, configuring a $TD = 4$ would be sufficient to collect the data on time (see Figure 3.8b). If the network engineer would want to be particularly conservative, configuring R-scheme for a tree depth $TD = 5$ would allow for sufficient leeway to ensure the data requirements will be achieved, at the expense of a higher communication cost. It may also be noted, that in the given scenario a $TD = 6$ or higher will result in most sensor nodes participating frequently in R-scheme trees leading to no further gain in terms of data delivery when further increasing TD.

INPUT VARIABLES:
TD - reactive routing tree depth (number of hops);
ID - node identity;
LOCAL VARIABLES:
$distR$ - number of hops to sink;
$parentR$ - parent node ID in routing tree;
$justCovered$ - flag indicating recent tree coverage;

MAIN ALGORITHM:
$distR = \infty$; $parentR = ID$;
$justCovered = false$;
WHILE true:
[R1] IF mobile sink comes in range:
  IF $TD > 0$ **AND** $justCovered == false$:
    $b.dist = 0$; $b.source = ID$; $b.distMax = TD$;
    call **R3**

[R2] IF R beacon b received:
  call **R3**

[R3] IF $b.dist < distR < TD$ **AND** $justCovered == false$:
  // update R-scheme
  $distR = b.dist$; $parentR = b.source$;
  $justCovered = true$;
  start $justCoveredTimer$;
  start $treeDestructTimer$;
  // broadcast tree if next hop smaller than TD
  IF $b.dist + 1 < b.distMax$:
    $b.dist = b.dist + 1$; $b.source = ID$
    broadcast b

[R4] IF $treeDestructTimer$ expires:
  $distR = \infty$; $parentR = ID$;

[R5] IF $justCoveredTimer$ expires:
  $justCovered = false$;

[R6] IF any data **AND** sink in range:
    send *all* data to sink;
  ELSE IF any data **AND** $parentR \neq ID$:
    send *all* data to $parentR$;

Table 3.2: Pseudocode of R-scheme.

3.3 Summary

In this chapter we presented and described example implementations of proactive and reactive data collections schemes, namely P-scheme and R-scheme. Proactive algorithms send their data to storage nodes from which data is collected by mobile sinks. Proactive data collection schemes depend on predictable sink mobility in order to enable the selection of adequate storage nodes. Furthermore, proactive schemes have a lower routing maintenance overhead due to infrequent changes in storage nodes. Network engineers can trade-off data latency for communication cost by only selecting few storage nodes in the most frequently visited places, that deliver data to sinks with low latency at the expense of higher communication costs. Conversely, a lower communication cost can be achieved if a network engineer selects many storage nodes in less frequented areas, thereby reducing the size of the data collection trees at the expense of higher latency.

Reactive data collection schemes frequently build collection trees around mobile sinks, allowing it to easily adapt to change of mobile sink mobility. The frequent construction of trees allows reactive schemes to quickly adapt to changing sink mobility patterns, however at a higher routing maintenance cost. Network engineers can trade-off data latency for communication cost by increasing the depth of data collection trees around mobile sinks. Conversely, by building smaller data collection trees the communication cost can be reduced, hence allowing for a seamless trade-off between communication cost and achieved data latency.

We have described example implementations of the data collection schemes, namely P- and R-scheme. We have shown how these implementations can be configured and how they behave when varying their primary configuration parameters, namely storage node (SN) selection for P-scheme and tree-depth (TD) for R-scheme. The static implementations of P- and R-scheme require a network en-

gineer to configure the data collection schemes for each individual data collection scenario. In the following chapter we will describe adaptive versions of the collection schemes that adjust their behaviour to given scenarios and changes occurring during the deployment. We will show that the proposed adaptive algorithms relieve engineers from the tedious task of parameter configuration and therefore ease the deployment of sensor networks in a large range of scenarios.

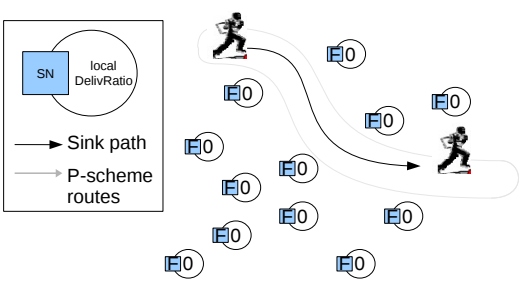
Chapter 4

Adaptive Routing Protocols

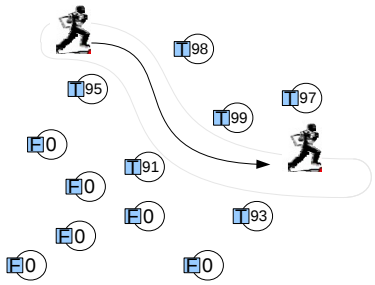
The static pure routing algorithms, described in the previous chapter, require careful configuration by an expert engineer prior to deployment. This is often impractical, and can lead to misconfigurations if the engineer cannot accurately predict the sink mobility patterns, or if the latter change over time. In this chapter, we propose adaptive versions of the P- and R-schemes, which can configure their behaviour according to observed mobility patterns, and are able to adjust their parameters in response to mobility changes. They are designed to adhere to application-specific requirements for latency and delivery ratio, while minimising the communication cost required to do so. Details of the proposed algorithms are presented below.

4.1 Adaptive P-scheme

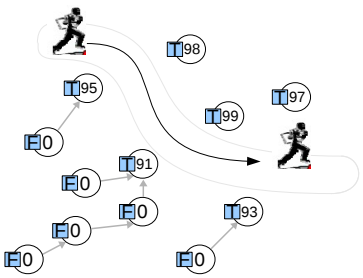
In this section, we introduce Adaptive P-scheme, a novel proactive routing algorithm that self-configures its parameters taking into account application requirements and observed patterns of sink mobility. Recall that in the static P-scheme, the network engineer had to manually select storage nodes, where sensor data were to be gathered and later on picked up by mobile sinks. The proposed Adap-



(a) An initial network with no selected storage nodes ($SN = False$). Nodes begin registering mobile sink visits.



(b) After having registered a sufficient amount of sink visits, nodes can determine their $localDelivRatio$, namely the percentage of data that can be delivered within the application latency requirements. Nodes that achieve a sufficiently high delivery ratio, select themselves as storage nodes ($SN = True$).



(c) Following the storage node selection, the data collection trees are constructed.

Figure 4.1: Example adaptive storage node selection within a network.

tive P-scheme relieves the network engineer from this tedious task. The main idea is that each sensor node keeps track of its own sink visits, and makes local decisions about whether to become a storage node, or to deselect itself from this role.

We do not assume knowledge of the geographic location nor do we assume control over the movement of the mobile sinks (see Section 1.1.1). Furthermore, we assume that the mobile sinks exhibit a common visiting pattern from which we can predict the future visiting behaviour. This common visiting pattern may be based on individual sink mobility or drawn from an aggregate pattern of multiple mobile sinks. Consequently, our approach to selecting storage nodes is simply

based on analysing the history of sink visits and identifying whether a node is likely to be visited frequently enough to enable a timely data collection. Figure 4.1 illustrates the principal behaviour of the adaptive storage node selection. The following sections describe the Adaptive P-scheme algorithm and explore the impact of varying its parameters on performance.

4.1.1 Algorithm

Four new tasks were added to the previously described pure P-scheme algorithm that enable us to identify appropriate storage nodes (see Table 4.1): 1) Registration of sink visits. 2) Setting asymmetric selection and deselection latency requirements. 3) Determining the local delivery ratio. 4) Storage node selection/deselection. Each will be now be described in more detail.

Registration of sink visits (task P5A): Every fixed sensor simply logs the time and duration of each sink visit (see task P5A in Table 4.1). Note how we do not log the sink ID, as we are only interested in the common visiting pattern and not in specific sink IDs. This reduces both complexity of the registration process as well as reduces the memory footprint used for storing the sink visit information to two integers (time of arrival, duration of visit).

Setting asymmetric selection and deselection latency requirements (task P6A): A subtle point in the design of the algorithm is the use of *asymmetric* criteria to select and deselect storage nodes. By only conservatively selecting storage nodes, we minimise the potential of storage nodes being selected that do not satisfy the data requirements in the long term.

We do so by letting nodes become storage nodes, if they satisfy a lower (stricter) latency requirement than that required by the application. We refer to the latency threshold used for selecting storage nodes as *selectLatency*. For example, if the latency requirement set by the application is 100s, and we want to be 30% conser-

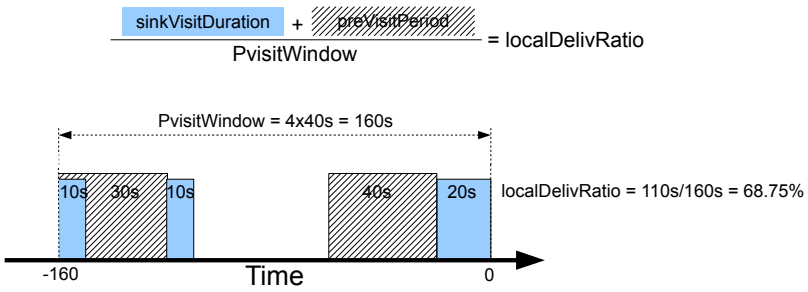


Figure 4.2: Example scenario for a sensor node's historically achieved P-scheme delivery ratio, for a data latency of 40 seconds and a $PvisitWindow = 4 \times 40s = 160s$. The delivery ratio is determined by first finding the time data would have been delivered on time, consisting of, firstly, the time a sink was in touch with a sensor (see $sinkVisitDuration$, $10s + 10s + 20s = 40s$) and, secondly, the period prior to a sink visit for which data would still be delivered on time (see preVisitPeriod, $30s + 40s = 70s$). In this example, the $localDelivRatio = (sinkVisitDuration + preVisitPeriod)/PvisitWindow = (40s + 70s)/160s = 68.75\%$

.

vative, we get a selectLatency of 70s, as shown below:

$$\begin{aligned} selectLatency &= LatencyReq \times (100\% - PselectLatencyRate) \\ &= 100s \times (100\% - 30\%) \\ &= 70s \end{aligned}$$

Note how the resulting $selectLatency = 70s$ is more conservative and harder to achieve than the original application $LatencyReq = 100s$. Conversely, in order to assess whether a node should be deselected as a storage node if a node is already a storage node, the selection latency is simply set to the application latency requirement:

$$selectLatency = LatencyReq = 100s$$

By having an asymmetric selection and deselection latency, we can ensure that only sufficiently overachieving nodes are selected as storage nodes that are likely to continue achieving the data latency requirements.

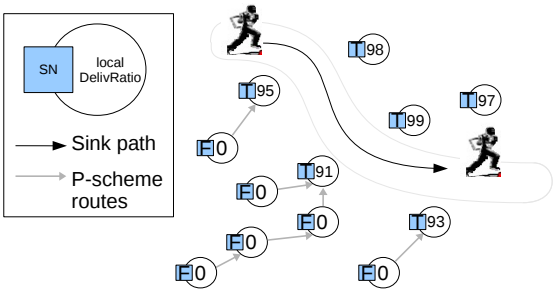
Determining the local delivery ratio (task P7A): Each node periodically looks at its recent history of sink visits in order to estimate its local delivery ratio ($localDelivRatio$), i.e. the percentage of sensor readings that it can deliver within its $selectLatency$. The $localDelivRatio$ is determined according to how often generated data would have been delivered on time to a mobile sink in the recent period. The time data would have been delivered ($timeDelivered$) can be split into two separate periods, namely the $sinkVisitDuration$ and the $preVisitPeriod$ (see Figure 4.2). Firstly, the $sinkVisitDuration$ is the time during which a node is in direct contact with a sink, enabling it to deliver all data generated during this period on time. Secondly, the $preVisitPeriod$ is the period prior to a sink visit in which generated data can be delivered on time due to the imminent sink visit. The $preVisitPeriod$ is bound by the data latency, namely the previous set $selectLatency$, as only data generated within the latency requirements would have been delivered on time (see Figure 4.2)

Only sink visits during the $PvisitWindow$ are taken into consideration in order to ensure sample freshness as well as to restrict the sample size. Having computed the time data would have been delivered during the $PvisitWindow$, we can infer the $localDelivRatio$ as follows:

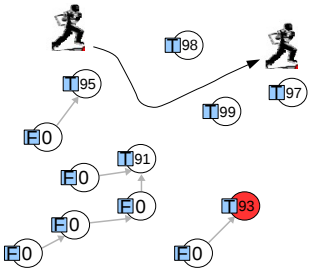
$$localDelivRatio = \frac{timeDelivered}{PvisitWindow} = \frac{sinkVisitDuration + preVisitPeriod}{PvisitWindow}$$

A statically sized sliding visit window ($PvisitWindow$) is used as it offers the simplest implementation. Figure 4.1b provides an example of how nodes within a WSN determine their respective delivery ratio.

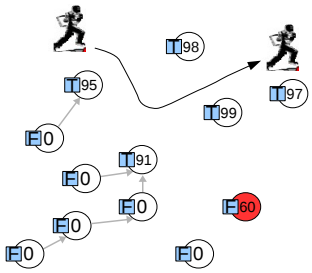
Storage node selection/deselection (task P8A): Once a node has determined its historic delivery ratio ($localDelivRatio$), it determines whether to select or deselect itself as storage node, depending on whether it meets the data delivery



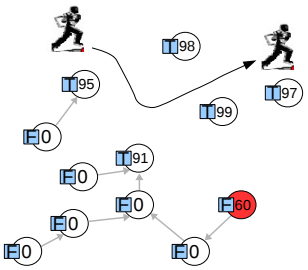
(a) Original network setup with selected storage nodes close to the sinks routes.



(b) One storage node (highlighted) is no longer visited after the mobile sink route changes.



(c) In due course, the highlighted node deselects itself as storage node ($SN = False$) and destructs the routing tree.



(d) The nodes rejoin another routing tree in order to deliver their data in a timely manner.

Figure 4.3: An example illustrating how a storage node deselects itself and joins another routing tree following a change a mobile sink's path.

requirement set by the application:

IF $localDelivRatio >= DeliveryRatioReq$:

$SN = True$;

ELSE:

$SN = False$;

Note that nodes that are not storage nodes use the more conservative $selectLatency$, which will be reflected in a more conservative $localDelivRatio$. An example storage node deselection is shown in Figure 4.3 in which a storage node is no longer visited by mobile sinks, leading to an insufficient $localDelivRatio$ and the node being deselected as a storage node.

INPUT VARIABLES:
$DeliveryRatioReq$ - Application delivery ratio requirement;
$LatencyReq$ - Application latency requirement;
$PselectLatencyRate$ - Storage node selection threshold;
$PvisitWindow$ - Sliding window size for sink visits;
LOCAL VARIABLES:
$visits$ - List of sink visits;
$selectLatency$ - The minimum latency (de-)selection requirement;
$localDelivRatio$ - The delivery ratio achieved by the node;

[P5A]: If sink within reach:
register time and duration of sink visit;
IF $PadaptiveTimer$ inactive:
start $PadaptiveTimer$

[P6A]: IF $PadaptiveTimer$ expires:
// Set asymmetric selection/deselection latency
IF SN==False:
// Make initial storage node selection more conservative
$selectLatency = LatencyReq \times (100\% - PselectLatencyRate)$;
ELSE:
$selectLatency = LatencyReq$;

[P7A]: IF $PadaptiveTimer$ expires:
// Determining a nodes delivery ratio
$timeDelivered$ = 0;
FOR each registered sink visit:
IF visit within $PvisitWindow$:
$timeDelivered$+= $visit.duration$
IF time to previous visit $> selectLatency$:
$timeDelivered$ += $selectLatency$
ELSE:
$timeDelivered$ += time to previous visit
// Calculate the node delivery ratio
$localDelivRatio$ = $timeDelivered/visitWindow$;

[P8A]: IF $PadaptiveTimer$ expires:
// Storage nodes selection/deselection
IF $localDelivRatio >= DeliveryRatioReq$:
$SN = True$;
ELSE:
$SN = False$;
restart PadaptiveTimer

Table 4.1: Pseudocode of Adaptive P-scheme Configuration Algorithm.

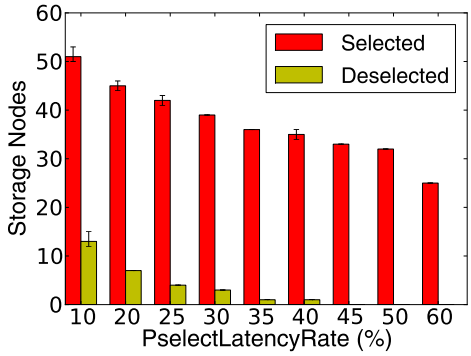
4.1.2 Parameter Sensitivity Analysis

After having described the Adaptive P-scheme algorithm, we now conduct a sensitivity analysis of the two principal algorithm parameters, namely $PselectLatencyRate$ and $PvisitWindow$. To empirically evaluate the impact of these parameters on the performance of the Adaptive P-scheme, we use the same experimental setup described previously in Section 3.1.2.1.

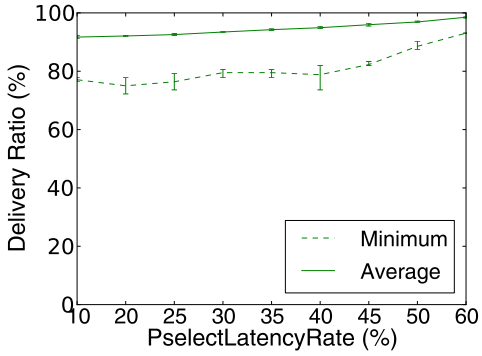
The **PselectLatencyRate** ensures a conservative selection of storage nodes by only selecting storage nodes that sufficiently exceed the application latency requirement. The $PselectLatencyRate$ is defined as the percentage by which a node has to exceed the application latency requirements in order to be selected as a storage node ($selectLatency = LatencyReq \times (100\% - PselectLatencyRate)$).

As the $PselectLatencyRate$ increases, fewer and more stable storage nodes are selected as the $selectLatency$ becomes increasingly more conservative. If set too leniently, storage nodes are selected that in the long term cannot achieve the data requirements, leading to storage nodes being deselected (see Figure 4.4a). If set too conservatively, too few storage nodes are selected leading to longer routing paths, higher data traffic congestion around the few storage nodes and a higher communication cost. Figure 4.4b shows how a low $PselectLatencyRate$ with more unstable storage nodes leads to a lower delivery ratio, while inversely a conservative configuration overachieves the data delivery ratio requirement of 80%. Generally, it is undesirable to have storage node deselections as it greatly increases the chances of data not being delivered on time. However, in this scenario the deselection of a storage node has a minimal effect due to its likely proximity to one of the many remaining storage nodes. In this setting, a $PselectLatencyRate$ of 50% is a reasonable choice since it slightly overachieves the data delivery ratio (of 80%) while offering a near minimal radio duty cycle.

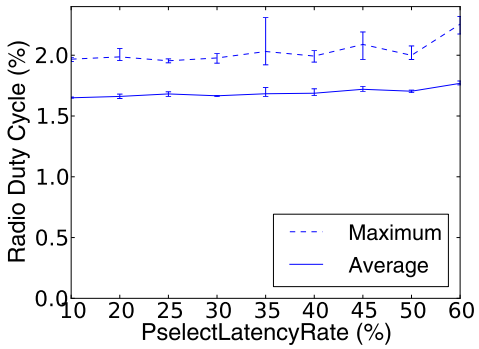
We now examine the impact of **PvisitWindow** on the performance of Adaptive



(a) Storage node selection



(b) Data delivery ratio

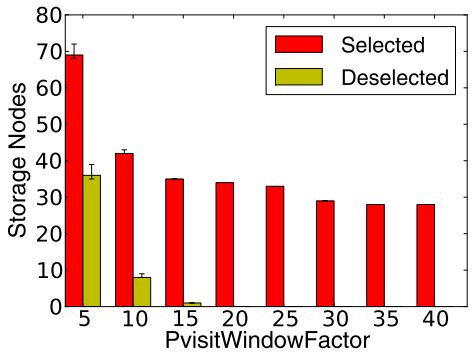


(c) Communication cost

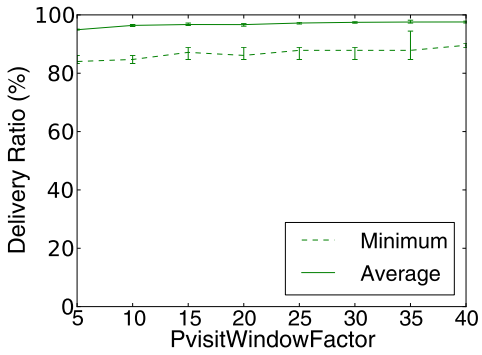
Figure 4.4: The $PselectLatencyRate$ determines by how much a node has to exceed the application latency requirements in order to be selected as a storage node. (a) As we increase the $PselectLatencyRate$, fewer and more stable storage nodes are selected. (b) An increased $PselectLatencyRate$ results in a higher data delivery ratio as the storage node selection becomes increasingly more conservative. (c) Fewer storage nodes lead to a higher communication cost as the routing paths to the storages nodes become longer when fewer storage nodes are available. In this experiment the data requirements were set to $LatencyReq = 200s$ and $DeliveryRatioReq = 80\%$.

P-scheme. The $PvisitWindow$ defines the sliding window within which registered sink visits are used to evaluate the suitability as a storage node. We express $PvisitWindow$ as a factor $PvisitWindowFactor$ of the application latency requirement:

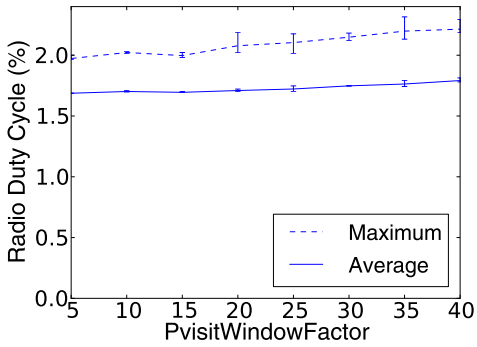
$$PvisitWindow = PvisitWindowFactor \times LatencyReq$$



(a) Storage node selection



(b) Data delivery ratio



(c) Communication cost

Figure 4.5: The $PvisitWindowFactor$ determines the timeframe during which registered sink visits are analysed. (a) As we increase the $PvisitWindowFactor$, the adaptive P-scheme can analyse a larger sample of registered sink visits leading to fewer but more stable storage nodes. (b) A smaller $PvisitWindowFactor$ results in a smaller data delivery ratio. (c) As we increase the $PvisitWindowFactor$ a higher communication cost is incurred due to fewer selected storage nodes that result in longer data routes. Once more the data requirements were set to $LatencyReq = 200s$ and $DeliveryRatioReq = 80\%$.

By using a factor of the application latency requirement, we can configure the algorithm in a less latency dependent manner, as the required sink visit sample size will scale according to the required data latency.

As we increase the $PvisitWindowFactor$, fewer and more stable storage nodes are selected as a larger sample of sink visits is analysed (see Figure 4.5). If the $PvisitWindowFactor$ is too small, an inaccurate storage node selection could oc-

cur due to the number of registered sink visits analysed being too restricted. If set too large, the algorithm will adapt too slowly to change as old and outdated sink visit patterns get to impact the storage node selection decisions.

Figure 4.5 shows these trends for the given scenario. For a configuration of $PvisitWindowFactor$ of 20 or larger no storage node deselections take place. The default value of $PvisitWindowFactor$ is set to 30 in order to ensure a stable storage node selection.

4.2 Adaptive R-scheme

In this section we propose an adaptive version of the R-scheme algorithm that can adjust its behaviour to fit a particular application scenario, similar to the adaptive P-scheme presented in the previous section. The goal of Adaptive R-scheme is to achieve the given data latency and delivery ratio requirements while minimising the communication cost, thereby extending the lifetime of the WSN. The primary configuration parameter for R-scheme is the tree-depth TD, consequently the Adaptive R-scheme algorithm has to find the optimal tree-depth that achieves the trade-off between communication cost and timely data delivery. A node could therefore decide whether the frequency of local sink visits (0-hop) is sufficient or whether multi-hop data collection trees are required to ensure timely data collection. The adaptive algorithm can be split into four tasks.

Firstly, as with Adaptive P-scheme, we assume that from the sink visit history we can predict the future visiting pattern, and accordingly identify the appropriate tree-depth (TD). To do so, each node logs the sink visit times and durations (identical to Adaptive P-scheme), as well as the hop-distance to the sink for multi-hop R-scheme trees.

Secondly, once a node has obtained a sufficient sink visit log, it can analyse the

log and infer the tree-depth the node requires to satisfy the data requirements.

Thirdly, after a node has established its tree depth requirement, it will inform its surrounding nodes of this requirement. Once surrounding nodes know the tree-depth requirements of nearby nodes, they can adjust the tree-depth as required.

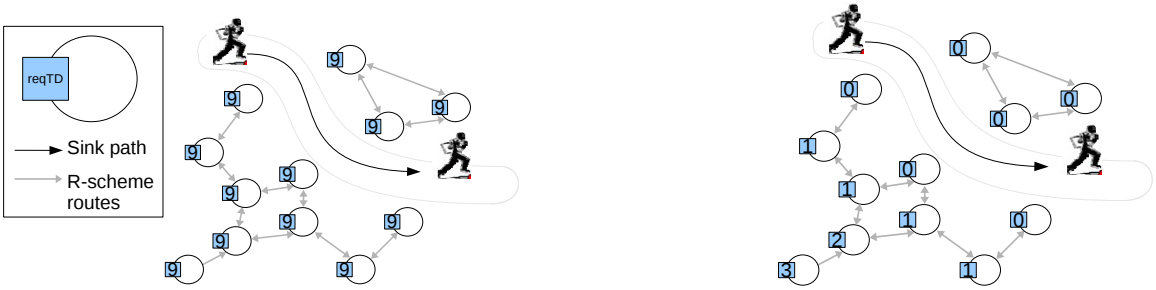
Finally, nodes will have to periodically explore whether the current tree-depth is adequate by periodically increasing the reach of the data collection tree. This exploration is particularly important when the visiting behaviour of mobile sinks changes such that larger trees are required.

In the following section, this approach will be described in more detail and will be illustrated using a pseudo code algorithm implementation.

4.2.1 Algorithm

After having briefly described the four tasks of the Adaptive R-scheme algorithm, we will now present the tasks in more detail, providing pseudo-code as well as example cases. The pseudo-code inherits all the tasks of the Static R-scheme algorithm, of which some are overridden, and adds a couple of new functions (see Table 4.2 and Table 4.3). Overridden functions share the same task number in the original R-scheme (tasks R1-R6), but sport an appended ”A” in the task name to distinguish between static and adaptive implementations of each (e.g. overridden task R1 will be named task R1A). Four new tasks were added: 1) Determining a node’s required tree-depth. 2) Disseminating tree-depth requirements. 3) Tree construction and 4) child tree-depth exploration. Each will now be described in more detail.

Determining a node’s required tree-depth: The R-scheme algorithm adjusts the tree-depth of data collection trees to achieve a trade-off between timely data collection and communication cost (see Section 3.2). Consequently, the aim of Adaptive R-scheme is to find the optimal tree-depth in order to minimise com-



(a) Initial network with a maximum tree-depth of $reqTD = 9$

(b) Nodes having determined their individual $reqTD$

Figure 4.6: Example of Adaptive R-scheme nodes determining their individually required tree depth ($reqTD$) that is needed to fulfil the data requirements. (a) Initially, the adaptive R-scheme trees are built using the maximum tree-depth (in this case the maximum $TD = 9$). This allows the nodes to determine the tree depth that they require to achieve the data latency requirements. (b) Once a sufficient sink visit log has been established, the nodes can determine their required R-scheme tree-depth. Note, how nodes close to the regular sink visit path require a smaller tree-depth than nodes further away.

munication cost while achieving the data requirements. The Adaptive R-scheme tree-depth is determined using a sink visit log, the underlying assumption being that the future visiting pattern will be similar to the past. Each node does this individually, as the number of sink visits varies between different nodes, thereby potentially requiring different tree-depths. The more frequently nodes are visited, the shorter the required tree-depth can be, and vice-versa. Initially, the R-scheme tree-depth $reqTD$ is set to the maximum, allowing the R-scheme trees to cover as many nodes as possible and allowing nodes to establish a comprehensive sink visit log (see Figure 4.6a). This is done by Task R8A that simply logs the time, duration and tree-depth of R-scheme sink visits.

Once a sufficient sink visit log has been established, each node can analyse its visit log to determine whether a specific tree-depth warrants sufficiently frequent visits to allow it to achieve the data latency requirements (as described in Task R9A). It does so by first restricting the sink visit analysis to only local sink visits

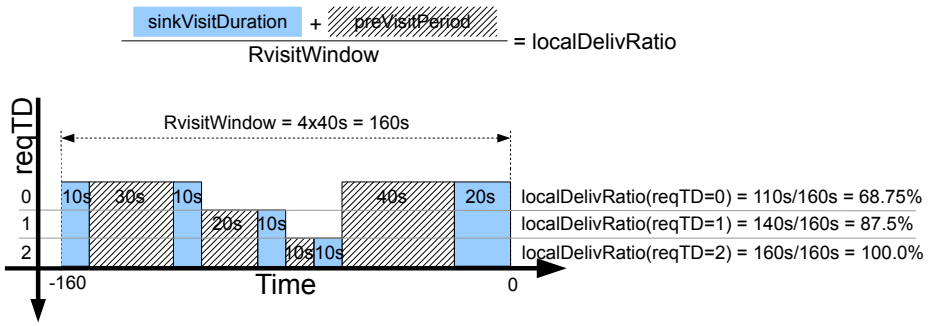
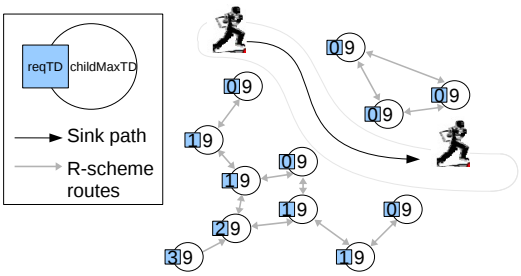
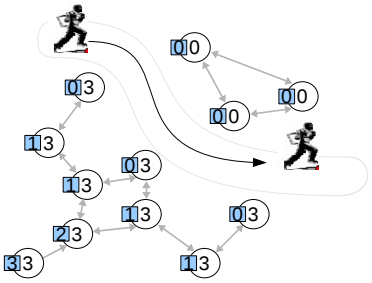


Figure 4.7: Example of Adaptive R-scheme determining a node's tree-depth requirement ($reqTD$) for a data latency of 40 seconds and a $RvisitWindow = 4 \times latency = 160s$. A node's required tree-depth is computed by progressively increasing the tree-depth and verifying whether the achieved delivery ratio is sufficient. The achieved delivery ratio is determined by comparing the time data would have been delivered on time during the $RvisitWindowFactor$ by adding the sinkVisitDuration and preVisitPeriods. In this example, A tree depth $reqTD = 0$ achieves a $localDelivRatio = (10s + 30s + 10s + 40s + 20s)/160s = 68.75\%$. If the delivery ratio requirement is 90%, a tree-depth $reqTD = 1$ could have only achieved a $localDelivRatio = 87.5\%$, while a tree-depth $reqTD = 2$ is sufficient as it achieves a $localDelivRatio = 100\%$ due to the frequent multi-hop sink visits.

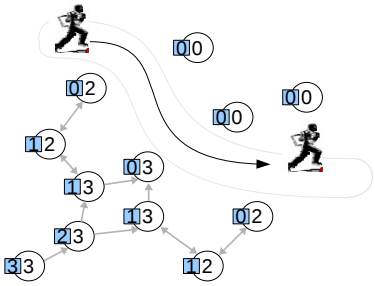
(tree-depth $reqTD = 0$). If the local sink visits are not frequent enough to achieve the data requirements, the number of sink visits taken into account is increased by considering all sink visits that have a smaller or equal tree-depth of $reqTD = 1$. The application latency used during the analysis is set more conservatively by the $RlatencyFactor$ in order to ensure a timely data delivery. Figure 4.7 describes an exemplary calculation of the required tree-depth. This is done until a tree-depth is found with sufficiently frequent visits to achieve the application latency requirements, or the maximum viable R-scheme tree-depth is reached. Figure 4.6b shows the result of all nodes having determined their individual required tree-depth $reqTD$.



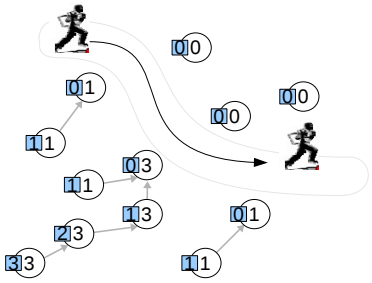
(a) Initial network with $childMaxTD = 9$ set to the maximum value. The nodes have already determined their own required tree-depth $reqTD$



(b) Nodes adjust the $childMaxTD$ to the maximum $childTD = 3$ (the children's $reqTD$) values gathered in the first iteration.



(c) Nodes restrict the tree construction according to the previously set $childMaxTD$, and further adjust the value according to the $childTD$ values within their previous $childMaxTD$ range.



(d) The Adaptive R-scheme algorithm finally settles at its optimal tree-depth configuration after determining the $childMaxTD$, respectively.

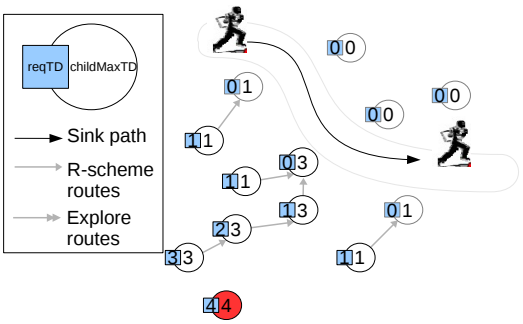
Figure 4.8: Example of Adaptive R-scheme nodes disseminating their tree-depth requirements, thereby adjusting the maximum required tree depth for child nodes ($childMaxTD$). Nodes piggyback their own $reqTD$ as $childTD$ along the routes, each parent taking note of the children's maximum required tree-depth ($childMaxTD$). Note how the trees (as induced by $childMaxTD$) progressively become smaller, as only the locally relevant tree-depths are taken into consideration. The data collection trees therefore iteratively become smaller until the final configuration is reached.

Disseminating tree-depth requirements: After a node has established its tree depth requirement, it will inform its surrounding nodes of the R-scheme tree-depth the node requires to satisfy the data requirements. The surrounding nodes can then ensure that the tree-depth of data collection trees is sufficiently large to achieve the data requirements.

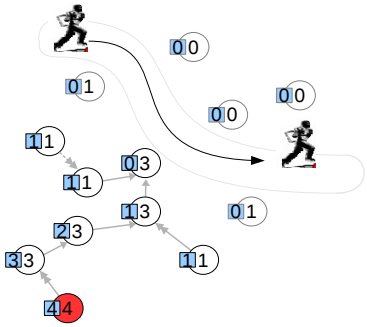
A node informs its surrounding nodes of its tree-depth requirement $reqTD$ by piggybacking it to data packages that are sent along the R-scheme trees, thereby informing all its parent nodes (as described in Task R9A in Table 4.3). Parent nodes that receive children's tree-depth requirements (named $childTD$ in Task R10A) keep track of the maximum tree-depth requirement, namely $childMaxTD$ as depicted in Figure 4.8b. The child's maximum tree-depth requirement $childMaxTD$ is the maximum tree-depth required by the surrounding nodes to achieve the data requirements. If the maximum tree-depth $childMaxTD$ were to be ignored, the trees would not reach all the surrounding nodes, potentially leading to data not being delivered on time. The adherence to the $childMaxTD$ is therefore paramount to allow surrounding nodes to deliver their data in a timely manner.

Tree construction: Once a node has determined its children's maximum tree-depth requirement $childMaxTD$, it can adjust the tree-depth of its constructed data collection trees accordingly, allowing it to save data communication cost while still achieving the data requirements. This is done by task (R1A) that initiates the tree construction when a sink moves into the vicinity of a fixed sensor node (see Task R1A in Table 4.3).

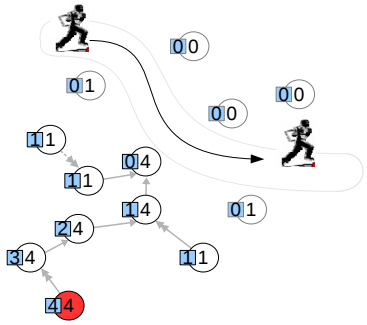
As these tree-depth requirements are subject to change, a node will reset the tree depth required by its neighbours after initiating the tree construction ($childMaxTD = 0$) in order to reassess the tree-depth requirement. More precisely, the children's tree-depth requirements are re-collected after every data collection tree, allowing a node to obtain up-to-date tree-depth requirements. An ex-



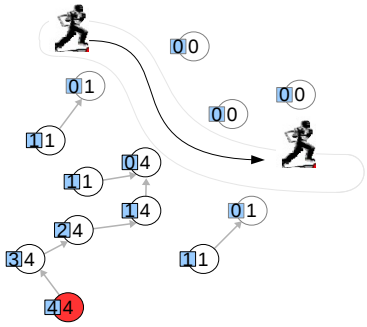
(a) A new node emerges (highlighted) that needs to join the existing R-scheme trees in order to fulfil the data latency requirements. The current $childMaxTD = 3$ is insufficient for it to be reached.



(b) At a given frequency $RexploreRate$, trees will explore their neighbours, as indicated by the "Explore routes" arrow, in order to verify that the current tree-depth is sufficient to cover all nodes. This exploration allows the R-scheme tree to reach the new node with $reqTD = 4$.



(c) The new node can notify its parents that it needs to join the R-scheme tree, allowing parent nodes to register the updated maximum child tree-depth of $childMaxTD = 4$.



(d) The ancestor nodes register the new $childMaxTD = 4$ and subsequent trees are built using a deeper tree-depth such that the node is now also covered.

Figure 4.9: Example of an Adaptive R-scheme tree exploring whether the current $childMaxTD$ is sufficient to cover all potential child nodes. An additional node with $reqTD = 4$ is shown to illustrate how the exploration phase allows the tree to expand accordingly. Note how in (d) nodes with a low $childMaxTD = 1$ ignore suboptimal trees with $childMaxTD = 4$, allowing for asymmetric tree-construction.

ample of continual re-collection and subsequent adjustment of the tree-depth can be seen in Figure 4.8, in which the tree-depth is continually reassessed until a final stable state is reached.

Child tree-depth exploration: By continually adjusting the data collection trees (by restricting the tree-depth to $childMaxTD$), Adaptive R-scheme runs the risk of potentially ignoring nodes or being unable to adapt to changes in mobility that

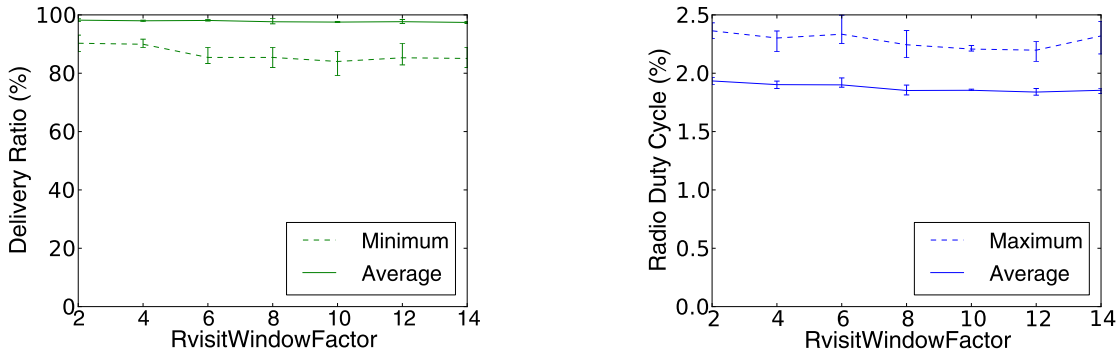
require a larger tree-depth. To counter this, the tree-depth requirements of the surrounding nodes are explored more aggressively at a given $RexploreRate$. This is done by simply increasing the tree-depth of the data collection trees by one hop, thereby also gathering data from nodes further away (see Task R1A in Table 4.3). The process of exploring surrounding nodes tree-depth requirements is illustrated by example in Figure 4.9.

A final enhancement of the adaptive R-scheme tree is conducted by that nodes forward R-scheme beacons based on their own $childMaxTD$, rather than the $childMaxTD$ of the root of the tree. This allows asymmetrically sized trees that extend further into areas with low visiting frequencies, and restricts unnecessary tree construction in frequently visited areas, as depicted in Figure 4.9d.

4.2.2 Parameter Sensitivity Analysis

After having described the Adaptive R-scheme algorithm, we show how its performance depends on the main configuration parameters. Three such parameters exist: 1) The $RvisitWindowFactor$ parameter which defines the sliding window size during which sink visits are used to determine a node's tree-depth ($reqTD$). 2) The $RexploreRate$ parameter that determines how often the tree-depth will temporarily be increased to explore the neighbours tree-depth requirements. 3) The $RlatencyFactor$ sets a stricter, more conservative, latency in order to ensure a timely data delivery. Each parameter and their respective impact on algorithm performance will now be explored in more detail. The simulation setup for each of the parameters remains the same as described previously, with 64 static sensor nodes laid out in a 8x8 grid as well as 3 mobile sinks (see Section 3.1.2.1).

The **RvisitWindow** parameter determines how large the window of the sink visit log should be when determining a node's required tree depth ($reqTD$). The $RvisitWindowFactor$ is defined as a factor of the data latency, making it indepen-



(a) Data delivery ratio

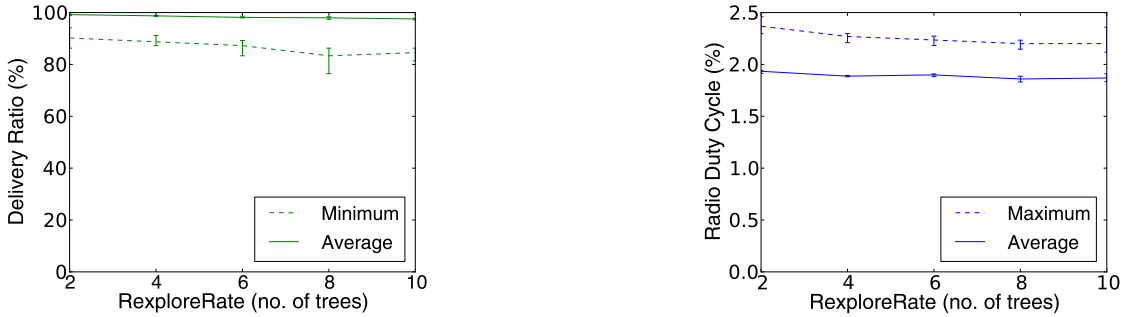
(b) Communication cost

Figure 4.10: Impact of $RvisitWindowFactor$ on Adaptive P-scheme. a) The $RvisitWindowFactor$ has a marginal negative effect on the delivery ratio when becoming large, as the reactive nature of the data collection algorithm allows it to adapt to change better when using a small window size. b) Likewise, a larger $RvisitWindowFactor$ results in a marginally lower communication cost as the tree-depths are more accurately predicted. The default configuration was chosen to be $RvisitWindowFactor = 4$ as it offered the highest delivery ratio while reducing the communication cost.

dent of data requirements, similar to P-scheme's $PvisitWindowFactor$:

$$RvisitWindow = RvisitWindowFactor \times LatencyReq = 4 \times 100s = 400s$$

If configured too large the adaptive configuration becomes sluggish as it takes too long to adjust to changes. If set too small, the configuration will be based on too little information potentially leading to a misjudgement. R-scheme can quickly adjust to changes as the trees are frequently built and destructed, unlike P-scheme, allowing the $RvisitWindowFactor$ to be relatively small. Figure 4.10a shows how the delivery ratio is affected by the $RvisitWindowFactor$, for which a smaller factor achieves a minimally higher delivery ratio due to R-scheme being more responsive to change. Figure 4.10b shows how the radio duty cycle varies when changing the $RvisitWindowFactor$. The configuration was set to $RvisitWindowFactor = 4$ to allow R-scheme to quickly adjust to changes in mobility patterns while ensuring



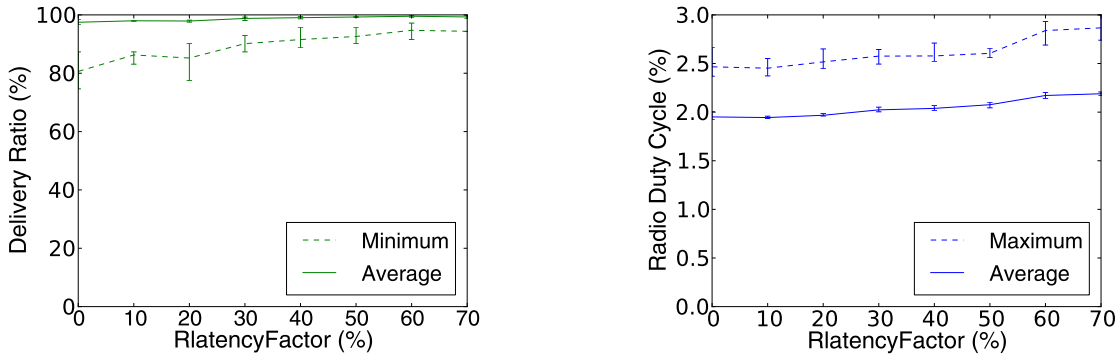
(a) Data delivery ratio

(b) Communication cost

Figure 4.11: $RexploreRate$ parameter configuration for the adaptive R-scheme algorithm. The shorter the interval at which the tree-depth is temporarily increased to scope neighbours' requirements, the higher the delivery ratio and the higher the radio duty cycle. A configuration of $RexploreRate = 4$ was chosen as the radio duty cycle gain at a higher value was marginal, while reliably achieving the delivery ratio of 80%.

energy-efficient and reliable data delivery.

The **RexploreRate** describes the frequency at which the tree-depth is temporarily increased by 1-hop to explore whether neighbouring nodes require a higher tree-depth than is currently selected, possibly induced by changing mobility patterns. By increasing the tree-depth, neighbouring nodes are given a chance to piggyback their tree-depth requirements, thereby allowing parent nodes to register their children's tree-depth requirements. The registered requirements will then be used to determine the tree-depth for future tree-constructions. The exploration phase is executed every $RexploreRate$ tree constructions, the number of tree constructions therefore being its unit. If the $RexploreRate$ parameter is set too large, the exploration phase is executed very infrequently, potentially allowing R-scheme to not explore surrounding nodes frequently enough and leading to an insufficiently low delivery ratio (see Figure 4.11a). If set too small, the frequent exploration will lead to a high radio duty cycle as the trees are extended unnecessarily (see Figure 4.11b). A reasonable configuration trade-off was found to be



(a) Data delivery ratio

(b) Communication cost

Figure 4.12: Impact of $RlatencyFactor$ on the Adaptive R-scheme algorithm. The larger and more conservative the $RlatencyFactor$, the larger and more conservative the tree-depth. This in turn leads to a larger communication cost and a higher delivery ratio. The default value of $RlatencyFactor$ was set to 40%.

in $RexploreRate = 4$ as it handily achieves the data delivery ratio of 80% while sufficiently minimising the radio usage.

The **RlatencyFactor** ensures the tree-depth is set more conservatively in order to ensure a timely data delivery. A node's required tree-depth ($reqTD$) is determined by analysing the sink visit history with regards to the latency requirement. Recall that the latter is set more conservatively by the $RlatencyFactor$. By using a larger and more conservative $RlatencyFactor$, the nodes required tree-depth $reqTD$ will be set to a more conservative and larger value, thereby resulting in larger trees being constructed. This in turn increases the communication cost while also increasing the data delivery ratio. Conversely, if the $RlatencyFactor$ is set too small, the trees will be insufficiently sized and potentially lead to the nodes not achieving the desired data delivery ratio requirements. Figure 4.12 shows this trade-off. The configuration was set to $RlatencyFactor = 40\%$ as the communication cost was increased by a small amount while ensuring a timely data delivery.

INPUT VARIABLES:
$DeliveryRatioReq$ - Application Delivery Ratio Requirement;
$LatencyReq$ - Application Latency Requirement;
$RvisitWindow$ - Sliding window size for sink visits;
$RexploreRate$ - R-scheme adaptive requirement exploration intervals;
$RlatencyFactor$ - Factor by which the latency is made more conservative;
LOCAL VARIABLES:
$visits$ - List of R-scheme visits;
$reqTD$ - A node's tree depth requirement;
$childMaxTD$ - Maximum tree depth requirements for child nodes;
$exploreCount$ - Tracks when a tree-depth exploration should be initiated;

// Logs sink visits
[R7A] IF $sink$ within range or receive R beacon:
  register time, duration and tree-depth in $visits$;
  IF RadaptiveTimer not running:
    start RadaptiveTimer

// Re-evaluates a node's required R-scheme tree-depth
[R8A] IF $RadaptiveTimer$ expires:
  $reqTD = 0$; $localDelivRatio = 0$;
  // Search for reqTD that achieves the data delivery ratio
  WHILE $reqTD < maxTD$ **AND** $localDelivRatio < DeliveryRatioReq$:
    // Calculate delivery ratio for RvisitWindow
    $timeDataDelivered = 0$;
    FOR each registered sink $visit$ **WHERE** $visit.treedepth <= reqTD$:
      IF sink visit within $RvisitWindow$:
        // Calculate time sink visit would have contributed to timely data delivery
        $timeDataDelivered$ += $visit.duration$
        // Make latency more conservative to ensure timely delivery
        $conservativeLatency$ = $LatencyReq * (100\% - RlatencyFactor)$
        IF time to previous visit $> conservativeLatency$:
          // Only packages in last $conservativeLatency$ are delivered on time
          $timeDataDelivered$ += $conservativeLatency$
        ELSE:
          // Whole intervisit timespan delivers packages on time
          $timeDataDelivered$ += time to previous visit
    // Node delivery ratio based on timely data delivery within RvisitWindow
    $localDelivRatio = timeDataDelivered/RvisitWindow$;
    IF $localDelivRatio < DeliveryRatioReq$:
      // Node delivery ratio insufficient. Keep searching with higher tree-depth
      $reqTD$++
  restart RadaptiveTimer

Table 4.2: Pseudocode of adaptive R-scheme functions that register sink visits (R7A), compute the node's required tree-depth $reqTD$ (R8A).

INPUT VARIABLES:
$DeliveryRatioReq$ - Application delivery ratio requirement;
$LatencyReq$ - Application latency requirement;
$RvisitWindow$ - Sliding window size for sink visits;
$RexploreRate$ - R-scheme adaptive requirement exploration intervals;
$RlatencyFactor$ - Factor by which the latency is made more conservative;
LOCAL VARIABLES:
$visits$ - List of R-scheme visits;
$reqTD$ - A node's tree depth requirement;
$childMaxTD$ - Maximum tree depth requirements for child nodes;
$exploreCount$ - Tracks when a tree-depth exploration should be initiated;

// Piggybacks a node's tree-depth requirement to its parent nodes
[R9A] IF $parentR \neq ID$:
piggyback $reqTD$ as $childTD$ using data to $parentR$;

// Registers and forwards child nodes tree-depth requirements
[R10A] IF receive $childTD$:
// Check whether the required tree depth is sufficient
IF $childTD > childMaxTD$:
$childMaxTD = childTD$;
IF $parentR \neq ID$:
forward $childTD$ piggybacked to data to $parentR$

// Initialises tree construction
[R1A] IF mobile sink comes in range:
$TD = childMaxTD$; $exploreCount$++;
// Explore not-covered neighbour requirement tree-depth (TD)
IF $exploreCount == RexploreRate$:
// Increase tree-depth (TD); reset explore count
TD += 1; $exploreCount = 0$
// Initiate R-scheme tree construction
call **R1**;

// Forwards and trims tree construction beacons
[R2A] IF R beacon b received:
IF $b.distMax > childMaxTD$:
// Restrict maximum tree depth size to local requirements
$b.distMax = childMaxTD$
// Propagate R-scheme tree construction
call **R3**

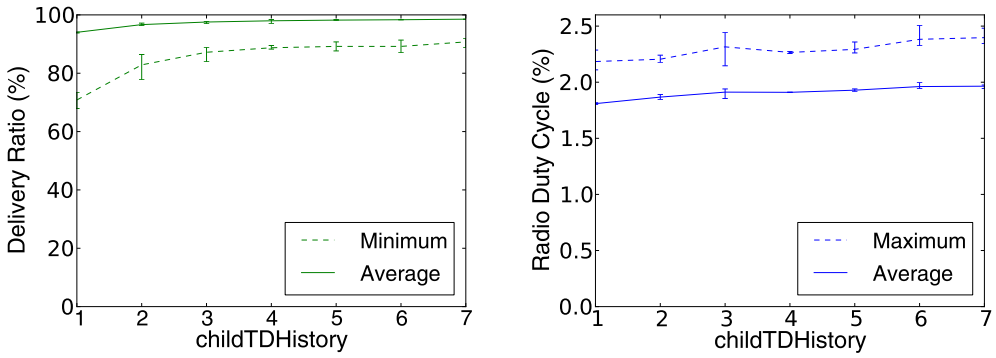
Table 4.3: Pseudocode of adaptive R-scheme functions that that disseminate and register tree depth requirements (R9A, R10A), initiate and propagate data collection trees (R1A, R2A).

4.2.3 Implementation Details

In this section we describe implementation details of the Adaptive R-scheme algorithm that were omitted from the previous description as they are only of importance when implementing the algorithm. In particular, we explain how the Adaptive R-scheme algorithm deals with child nodes' failure to notify parents of their required tree-depth ($childTD$).

Recall that in Section 4.2 we described the Adaptive R-scheme algorithm in which each node analyses its sink visit history and therefrom deducts its own required tree-depth ($reqTD$). The surrounding nodes are then informed about each other's required tree-depth by piggybacking the tree-depth requirements to data messages, allowing all parent nodes en route to register the $childTD$ requirements. While implementing the Adaptive R-scheme algorithm, it became apparent that parent nodes often were not notified of all $childTD$ requirements due to transmission errors. If parent nodes miss some $childTD$ information they are likely to be unaware of the real required tree-depth. An insufficiently sized data collection tree will result in potentially not achieving the application requirements.

A simple solution to temporarily missing child tree-depth requirements was to not only record the latest tree-depth requirements, but also store the previous tree-depth requirements. The history of the previous tree-depth requirements is retained to deal with the possibility of child nodes failing to inform their parents. The larger the $RchildTDHistory$ the more requirement tree-depths are stored, leading to a more conservative (larger) tree-depth which in turn leads to a higher delivery ratio (as shown in Figure 4.13). By retaining previous, potentially more conservative, child tree-depth requirements ($childTD$) the constructed data collection trees are likely to have a larger tree-depth, leading to a higher communication cost (see Figure 4.13b). For this scenario, the $RchildTDHistory$ was set to 5 as it reliably achieved the data delivery ratio requirements of 80%.



(a) Data delivery ratio

(b) Radio duty cycle

Figure 4.13: Impact of the $childTDHistory$ parameter on the Adaptive R-scheme algorithm. If the $childTDHistory$ is too small the data delivery ratio requirement of 80% is not achieved, due to insufficient past tree-depth requirements ($childMaxTD$) being retained. The default value of $childTDHistory$ was set to 5.

4.3 Summary

In this chapter, we presented and described the behaviour of adaptive versions of the P- and R-scheme algorithms. At first, we described the Adaptive P-scheme algorithm in which nodes self-select themselves as storage nodes if they have been visited frequently enough, assuming that we can predict future sink visiting patterns from the past. Following this we conducted a sensitivity analysis of Adaptive P-scheme parameters. By selecting reliable storage nodes, Adaptive P-scheme can adapt and change according to given data latency or delivery ratio requirements or sink visiting patterns.

We also presented the Adaptive R-scheme algorithm in which nodes determine their required tree-depth to achieve given data requirements. Neighbouring nodes are then informed of the tree-depth requirements, resulting in data collection trees being built according to the local requirements. A sensitivity analysis of Adaptive R-scheme parameters was conducted in order to investigate and optimise its be-

haviour.

The purpose of both adaptive algorithms is to enable the network to self-configure itself according to network conditions, such as the data latency and delivery ratio requirements as well as the sink visiting pattern. The adaptive algorithms automatically reconfigure themselves whenever such network conditions change, allowing for a constant optimisation of the network. Another aim of the adaptive algorithms is to minimise the effort required by network engineers to deploy a WSN, thereby reducing the margin of error by self-configuring the most crucial algorithm parameters, namely the storage node selection for P-scheme and the tree-depth for R-scheme. While the adaptive algorithms require configuration of their parameters themselves, they are of a more simplistic nature and have a reduced impact than the primary P- and R-scheme parameters.

In conclusion, the adaptive algorithms will allow self-configuration and self-adjustment to changing network conditions while reducing the complexity of deploying a WSN.

Chapter 5

Hybrid Routing Protocol (TwinRoute)

In the previous chapter we presented a proactive and a reactive algorithm for data collection. In particular, P-scheme is designed to exploit long-term historical information about sink visits and elect relatively stable storage nodes to which sensors proactively push their data. In contrast, R-scheme is designed to react to real-time sink movements and allows a sink to pull data from nearby sensors from their neighbourhood. We are now in a position to present *TwinRoute*, a hybrid algorithm that combines the proactive and reactive data collection techniques with the aim of taking advantage of each of their respective strengths.

Proactive and reactive data collection algorithms are inherently suited to different kinds of environments (see Table 5.1). Proactive data collection thrives when the environment is stable and sink mobility or application requirements remain unchanged, as changes may temporarily result in unfavourable storage node deselections. In contrast to this, reactive algorithms can adapt with relative ease to changing environments as the data collection trees are frequently re-constructed.

Proactive and reactive schemes also exhibit differing characteristics with re-

Environment Properties	P-scheme	R-scheme
Spatial visit distribution	Concentrated	Distributed
Change in sink mobility	Slow adaptation	Quick adaptation
Change of application requirements	Slow adaptation	Quick adaptation

Table 5.1: Suitable environments for proactive and reactive data collection algorithms

gard to environmental factors such as the sink visiting frequency and geographic distribution of visits (see Table 5.1). Proactive schemes require frequent visits at specific storage nodes in order to achieve the application requirements, namely they benefit from concentrated sink visits. Reactive schemes, however, only require sufficiently frequent visits in the broader area in order for data to be collected on time, in other words reactive schemes perform well when the sink visits are distributed.

The suitability for different types of environments (as described in Table 5.1) suggests that a joint algorithm will have the ability to adapt to a wide range of different scenarios. In short, proactive algorithms thrive when many storage nodes can be selected in a stable manner due to frequent and concentrated sink visits. Reactive algorithms exert their strengths when the sink visits are distributed or when change of the application requirements or sink mobility patterns occur. By designing a hybrid data collection algorithm, the pure strengths can potentially be combined while avoiding their individual weaknesses. The following section will describe how proactive and reactive algorithms can be combined to form the TwinRoute algorithm.

5.1 TwinRoute Algorithm

TwinRoute is essentially a parallel merge of Adaptive P-scheme and Adaptive R-scheme. In other words, each sensor maintains both the sink visit history required

by proactive routing, as well as tracks each sink as it comes in and out of range as specified by reactive routing. In addition, each sensor also listens to and broadcasts beacons for both schemes. As a result, a network can exploit both relatively stable and often large routing trees created by P-scheme, and relatively transient and small routing trees created by the R-scheme. Each sensor is thus always part of a proactive tree, whilst it is occasionally presented with the choice to dynamically participate in a reactive tree whenever a mobile sink appears in its vicinity.

Typically, great advantages can be gained over a pure proactive scheme when a mobile sink unexpectedly appears near sensors that are many hops away from the closest storage node. In TwinRoute, sensors have the opportunity to temporarily switch to the reactive routing tree and offload their data on much shorter paths. On the other hand, TwinRoute will demonstrate its superiority over a pure reactive scheme as long as there are areas in the network where mobile sink visits are consistently concentrated. Since we can be confident that mobile sinks will always revisit these areas in the future, there is little need to demolish routes once they are established (as in a pure reactive scheme).

TwinRoute reuses most parts of the Adaptive R- and P-schemes to handle mobile sink visits and tree construction beacons. A critical change is made in the way reactive beacons are disseminated to build R-scheme trees. An additional condition $b.dist + SP < distP$ is enforced in the tree construction step of Adaptive R-scheme (step $R2A$), where SP is a scheme preference parameter that regulates by how many hops the reactive tree has to be shorter than the proactive tree. When $SP > 0$, a sensor becomes part of reactive tree only if its distance to the sink on that tree ($b.dist$) is shorter (by at least SP hops) than its distance to a P-scheme storage node ($distP$). That is, the scheme preference value SP denotes the relative distance gain necessary to switch from P-scheme to R-scheme.

In addition to the scheme preference SP distance gain, we restrict Adaptive

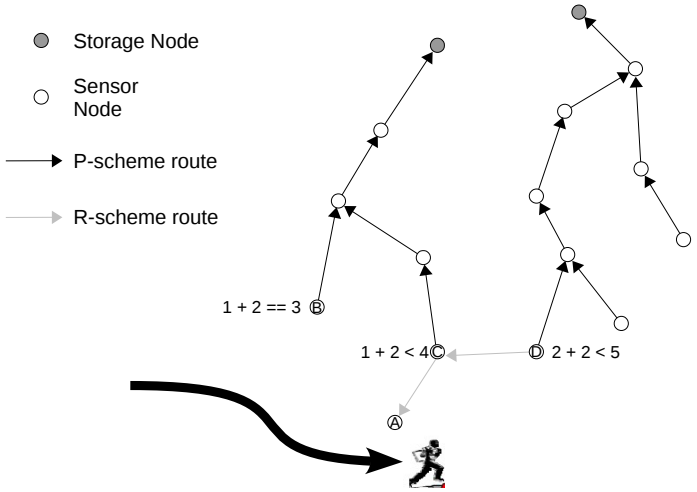


Figure 5.1: TwinRoute in action with $SP = 2$ in an example sensor network. As the mobile sink passes the static sensor nodes, an R-scheme tree is constructed only when the distance to the sink is at least 2 hops better than the P-scheme tree (nodes A, C and D). Conversely, node B does not participate in a R-scheme tree as the node is closer to its storage node. Note how the reactive tree-depth TD is always smaller than 3 hops.

R-scheme trees to a maximum tree-depth of $TD = 3$ when P-scheme trees are present, in order to increase the efficiency of data collection. A higher R-scheme tree-depth TD results in less efficient data collection trees, as multi-hop collection incurs a higher communication cost. By restricting the tree-depth, we increase the amount of data being collected with lower tree-depths, resulting in more efficient R-scheme data collection.

Figure 5.1 showcases the critical process of a switch to R-scheme when $SP = 2$. As a mobile sink travels by, node A becomes the root of a reactive tree and broadcasts R beacons. Nodes B, C and D are currently part of two separate P trees. The beacons fire the scheme switching condition on nodes C and D but not on node B.

The second subtle point in combining P-scheme and R-scheme concerns the data-forwarding step. Once a sensor has switched to become part of a reactive tree, it always favours R-scheme when reporting data. All buffered data is sent

along branches of the reactive tree, if one exists, to take full advantage of the existing routes to the sink. In the absence of a reactive tree, a sensor returns to using the routes of its proactive tree. However, unlike the pure P-scheme, data is not forwarded in a zero-wait mode, but in a delay-tolerant manner. Not forwarding data items as soon as they are generated at the source node ensures that there will be some data to capitalise on a switch to R-scheme should it happen. In order to do this, when storage nodes initiate tree construction around them (in the P-scheme), they include their storage delays in the tree construction beacons. The time that a message can be buffered locally at the source node must not exceed its data latency threshold, minus the storage delay at the P-scheme storage node.

Since TwinRoute combines the proactive and reactive schemes, it inherits two important parameters from these schemes: 1) the selection of storage nodes, which is inherited from Adaptive P-scheme, and 2) the tree depth (TD) of reactive trees formed dynamically around roaming sinks, which is inherited from Adaptive R-scheme. By carefully tuning the scheme preference parameter (SP), we can come up with different flavours of TwinRoute, to suit the requirements of a particular application.

5.2 Algorithm Parameter Sensitivity Analysis

After having described the TwinRoute algorithm, which essentially combines Adaptive P- and R-scheme algorithms, we now describe the sensitivity analysis of the main TwinRoute algorithm parameter, namely the scheme preference (SP). Note that the remaining parameters of the adaptive algorithms are set to their default values, as specified in Chapter 4.

In order to assess the effect of changing the scheme preference SP, we utilise a special scenario that allows ample use of the R-scheme data collection. We

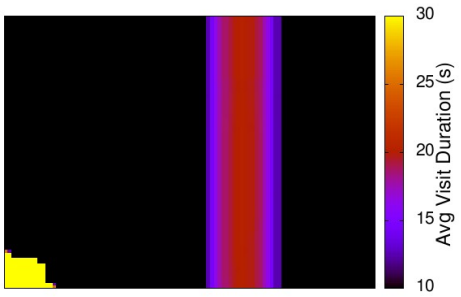
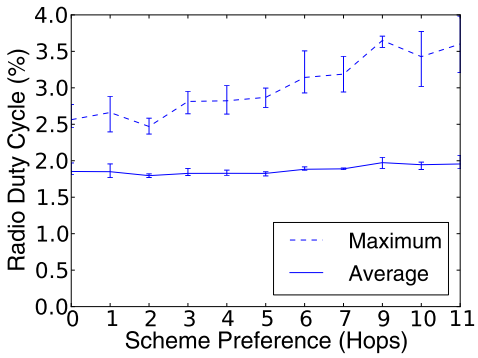


Figure 5.2: Illustration of movement of mobile sink in scenario using one gateway node as well as two mobile sinks that follow a pathway at a speed of 5m/s.

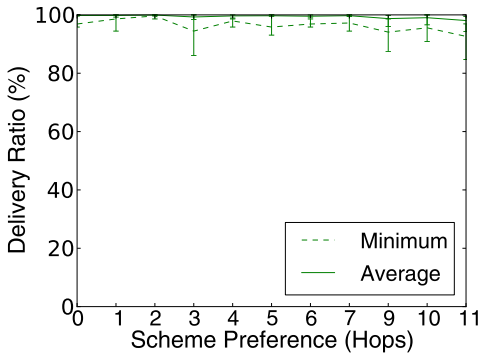
use the same basic scenario as previously (see Section 3.1.2.1), however restrict the sinks to roaming on a given pathway at 5m/s through the sensor network, as well as a gateway node that is placed at the boundary of the deployment (see Figure 5.2). Storage nodes are only selected within range of the gateway node due to the visiting frequency of the roaming sinks being insufficient for further storage node selection. The application requirements remain unchanged with a latency of 200s and a delivery ratio of 80%.

The impact of varying the scheme preference SP parameter is shown in Figure 5.3a and b. As we increase the scheme preference SP, the maximum radio duty cycle increases significantly due to the increase in routing traffic via the gateway node. This is due to a higher scheme preference resulting in less use of R-scheme and routing more data via the gateway node. This is further illustrated in Figure 5.4 which depicts the data delivery and transmissions between the different routing schemes. As the scheme preference parameter SP is increased, a higher radio duty cycle is incurred as more data is routed through the gateway node. Conversely, the lower the scheme preference SP, the more data is offloaded to mobile sinks resulting in a lower routing load at the storage / gateway nodes.

Notably, when having a scheme preference of 10 or higher, R-scheme data



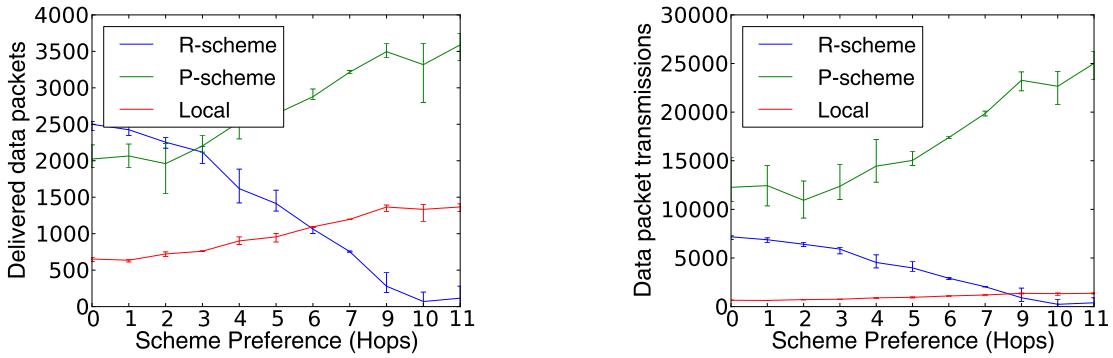
(a) Communication Cost



(b) Data Delivery Ratio

Figure 5.3: Impact of scheme preference (SP) on the radio duty cycle and delivery ratio of the fixed sensor network. As the scheme preference parameter SP is increased, a higher radio duty cycle is incurred as more data is routed through the gateway node. Conversely, the lower the scheme preference SP, the more data is offloaded to mobile sinks resulting in a lower routing load at the storage / gateway nodes. This marginally effects the average radio duty cycle, while having a significant impact on the maximum radio duty cycle as a lower scheme preference reduces the hotspot problem. In this scenario, a reasonable value for SP is 0, since it achieves the required latency and delivery ratio, with the minimum radio duty cycle.

collection is close to zero and P-scheme data collection reaches a maximum. In this scenario, a reasonable value for SP is 0, since it allows to offload traffic from the already existing hotspots surrounding the gateway node.



(a) Scheme Data Delivery

(b) Scheme Data Transmissions

Figure 5.4: Data transmission distribution between different data collection schemes when varying the scheme preference SP parameter. As we increase the scheme preference SP, TwinRoute discriminates more against R-scheme, resulting in less data being collected using R-scheme and more data being routed through P-scheme. By selecting a low scheme preference SP, the hotspot problem at nodes surrounding the gateway node is eased, resulting in a lower radio duty cycle. (A data transmission is a one hop transfer of a data packet).

5.3 Summary

In this chapter, we described the design of a hybrid data collection algorithm, namely TwinRoute. At first, we reasoned about the scenarios that favour proactive and reactive algorithms, and those that could benefit from combining the two routing schemes into a hybrid proactive-reactive scheme. Proactive algorithms work well in stable environments with concentrated visits such that storage nodes can be reliably selected. Reactive algorithms have strengths when the sink mobility is distributed and changes to the WSN environment, such as application requirements or sink mobility patterns, are frequent. Hybrid algorithms promise to combine the strengths of the pure algorithms. By tuning the scheme preference parameter of TwinRoute, an engineer is able to strike a good balance in the parallel use of proactive and reactive trees. The higher the scheme preference parameter, the less extended the use of reactive trees, and the higher the volume of data routed through

the proactively selected storage nodes. In Chapter 7, TwinRoute will be evaluated in a variety of scenarios and will be compared with the pure adaptive routing protocols. But, first, Chapter 6 puts these experimental results in context, by providing an overview of the system implementation and the simulation environment.

Chapter 6

Implementation and Setup

In this chapter we describe all the underlying details and choices with regards to the implementation as well as the testing of the algorithms. The main aim of this chapter is to describe the factors that influenced the results and allow potential researchers to reproduce them.

At first, we describe the configuration choices made within the WSN operating system, namely Contiki OS. Secondly, we describe the experimental setup. Finally, we describe the implementation architecture of the routing algorithms.

6.1 WSN Operating System

In this section we describe the operating system and the implementation and configuration choices thereof. Contiki OS is the operating system upon which the implementation of routing algorithms was conducted, it being an open source, single threaded WSN operating system that is highly customisable. Contiki OS, as a WSN operating system, requires minimal amounts of memory and is very energy efficient in order to elongate the life-time of the WSN.

Contiki OS uses native C code, allowing developers to easily adopt and utilise

the Contiki OS features. Amongst the advantageous features of Contiki OS are protothreads [9], lightweight non-preemptive threads that utilises a minimum amount of RAM (2 bytes). Protothreads are stackless and require little overhead to switch between threads. The stackless implementation results in thread specific data not being persistent when switching between protothreads, therefore forcing developers to store persistent data in global static variables. While this obstacle might sound significant, its effect is minimal when fully understood. Careful consideration has to be made such that computations do not take exceedingly long, as the non-preemptiveness of the protothreads restricts the automated scheduling capabilities.

WSN development needs to be optimised for energy efficiency, which typically is achieved by putting the sensor nodes into low-energy sleep mode as often as possible. Contiki OS offers two methods of managing the control flow in an energy efficient manner: 1. event based execution and 2. timer based execution. Both allow nodes to only become active when actually needed, instead of remaining in a high-energy waiting loop until events occur or code needs executing.

The use of Contiki was further motivated by the openly available and supported communication protocols, such as the Contikimac protocol and the uIP IPv6 network stack. In the following subsections, we provide an overview of Contiki's tools and protocols that were used for the experiments, including the Cooja network simulator, Contiki's network stack and the Contikimac protocol.

6.1.1 WSN Simulator

Contiki features a WSN simulator that was used to obtain all simulation results, namely Cooja. Cooja [37] is a cross-level WSN simulator that simulates the whole or parts of the software stack of a sensor node by emulating the hardware behaviour as well as simulating connectivity between sensor nodes. Cooja utilises

the mspsim library to emulate the hardware of the Tmote sky sensor node, the primary sensor node used during the WildSensing project. The ease of configuration through Cooja's optional GUI as well as the interaction with the simulation, for example by adjusting the position of a sensor node or inspecting run-time variables, allow for an efficient development process.

The main platform of the WildSensing project (which motivated the work in this thesis) was the TmoteSky sensor nodes. These feature a MSP430 microprocessors, which Cooja supports through the mspsim instruction set emulator. Consequently, all simulations used the mspsim emulator allowing us to gather results that closely emulate hardware based results, without having to deploy the code using real hardware. The downside of the high accuracy is a limitation by the increased computational complexity, which, in practise, limit simulations to contain fewer than 100 nodes. The standard simulation set used in this thesis contains 64 static nodes and up to 10 mobile nodes, resulting in a near real time emulation on an Intel Core i5 2800 with 4GB of RAM.

6.1.2 Contiki's uIP IPv6 Network Stack

After having described the Contiki's Cooja simulator, we now describe Contiki's IPv6 network stack, namely uIP. Contiki's uIP IPv6 network stack was chosen as the network stack on which the code was based as it offers a flexible and interoperable deployment environment [15]. uIP is the smallest IPv6 compatible network stack available that is deployable on the TmoteSky platform, allowing us to utilise the large range of features such as transport protocols (TCP/IP or UDP/IP), CSMA (collision sensing / multiple access) or fragmentation and reassembly of data packages.

Contiki's uIP network stack was modified in multiple ways to suit the nature of a WSN with mobility and bursty data transfers incurred by the sink data collection.

The two main modifications, conducted to improve and enable data collection with mobile sinks, will now be described in more detail.

Burst data transfers: The Contiki network stack was modified to accommodate a data burst mode allowing for energy-efficient bulk data transfers. Instead of transferring a single datagram per wake-up cycle, multiple packages are transferred per wake-up cycle, thereby dramatically increasing the data throughput. The maximum data throughput was increased from a single data package (each being 64bits) per wakeup cycle to 10 data packages (640bits), thereby increasing the maximum throughput tenfold. The maximum throughput could theoretically be increased even further to IPv6's maximum transfer unit size (MTU=1500bit), however due to being memory constrained the network stack can only hold a maximum of 640 bits of data in RAM, thereby limiting the maximum throughput. While this was implemented separately with the help of the Contiki OS team, a similar implementation has been conducted by Duquennoy et al. who describe their findings in more detail in [10].

Instant neighbour discovery: The second modification improved the neighbour discovery mechanism of the IPv6 standard. The IPv6 standard decouples the link-layer address from the IP address, requiring neighbouring nodes to initiate an address resolution before any data transfer between neighbouring nodes can be conducted [1]. The address resolution is initiated by broadcasting a neighbour solicitation message, to which a neighbour will respond to with a neighbour advertisement message containing the requested link-layer address.

In delay-tolerant WSNs that experience bursty traffic, e.g. when mobile sinks collect data from fixed sensor nodes, the overhead of address resolution causes many problems. The neighbour solicitation broadcast in particular significantly increases interference as the broadcast monopolises the radio spectrum for the entire broadcast period. Initially, all nodes require address resolution, causing a

waterfall of broadcasts that interfere with each other, often causing more than half the nodes to fail to resolve their parent addresses. This significantly reduces the efficiency and ability of collecting data, requiring a significant change to enable the use of the IPv6 stack.

The resolution to this was to simply remove the address resolution overhead, by aligning link-layer and IP addresses and directly inferring them from each other. More precisely, whenever address resolution takes place for an IP address, we can directly determine the link-layer address as the two correlate within our network. This suppresses the need of the address resolution overhead and enables the use of IPv6 within our network. Naturally, this solution is inappropriate for use in non-homogeneous networks, however, the use of IPv6 within delay tolerant networks with mobile sinks will be very restricted until this issue is addressed.

6.1.3 MAC-Layer Protocols

After having described Contiki's uIP network stack and the modifications made to it, we now describe the MAC-layer protocol used. Contiki OS offers a range of MAC-layer protocols that can be interchangeably used with the uIP network stack. Contikimac, a low-power listening protocol (LPL), was chosen as the MAC-protocol as it is currently amongst the most sophisticated and energy saving MAC-protocols available. Low-power listening (LPL) MAC-layer protocols rely on nodes waking up at given intervals and overhearing whether other nodes request data transfers. The higher the wake-up frequency is set, the higher the throughput and the lower the latency, at the expense of a higher communication cost. The standard Contikimac protocol [10] was modified to incorporate a data burst-mode (as mentioned previously in Section 6.1.2), dramatically increasing the throughput per wake-up cycle, reducing the need for a very high wake-up frequency of the MAC-layer protocol. In our case, Contikimac was set to wake up at a frequency

of 16 times a second in order to accommodate the bursty nature of mobile data collection.

6.2 Experimental Setup

After having described the Contiki operating system as well as its IPv6 networking stack uIP, we will now describe the experimental setup used throughout the simulations. We first describe the configuration of the Cooja simulator, followed by the configuration of the mobile sinks and, thirdly, the setup choices for the static sensor nodes.

6.2.1 Cooja Setup

In this section we describe the configuration of Cooja, Contiki's simulator used to obtain all results. As described previously (see Section 6.1.1), Cooja was configured to use the mspsim library that emulates the hardware behaviour and thereby achieves the most realistic simulation possible.

Cooja's default unit-disc-graph model (UDGM) radio propagation model was used throughout all tests. The unit-disc-graph model simply defines a radio range within which communication is possible as well as an interference range that sets how far a node will interfere with other nodes. The interference range was set to twice the radio range, as it is the default setting in Cooja.

The range of the radio model was set to 75m. The choice of radio range was based on our findings from our real-life deployments in which we found a 75meter range to be realistic for standard antennas commonly used on sensor nodes.

All tests were conducted over a 3-hour or 6-hour simulation time period, depending on whether the sink mobility model changes during the simulations. The 3-hour simulations were run in scenarios where sinks used a fixed mobility model.

The 6-hour simulations were run in scenarios with dynamic changes in sink mobility, allowing for sufficient time to adapt to the change.

After having described the general Cooja configuration, we will now describe the assumptions made about the simulated static sensor nodes and mobile sinks.

6.2.2 Static Node Setup

Our main assumption for static sensor nodes is that they are limited by the amount of energy available, as they are powered by batteries. Consequently, the primary goal is to minimise the energy consumption, in our case incurred by the communication cost that is measured by the effective radio duty cycle. We assume that the sensor nodes are insufficiently powered to be connected to a mobile phone network or simply out of range of such a 3G backhaul. It is therefore the mobile sinks' task to collect and deliver the data.

Due to the computational limitations on the number of sensor nodes being simulated, the network consists of 64 nodes deployed in a 8x8 grid. The distance between the nodes was set to be 80% of the radio range in order to maximise the exposure of the network to sink mobility characteristics, the distance between nodes consequently being 60m for a 75m radio range. The deployment area was therefore 480m$\times$480m. The simulated network area and number of sensor nodes is similar to what would have been required for a real long-term deployment in Wytham Woods. Using 64 nodes is sufficient to highlight the efficiency differences between the routing algorithms, though a larger network would likely lead to a higher usage of hotspot nodes as more data would likely be routed through such hotspots, thereby further exaggerating the differences in energy usage of the routing algorithms.

The data generation rate was fixed to 1 data package every minute (1pkg/min). A fixed data generation rate covers the needs of a large class of sensor network

applications, including environmental and traffic monitoring. The study of bursty data generation patterns is left for future work. The size of the data packages was fixed to 64bits, large enough to contain typical sensor readings, such as temperature values, counts of animals, flow of cars, etc.

6.2.3 Mobile Sink Setup

After having described the static sensor node configuration, we now describe the assumption and configuration choices made for the mobile sinks. The mobile sinks roam through the network to continually collect data that is generated by the static sensor nodes. Unlike the static sensor nodes, we assume that the mobile nodes are not energy restricted as they either have a sufficiently powered battery that can readily be recharged (similar to a mobile phone) or is attached to a powered vehicle (such as a car). Furthermore, we assume that data is considered delivered once data has been collected by a mobile sink, due to the mobile sink being constantly connected to the server via a 3G or equivalent connection. The measured data latency therefore represents the time until data is collected by a mobile sink. This is a realistic assumption as the data could be made available to the end-user as soon as it has been uploaded by the mobile sink, which is assumed to be near instant on a 3G connection.

6.3 Routing Algorithms' Implementation Architecture

After having described the test setup, we will now describe in more detail the implementation architecture of the routing algorithms. While the previously described algorithms seem to be separate, they share many common tasks with regards to routing tree construction and data transfers. When constructing routing trees, P- and R-scheme require the prevention of routing loops and a way of identifying stale

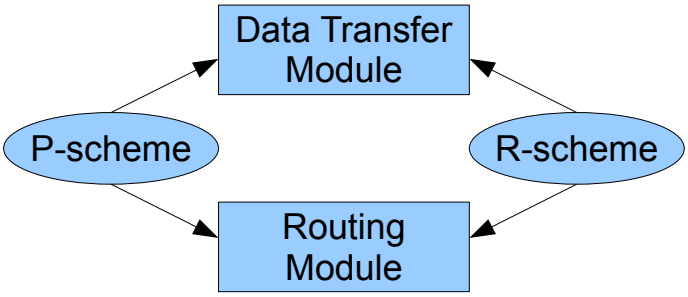


Figure 6.1: P-scheme and R-scheme share a common data transfer module and a common routing module.

routing information. For data transfers, both algorithms require safe custody transfer of data as well as a way of identifying and removing duplicate data packages.

To address these needs, shared routing tree construction and data transfer modules were implemented respectively that P-scheme and R-scheme utilised (see Figure 6.1). The advantage thereof is not only a simpler implementation due to code sharing, but also ensures that the two algorithms were treated on an equal basis when possible. Both the routing tree and the data transfer modules will now be described in more detail.

6.3.1 Routing Module

Routing tree construction is implemented using a simple UDP broadcast (technically a multicast to a group that includes all nodes). When a node receives a beacon and re-transmits a beacon (in order to build the routing tree), a short randomised waiting period is inserted before the beacon is broadcast in order to reduce clashes between nodes also broadcasting a beacon. Conceptually, this should be taken care of by the network stack, namely uIP, however currently the application level and lower-level transmission timings are directly coupled, forcing the application layer to consider the timing of multicast message transmissions. While a random waiting period will not remove the problem of radio interference, it

will reduce its impact.

Expected transmissions: Every node maintains a routing table containing its hop-count to the root node, for both P-scheme and R-scheme respectively. Instead of using the pure hop-count (as described in the pseudo-code of P-scheme and R-scheme), the implementation utilises the number of expected transmissions along the route to the root node. By using the expected number of transmissions, one can avoid and discriminate the use of lossy and inefficient links. If the route quality deteriorates significantly, a node will simply select the next best available route. Initially, this was made necessary by the real-life deployment WSN that resulted in the WildSensing's project publications [11, 12], however we decided to utilise this throughout the thesis as well.

The calculation of the expected number of transmissions was conducted in the routing module and shared by both P- and R-schemes. In order to obtain the expected number of transmissions, each node maintains a neighbourhood table that shows statistics of outgoing traffic. The expected transmissions etx per message from the current node to a neighbour node N is computed as follows: $etx(N) = attemptedtx(N)/successfultx(N)$. Instead of simply increasing the hop-count, as shown in the pseudo-code, the hop-count variable is increased by the expected transmissions to the neighbour N ($etx(N)$). Note that if all the links along a route have an $etx = 1$, the distance is equal to the number of hops along the route. The effective distance to the root node is consequently defined as the sum of expected transmissions on all links along the route.

Loop prevention: Loops within the routes are avoided by also transmitting unique sequence numbers within the beacons, as conducted by DSDV [38]. Each root node in the sensor network maintains a unique sequence number for every beacon it transmits. Whenever a beacon is transmitted, the sequence number is increased by two and associated with the routing information contained within the

beacon. If a node receives a beacon, the beacon freshness is compared to the locally stored freshness and only accepted if the beacon is fresher or the hop-count smaller than the currently stored information, thereby avoiding use of stale information.

6.3.2 Data Transfer Module

After having described the routing module, we now describe the data transfer module. Data is disseminated towards the mobile sinks on a local hop-by-hop basis (instead of end-to-end basis), such that connections are only established to the next hop and not to the final destination. This hop-by-hop transmission approach allows us to deal with data retransmissions on a local level, rather than having the end-points initiate retransmissions. UDP connections are used to transmit data along each hop towards the destination as they incur no additional overhead, unlike TCP/IP connections. In order to reduce the potential radio interference between contested data transmissions, the data transmission time window is set using a random truncated exponential back-off, as commonly used for contested radio mediums [2].

Upon receiving a data package from a child node, the parent node returns a UDP ACK message to confirm reliable transfer. Messages are always stored in onboard flash memory to guarantee eventual delivery of information even in the event of a node power failure. Duplicate data is avoided by comparing data sequence numbers to previously received data and storing only new unseen data to flash memory, this mainly occurring when ACK messages are lost due to interference. Messages are only marked for deletion once an ACK message is received from the parent indicating successful custody transfer. A background garbage collection routine periodically cleans up the flash memory, formatting pages where all messages have been successfully uploaded.

6.4 Summary

In this chapter we described the assumptions and implementation details utilised in the performance evaluation of the routing algorithms. We detailed our implementation assumptions and choices made for the operating system and the uIP network stack. Following this, we described our experimental setup that affect the tests conducted throughout the thesis. We then described the implementation architecture that will aid potential re-implementations.

Chapter 7

Performance Evaluation

After having described the three main data collection schemes, namely P-scheme, R-scheme and TwinRoute, we will now experimentally evaluate their comparative performance. At first, we will compare how Adaptive P- and R-scheme perform against their static counterparts. Following this, we will investigate how Adaptive P- and R-scheme as well as TwinRoute perform for a range of scenarios.

7.1 Adaptive and Static Algorithms

In this section we will compare Adaptive P- and R-scheme against their static versions. At first, we investigate P-scheme, including a detailed description of the required deployment procedures as well as the performance of adaptive and static versions for a range of scenarios. Following this, we investigate static and adaptive versions of R-scheme in a similar manner.

7.1.1 P-scheme

In this section we aim to show the differences that the adaptive and static versions of P-scheme exhibit. The differences will be described in two ways. Firstly, we

describe the deployment procedures that a network engineer would have to undertake to deploy adaptive and static P-scheme versions. Secondly, we evaluate the performance of adaptive and static P-scheme for a range of scenarios.

7.1.1.1 Deployment Procedures

We will now illustrate the tasks required by a network engineer to deploy static and adaptive versions of P-scheme, respectively.

Static P-scheme requires the network engineer to identify frequently visited storage nodes that achieve the application requirements, while minimising the communication cost. The first step in doing so, is to precisely map the locations of the static sensor nodes, also noting potential obstacles for communication. Following this, the connectivity between the sensor nodes needs to be established, either by physically testing or by estimating connectivity based on distance, the former being time consuming while the later potentially being imprecise. Once a map and connectivity has been established, the likely sink mobility has to be predicted or measured and drawn into the map. Now, the network engineer is able to identify storage nodes that are likely to achieve the given data latency and delivery ratio requirements. It is apparent that any of these tasks are prone to failure, forcing the network engineer to likely conservatively select storage nodes in order to ensure timely data delivery. Furthermore, the deployment is labour intensive which consequently raises the cost of the WSN deployment.

The deployment of *Adaptive P-scheme* is very simple. The network engineer deploys the sensor nodes in locations that are likely to yield connectivity between nodes, similarly to Static P-scheme. Following this, the network registers the current sink mobility pattern and self-configures itself accordingly. It becomes very apparent that the adaptive version significantly simplifies the deployment while removing many potential points of failure of Static P-scheme.

7.1.1.2 Performance Evaluation

After having described the deployment procedures, we now compare the performance of Static and Adaptive P-scheme for three deployment scenarios. The test environment was simulated in Contiki's cooja simulator using 64 static sensor nodes distributed in a grid with each sensor being spaced 60 meters apart, each test being run 4 times. The data latency requirement remains at 200s with the delivery ratio threshold set to 80%, a data package being generated once a minute, as described previously (see Section 3.1.2.1). The three scenarios consist of three mobile sinks roaming the network using the following mobility models:

1. A random waypoint model.
2. The Manhattan mobility model that simulates a change in activity between day and night.
3. The Manhattan mobility model with a varying latency requirement.

We will assess how Adaptive P-scheme compares to Static P-scheme, in which the storage nodes are optimally selected by an "oracle." More specifically, the "oracle" selection identifies storage nodes that throughout achieve the desired data requirements. In contrast to this, the adaptive algorithm selects appropriate storage nodes on-the-fly. Adaptive P-scheme's learning period at the start of the deployment can be no longer than the initial $PvisitWindow$, consequently we will only evaluate data delivery from the post-learning-period. For a scenario in which the mobility model remains unchanged, Adaptive P-scheme's target is to achieve a similar performance as Static P-scheme, due to the "oracle" selection of the Static P-scheme algorithm being near optimal.

Random waypoint model: Figure 7.1 shows how Adaptive P-scheme achieves a similar delivery ratio to the "oracle" Static P-scheme, showing how Adaptive P-

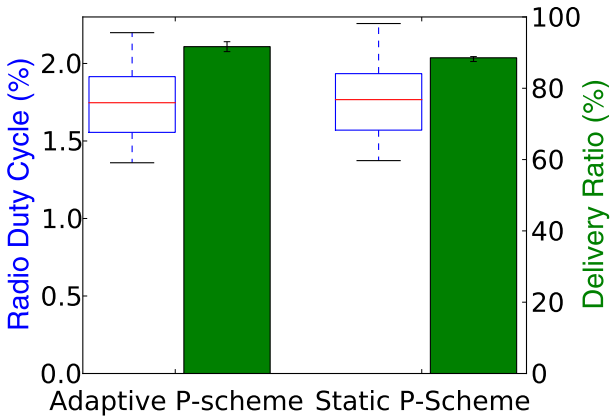


Figure 7.1: Random waypoint model. P-scheme radio duty cycle performance comparison of fixed storage node Static P-scheme against the adaptive storage node selection. The adaptive algorithm achieves a higher delivery ratio than the static selection and a lower maximum and median radio duty cycle.

scheme selects storage nodes that achieve the data requirements. Adaptive P-scheme achieves a slightly lower maximum radio duty cycle than Static P-scheme due to the ”oracle” inherently being more conservative in selecting storage nodes (see Figure 7.1). The mean and distribution of the radio duty cycle appear to be nearly equivalent, indicating that Adaptive P-scheme selects the same or similarly optimal storage nodes as Static P-scheme. This shows that Adaptive P-scheme successfully adapts to the scenario and achieves as good results as the ”oracle” Static P-scheme.

Manhattan model - night / day change: The second scenario represents a change of visiting frequency by simulating the night to day activity of a Manhattan mobility model. To do so, we simply increase the number of sinks half-way through the simulation from 3 to 6. Once more, Static P-scheme will use the selection of storage nodes that are optimally placed for when 3 mobile sinks are roaming the area. Unlike Static P-scheme, Adaptive P-scheme can change its storage node selection to the new mobility model and thereby selects more storage nodes when

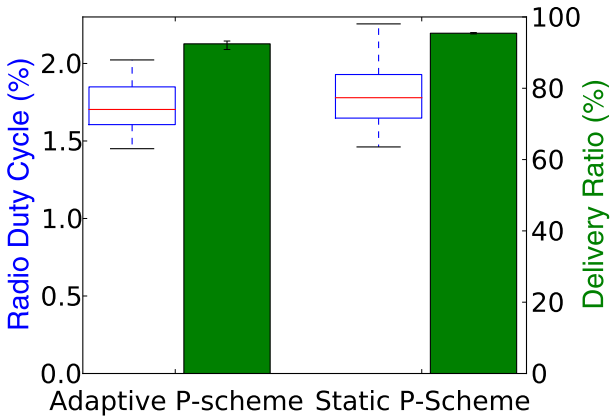


Figure 7.2: Manhattan model - night / day change. Comparison of Static P-Scheme and Adaptive P-scheme. Both algorithms overachieve the data delivery ratio requirements of 80%, while Adaptive P-scheme has a significantly reduced maximum radio duty cycle as it can adapt to the change in sink mobility.

more mobile sinks appear. The simulation duration was doubled to 6 hours to allow each mobility period to equally impact the data delivery.

Figure 7.2 shows how both algorithms achieve the data delivery ratio of 80%. However, Adaptive P-scheme manages to achieve a 15% lower maximum radio duty cycle and 5% lower median radio duty cycle, thereby reducing the first-point-of-failure within the WSN. Adaptive P-scheme is able to do so by adapting its behaviour when the visiting pattern changes due to the increased number of mobile sinks.

Manhattan model - change of data requirements: The third scenario consists of the Manhattan mobility model with a change of data latency requirements halfway through the 6 hour simulation. A change in data latency requirement may occur when a network engineer discovers that the data latency was set too low. In this case, the data latency requirements will be relaxed from 150s to 250s, resulting in more nodes being able to be selected as storage node. Static P-scheme will not be able to adjust to this change, and it will therefore remain with the initial

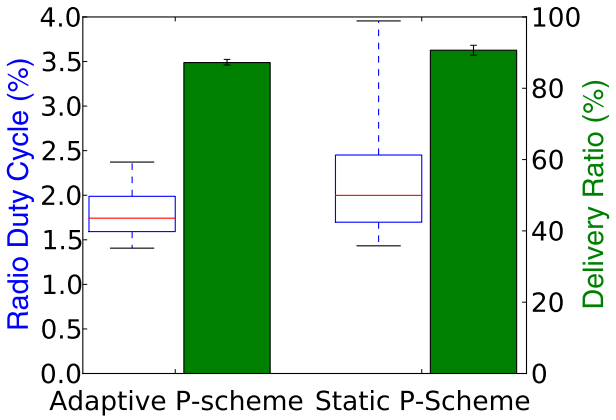


Figure 7.3: Manhattan model - change of data requirements. P-scheme radio duty cycle performance comparison of fixed storage node static P-scheme against the adaptive storage node selection. Both algorithms achieve the data delivery ratio requirement of 80% while Adaptive P-scheme has a significantly reduced maximum radio duty cycle by as much as 40%. The median radio duty cycle is also reduced by as much as 15%, due to the Adaptive R-scheme being able to change according to the new data latency requirements.

selection of storage nodes.

Figure 7.3 shows how both the Static and Adaptive P-scheme achieve the latency requirements. The maximum radio duty cycle of Static P-scheme is 67% higher than that of Adaptive P-scheme. Furthermore, the median radio duty cycle of Adaptive P-scheme is reduced by 15% with a more uniform distribution among the nodes. This scenario shows how the fixed storage node selection is unable to adjust to the change in latency requirements and consequently results in a significantly reduced network lifetime. In contrast to the static version, the adaptive algorithm can easily adjust to the new data requirements and change its configuration accordingly, enabling the WSN to have a significantly longer time-to-first-failure.

7.1.1.3 Summary

The three scenarios illustrate the advantages of the Adaptive P-scheme algorithm. The Adaptive P-scheme algorithm simplifies the deployment, as the network engineer only has to set the data latency and delivery ratio requirements, while adaptively optimising the communication cost according to the current mobility trends.

7.1.2 R-scheme

We now assess how Adaptive R-scheme performs when compared to the Static R-scheme algorithm. At first, we will describe the typical deployment procedures required by a network engineer in order to setup the WSN. Following this, we will compare and evaluate the performance of the Static and Adaptive R-scheme for three scenarios.

7.1.2.1 Deployment Procedures

We will now illustrate the tasks required by a network engineer to deploy static and adaptive versions of R-scheme, respectively. The basic deployment procedures are very similar to those previously described for P-scheme, and are therefore described in less detail.

Static R-scheme requires the network engineer to determine the required tree-depth (TD) that achieves the application requirements, while minimising the communication cost. The first step in doing so, is to precisely map the locations of the static sensor nodes, also noting potential obstacles for communication. Following this, the connectivity between the sensor nodes needs to be established, again either by physically testing connectivity or estimating it based on distance. Once a map and connectivity have been established, the likely sink mobility has to be predicted or measured and drawn into the map. Now, the network engineer is able

to estimate the maximum tree-depth required to collect all data in a timely fashion. It is apparent that any of these tasks are prone to failure, forcing the network engineer to likely overestimate the required tree-depth in order to ensure timely data delivery. Furthermore, the deployment is labour intensive which consequently raises the cost of the WSN deployment.

The deployment of *Adaptive R-scheme* is very simple. The network engineer deploys the sensor nodes in locations that are likely to yield connectivity between nodes. Following this, the network registers the sink mobility patterns and automatically configures itself accordingly.

It becomes very apparent that the adaptive version significantly simplifies the deployment while removing many potential points of failure of Static R-scheme.

7.1.2.2 Performance Evaluation

Three scenarios will be used to show the advantages that an adaptive algorithm can have, namely the same scenarios as used for Adaptive P-scheme performance evaluation (see Section 7.1.1.2). Firstly, a scenario in which the sink mobility model remains constant throughout. Secondly, a scenario in which the sink mobility model changes halfway through the simulation that requires the routing algorithm to adapt to the change. Thirdly, a scenario in which the data latency requirements change midway through the simulation. All other factors remain the same as described in Section 3.1.2.1. The results of each will now be described, respectively.

Random waypoint model: At first, we will assess how Adaptive R-scheme performs when the mobility model and data requirements remain the same. In this scenario three mobile sinks roam the network using a random waypoint model at a speed of 5m/s.

The adaptive configuration is expected to perform nearly as well or better than the static algorithm. Figure 7.4 shows how Adaptive R-scheme and Static R-

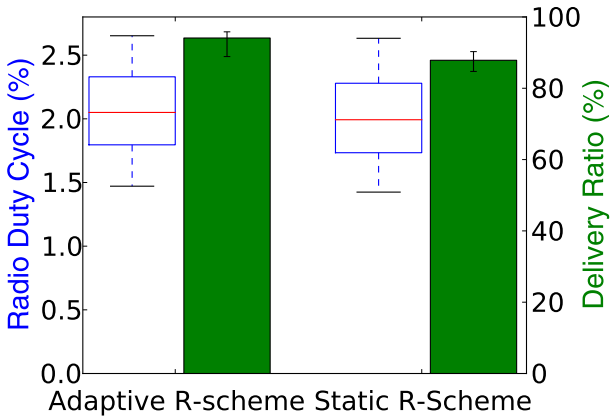


Figure 7.4: Random waypoint model. Comparison of Static R-scheme and Adaptive R-scheme. Both algorithms overachieve the data requirements, Adaptive R-scheme doing more so than the static version. Both algorithms achieve comparable maximum and median radio duty cycles.

scheme perform. Both achieve the required data delivery ratio of 80%, the Adaptive R-scheme achieving a slightly higher ratio than the Static R-scheme. Both algorithms achieve a similar median duty cycle of around 2.0% with a similar duty cycle distribution. The difference between the two is minimal due to the potential advantages of Adaptive R-scheme being negated by the overhead incurred by increasing the tree-depth periodically in order to disseminate the tree-depth requirements (see Section 4.2). The comparable result to an optimally configured Static R-scheme therefore shows how the Adaptive R-scheme algorithm performs well and can adequately adapt to a given environment.

Manhattan model - night / day change: The second scenario uses a Manhattan mobility model that simulates the activity from night to day time, namely a difference in the number of mobile sinks roaming the area. During the night time 3 mobile sinks roam through the area while at day time 6 mobile sinks exist, thereby increasing the visit frequency of the mobile sinks. The change of mobility model activity should trigger a change in Adaptive R-scheme configuration,

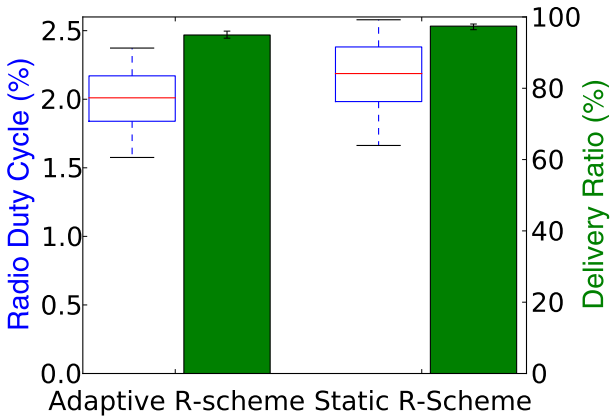


Figure 7.5: Manhattan model - night / day change. Comparison of Static R-scheme with Adaptive R-scheme. Adaptive R-scheme achieves a nearly as good data deliver ratio while outperforming Static R-scheme with regards to maximum and median radio duty cycle.

while the Static R-scheme will keep using its previous configuration. Figure 7.5 shows how Adaptive R-scheme and Static R-scheme perform. Both configurations achieve the minimum delivery ratio of 80%, both Static and Adaptive R-scheme achieving a far higher delivery ratio than is necessary. The maximum radio duty cycle of Static R-scheme performs worse than the adaptive version as it is unable to adapt to the change in effective visiting frequency, leading to a 10% higher maximum radio duty cycle. Adaptive R-scheme achieves a 15% better median radio duty cycle when compared to the static version.

Manhattan model - change of data requirements: The third scenario simulates the change of data latency requirements from 150s to 250s half-way through the 6 hour simulation. Such a change in data requirements might be instigated by a network engineer when discovering that the data latency requirements were set more stringently than actually required. Figure 7.6 shows how both Static and Adaptive R-scheme achieve the delivery ratio requirements of 80%. Adaptive R-scheme achieves a minimally better maximum radio duty cycle as well as a reduced

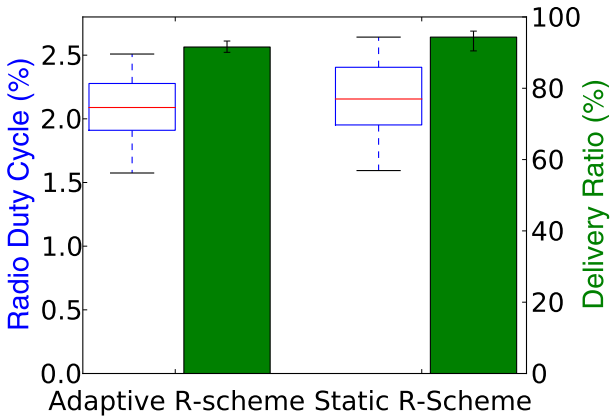


Figure 7.6: Manhattan model - change of data requirements. Comparison between Static R-scheme and Adaptive R-scheme. Both algorithms achieve the data delivery ratio of 80% while Adaptive R-scheme achieves a lower maximum and median radio duty cycle.

median radio duty cycle.

7.1.2.3 Summary

In all three scenarios, Adaptive R-scheme performs nearly as well or better than Static R-scheme, while both are based on the same basic routing algorithm. The Adaptive R-scheme algorithm ensures that the data requirements are obtained, if possible, while minimising the radio usage. This is particularly apparent when the mobility pattern of the mobile sinks changes or the data requirements change mid-way through the simulation.

7.1.3 Summary

In this section we illustrated how the deployment procedures of the adaptive algorithms are significantly simplified over the static versions of P- and R-scheme. We have illustrated how the static versions of P- and R-scheme require a large amount of data to enable a network engineer to deploy a WSN, let alone optimise according

to the given circumstances. In contrast to the static versions, the Adaptive P- and R-scheme algorithms require significantly less information and can largely configure themselves in a self-sufficient manner, allowing for a much more efficient WSN deployment.

We have also shown how the adaptive algorithms can reliably perform as well as or better than optimally configured static algorithms. Unlike the static algorithms, the adaptive algorithms can easily adapt to change within a deployment, such as a change in mobility pattern, visiting frequency or application requirements. In particular when the scenario may change, the adaptive data collection schemes can significantly reduce the radio duty cycle, thereby increasing the lifetime of a WSN while ensuring that the data requirements are met.

7.2 Performance of Adaptive Algorithms in Different Scenarios

After having compared the static and adaptive versions of P- and R-schemes, we will now focus on the adaptive algorithms, namely Adaptive P-scheme, Adaptive R-scheme and TwinRoute, and will evaluate their performance in a range of deployment scenarios. The goal of this section is to establish the strengths and weaknesses of the respective data collection schemes and identify which scenarios they are best suited for. To do so, we will define a range of mobility scenarios that will be analysed over a wide range of application requirements. The mobility models differ fundamentally in two ways. Firstly, the spatial distribution of sink visits throughout the deployment area and, secondly, the average inter-visit times of sinks at the nodes. The investigated scenarios differ distinctly in their combination of these two factors, allowing us to examine how these factors impact the performance of the adaptive algorithms.

The next section will describe the methodology used to analyse and derive the scenario characteristics. Following this, we will evaluate the adaptive algorithms for each individual mobility scenario and experimentally establish and verify their respective strengths and weaknesses.

7.2.1 Methodology

The methodology used to compare the proposed adaptive algorithms consists of three parts. Firstly, we analyse the scenario under investigation with regards to the spatial sink visit distribution and inter-visit times. Secondly, we shed light on how the adaptive algorithms behave for different data requirements. Finally, we empirically evaluate the performance of the adaptive algorithms. Each will now be described in more detail.

7.2.1.1 Sink Mobility Analysis

The sink mobility analysis allows us identify scenario characteristics by analysing the sink visiting pattern with regards to two primary attributes. Firstly, we examine whether the spatial visit distribution of the mobile sinks is concentrated at a few nodes or is uniformly distributed throughout the whole WSN.

Secondly, we examine whether the scenario has a low or high average inter-visit time with respect to the data latency requirements, i.e. whether nodes are visited often enough to dispatch data to mobile sinks before they expire. A low average inter-visit time increases the likelihood of allowing timely data collection. Hence, the distribution of inter-visit times provides useful information about the achievable data latency requirements.

As a final minor characteristic, we assess how the visit durations are correlated to the average inter-visit times in order to identify cases in which the visit durations

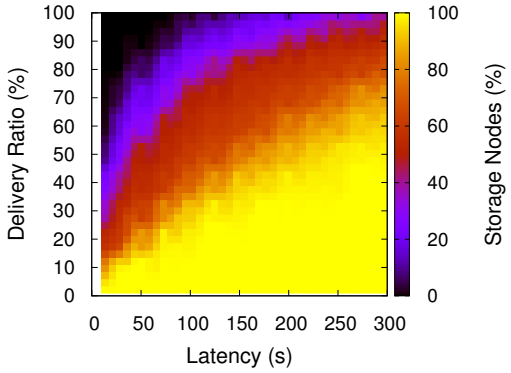


Figure 7.7: Example graph showing the behaviour of Adaptive P-scheme in a scenario with three sinks following the random waypoint model. It allows us to assess how many storage nodes would potentially be selected for a combination of data latency and data delivery ratio requirements. Scenarios with zero selected storage nodes represent data requirements that are unachievable by any P-scheme configuration, while boundary cases allow us to gauge the limits of achievable data requirements. The gradient of storage nodes indicates how the radio duty cycle is likely to be affected by a change in data requirements.

may be insufficiently short to ensure all data will be collected on time.

7.2.1.2 Algorithm Behaviour

After having discussed the sink mobility statistics, we delve into the algorithm behaviour. The algorithm behaviour is examined with regards to the estimated selected storage nodes and tree-depth required by the Adaptive P- and R-schemes, respectively, to achieve a given set of data requirements.

Recall from Section 4 that the adaptive algorithms require a sink visiting log, namely the visit time and duration of respective sink visits. This is derived by running a simple simulation with the sensor nodes at the pre-defined locations registering the visit time and duration as well as tree-depth, as would happen in a real deployment. Following this, we simply utilise the Adaptive P-scheme algorithm for an exhaustive range of data delivery ratio and data latency requirements. Consequently, we can establish the number of storage nodes over the total number of nodes in the network, thereby gauging whether any storage nodes will be selected

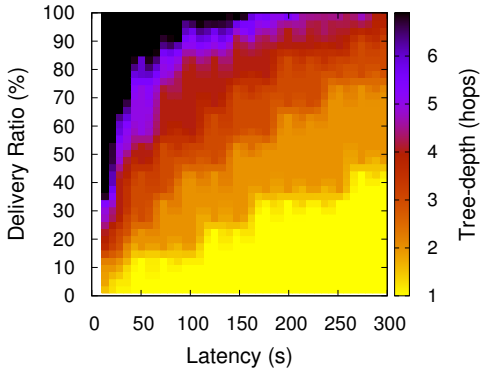


Figure 7.8: Example graph showing the behaviour of Adaptive R-scheme in a scenario with three sinks following the random waypoint model. It allows us to predict the tree-depth configuration for a combination of data latency and data delivery ratio requirements. The black regions represent data requirements that cannot be achieved by Adaptive R-scheme, while boundary cases allow the network engineer to gauge the limits of achievable data requirements. The gradient of change in tree-depth indicates what effect a change in data requirement may have on the potential radio duty cycle.

and how the scheme may potentially perform (see Figure 7.7 for an example graph illustrating the algorithm behaviour). The data delivery ratio ranges from 1% up to 100%, while the data latency requirements range from 10s to 300s.

If no storage nodes are selected, we can assume that the Adaptive P-scheme will be unable to achieve the scenario's data requirements. The more storage nodes are selected, the less imminent the hotspot problem will be, resulting in a potentially lower radio duty cycle. The gradient of selected storage nodes also indicates how, as we relax the data requirements, the radio duty cycle is likely to be affected, which directly affects the network lifetime.

The Adaptive R-scheme algorithm behaviour is also based on the simulation derived sink visit log, namely visit time and duration as well as tree-depth. Every node's required tree-depth is established by running the Adaptive R-scheme algorithm with the sink visit log for a specific data latency and data delivery ratio requirement. As we are primarily interested in the first-point-of-failure within the WSN, it is important to establish the maximum required tree-depth for each pair of

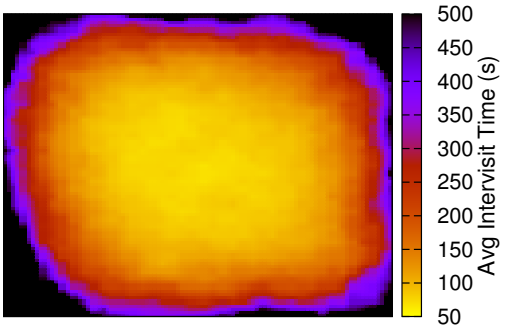
data requirements, as it is likely to indicate the maximum radio duty cycle within the network. We exhaustively analyse the same data delivery ratio ranges from 1% up to 100% and a data latency requirements range from 10s to 300s.

An example is given in Figure 7.8, which allows us to gauge whether Adaptive R-scheme can achieve given data requirements. The black regions represent data requirements that are unlikely to be achieved by Adaptive R-scheme, while boundary cases thereof allow the network engineer to gauge the limits of achievable data requirements. The gradient of change in tree-depth indicates what effect a change in data requirements may have on the potential radio duty cycle.

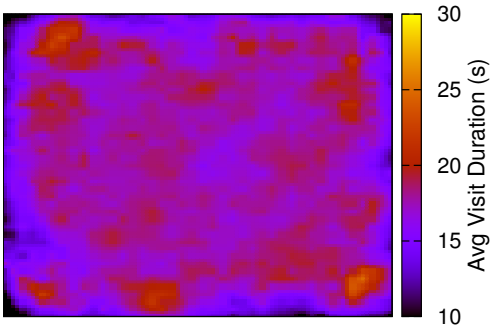
7.2.1.3 Experimental Results

Once we have shown how the pure adaptive algorithms are likely to behave in a particular scenario, we proceed to empirically evaluate the performance of the adaptive algorithms, pure or hybrid. For simplicity, we will fix the data delivery ratio to 80% and evaluate individual scenarios with data latency requirements ranging from 50-250s. We report on three aspects of algorithms' performance. Firstly, we will identify the range of data latencies that can be achieved by each of the algorithms. Following this, we discuss the maximum radio duty cycle which indicates the likely time-to-first-failure. Finally, we evaluate the average radio duty cycle which gives an indication of the network wide communication cost of the data collection schemes.

To summarise, our methodology has three steps: we present the scenario under investigation, show how pure adaptive algorithms behave in this scenario, and compare their performance in it. In what follows we apply this methodology to six different scenarios.



(a) Average intervisit time heatmap



(b) Average visit duration heatmap

Figure 7.9: Spatial distribution of inter-visit times and visit duration for the farm maintenance model in which 2 sinks roam the area using the random waypoint model (RWP). a) The spatial inter-visit distribution shows how the farm located at the centre is more frequently traversed than the outer regions. b) A relatively uniform visit duration is maintained throughout the area due to the constant speed of the sinks.

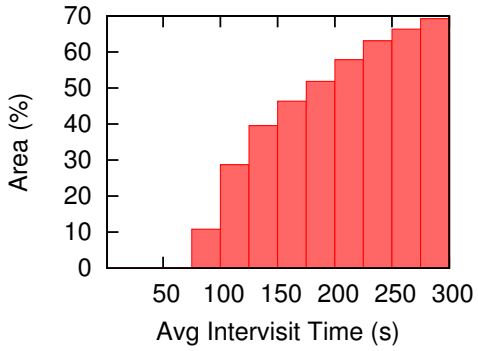
7.2.2 Farm Maintenance Model

The farm maintenance scenario simulates the behaviour of farm workers traversing the area using off-road vehicles, such as tractors or quad-bikes. The workers follow a random waypoint model (RWP) that simulates the semi-random behaviour of farmers working at random parts of the farm area while having an increased likelihood of passing by the centre of the farm. Two such farm workers constantly roam the area at a speed of 5m/s.

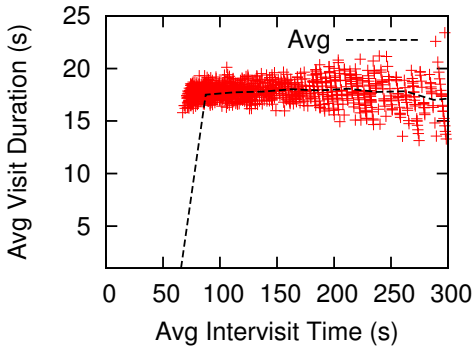
7.2.2.1 Sink Mobility Analysis

The spatial distribution of inter-visit times (see Figure 7.9a) shows how the farm workers move following the random waypoint model. The central regions are more likely to be visited as they are more likely to be traversed by the farm workers. In order to achieve the data requirements, the outer regions will have to rely on the more frequently visited central nodes for data collection.

The spatial distribution of the visit duration (see Figure 7.9b) has a relatively



(a) Spatial inter-visit time distribution



(b) Intervisit time vs. visit duration

Figure 7.10: Mobility trace statistics for the farm maintenance scenario. a) The average inter-visit distribution shows that only 10% of nodes are visited on average every 100s. This increases continually towards 70% for an inter-visit time of 300s. b) The visit duration is similar for most nodes and does not depend on the inter-visit time. A visit duration of 15-20s gives sufficient time for all sink visits to collect the available data.

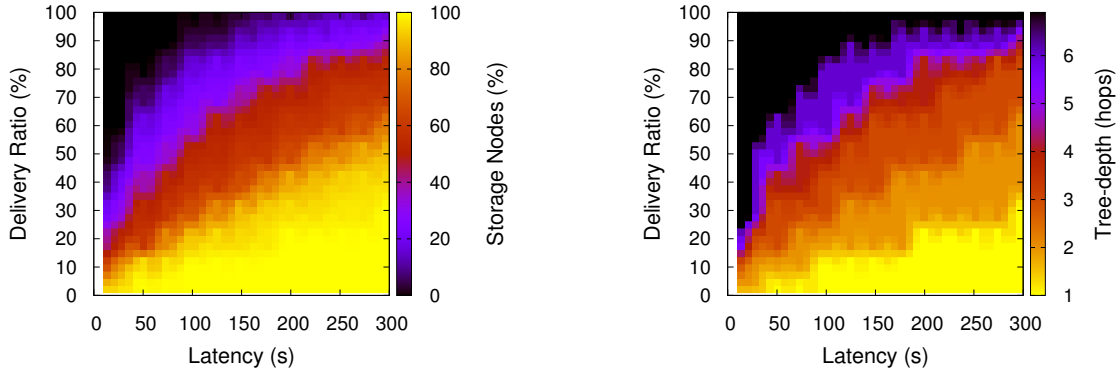
uniform distribution throughout the whole deployment area.

Figure 7.10 shows that only 10% of the deployment area is visited on average every 100s, this area being the central region. The distribution increases continually with 70% of the area being visited on average every 300s. The nodes that are visited with low inter-visit times are strongly concentrated within the central region, this being the distinguishing feature of the random waypoint model that the mobile sinks are following.

Figure 7.10b shows how the sink visit duration is not correlated with the inter-visit time, and that the majority of sink visits range from 15-20s, which is sufficient time to allow for timely data collection.

7.2.2.2 Algorithm Behaviour

We will now discuss how the adaptive algorithms are likely to behave in the farm maintenance scenario. In particular, we look at the number of selected storage nodes (of Adaptive P-scheme) as well as maximum tree-depth (of Adaptive R-



(a) Adaptive P-scheme behaviour

(b) Adaptive R-scheme behaviour

Figure 7.11: Algorithm behaviour graphs for the farm maintenance scenario. a) The Adaptive P-scheme behaviour graph shows how the number of potential storage nodes gradually increases when the data requirements are more relaxed, namely a higher data latency or a lower delivery ratio. b) The Adaptive R-scheme behaviour graph shows how the hop-count requirements also gradually decrease with more relaxed data requirements.

scheme) that are needed to achieve the data requirements. Recall that the data requirements determine acceptable data latency and data delivery ratios, the combination of the two resulting in different configurations for the data collection schemes (see Figure 7.11).

The algorithm behaviour for Adaptive P-scheme (see Figure 7.11a) shows how as we relax the data requirements, more storage nodes are selected which increases the likelihood of the scheme achieving the data requirements and/or reducing the hotspot problem within the WSN. More precisely, the higher the data latency is set the more storage nodes will be selected. Conversely, the higher the delivery ratio is set the fewer storage nodes are selected as the resulting data requirements are more difficult to achieve.

Similarly, Figure 7.11b shows how the maximum tree-depth of Adaptive R-scheme also continually increases as the required delivery ratio increases or the required latency decreases. For a delivery ratio of 80% a latency of 75s is not achievable; latency values of interest are above 100s.

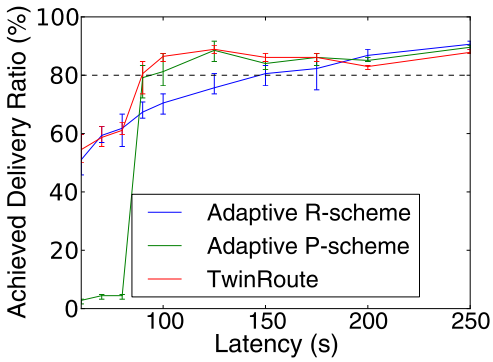
7.2.2.3 Experimental Results

After having analysed the farm maintenance scenario, we will now present and analyse the experimental results. The experimental setup remains the same as mentioned in Section 6.2 with 64 nodes being employed in a grid with 60m distance between the nodes. The mobility model used for the farm maintenance scenario remain the two random waypoint sinks as described previously. For simplicity, we will fix the delivery ratio to 80% and vary the latency from 50-250s, allowing us to analyse how the performance of the data collection schemes changes as we vary the latency.

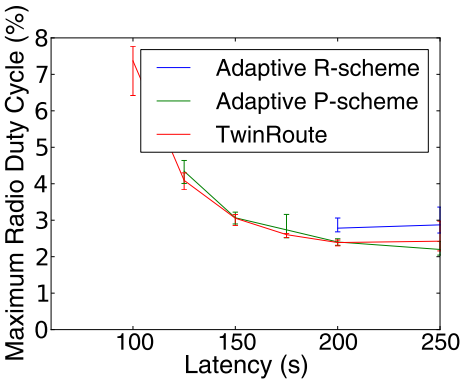
The evaluation of the experimental results is tied to three factors (see Section 1.1.2). Firstly, we are interested in whether the given data requirements are achieved. Secondly, we compare the maximum radio duty cycle of the data collection schemes as this indicates the lifetime of the WSN until the first point-of-failure. Thirdly, the average radio duty cycle shows how much energy the network collectively spends (total number of batteries replaced). We will now assess the farm maintenance model for each of these factors.

Figure 7.12a shows how Adaptive R-scheme only achieves latency requirements higher than 200s. In contrast to this, Adaptive P-scheme and TwinRoute achieve substantially lower data latency requirements, with a minimum of 100s and 125s respectively. This is due to both Adaptive P-scheme and TwinRoute relying on the proactive components to collect data at the frequently visited central region.

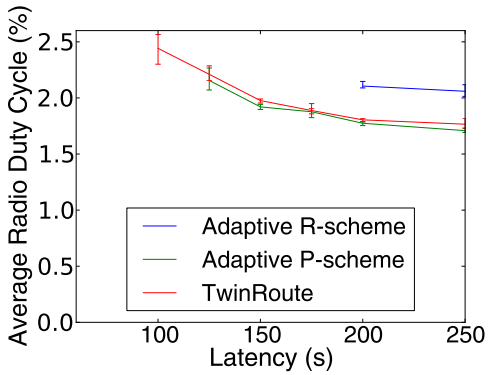
Adaptive R-scheme requires a comparatively higher maximum radio duty cycle than Adaptive P-scheme and TwinRoute due to it incurring a higher routing overhead (see Figure 7.12b). TwinRoute and the Adaptive P-scheme have a nearly indistinguishable maximum radio duty cycle. TwinRoute achieves a lower data latency requirement of 100s due to it off-loading some of the routing load using the



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.12: Experimental results for the farm maintenance scenario. a) Adaptive R-scheme achieves only data latency requirements from 200s onwards, while Adaptive P-scheme and TwinRoute achieve substantially lower latency requirements (125s and 100s respectively). b) Adaptive P-scheme as well as TwinRoute achieve a lower maximum radio duty cycle than Adaptive R-scheme for all the scenarios for which it can achieve the data latency requirements. c) Adaptive P-scheme and TwinRoute average radio duty cycle is lower than Adaptive R-scheme for all latency scenarios. This is likely to be due to the higher routing overhead of Adaptive R-scheme.

reactive component. However, the increase of the maximum radio duty cycle is dramatic due to the increased hotspot problem caused by very few storage nodes having to bear the majority of the routing load.

The average radio duty cycle for the data collection schemes (see Figure 7.12c) is lower for Adaptive P-scheme and TwinRoute for all latency scenarios than Adaptive R-scheme can achieve, which is likely to have been caused by the higher

routing overhead incurred by the reactive tree construction in R-scheme. Once more, the difference between Adaptive P-scheme and TwinRoute is too small and falls within the expected margin of error.

7.2.2.4 Summary

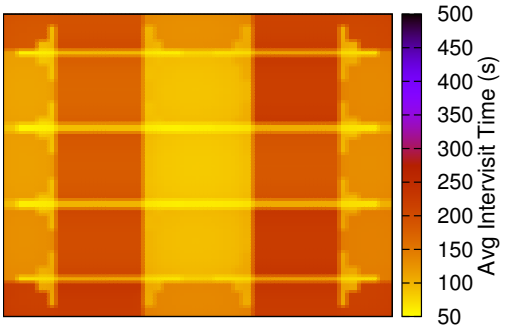
For the farm maintenance model Adaptive P-scheme and TwinRoute show a significant strength over Adaptive R-scheme by achieving significantly lower data latency requirements. For scenarios that Adaptive R-scheme can fulfil, the average and maximum radio duty cycle of R-scheme is higher due to it incurring a higher routing overhead. Generally, this scenario suits the schemes that select storage nodes , namely P-scheme and TwinRoute, as they can achieve a wider range of latency scenarios as well as a lower radio duty cycle. TwinRoute can achieve a lower data latency requirement than any of the pure adaptive approaches as it can draw on both reactive and proactive components, thereby being more flexible.

7.2.3 Town Suburb

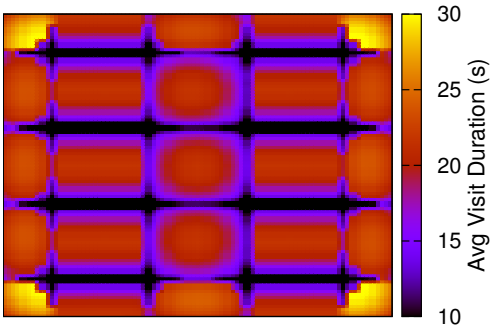
The town suburb scenario simulates mobile sink behaviour in which humans follow a Manhattan mobility model at a slow car pace or bicycle speed of 5m/s. We simulate the movement of an example suburb by having three humans constantly roam the roads of a 5x3 town suburb grid. When reaching a cross-road, mobile sinks have an equal likelihood of moving straight ahead or turning left or right.

7.2.3.1 Sink Mobility Analysis

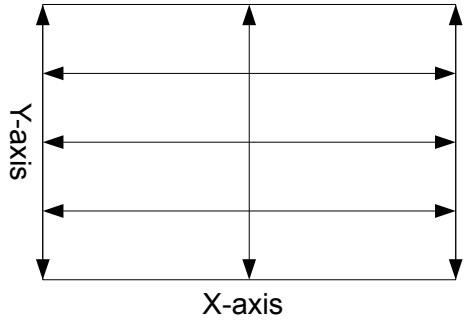
The spatial distribution of average inter-visit times and visit durations can be seen in Figures 7.13a and b, respectively. The Manhattan grid is not directly obvious and has therefore been highlighted separately in Figure 7.13c. The average inter-visit



(a) Average intervisit time heatmap



(b) Average visit duration heatmap

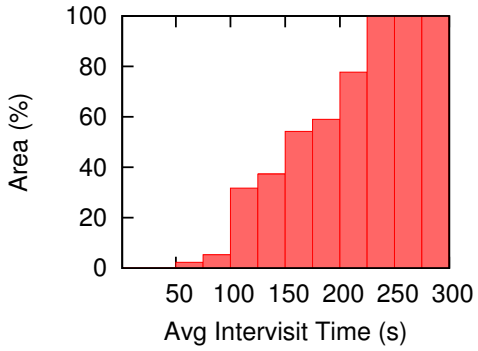


(c) Road Layout

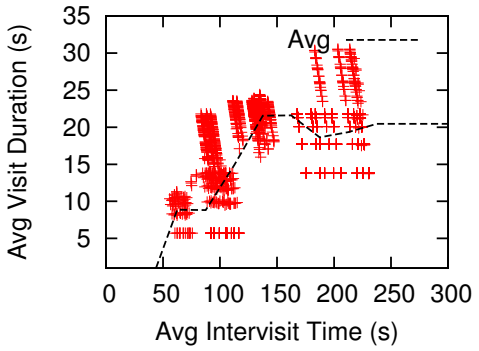
Figure 7.13: Spatial distribution of inter-visit times and visit duration for 3 sinks roaming a town suburb according to a Manhattan mobility model with a road grid of 3x5. a) The spatial inter-visit distribution is relatively uniform throughout the deployment area with the majority of the area having an average inter-visit time between 100-250s. b) The visit duration depends on the distance to the roads. The closer a node is to the road, the longer the visit duration is going to be due to nodes being longer in range of the mobile sink. Minor visit duration peaks exist at the corner nodes due to longer exposure caused by the pathway having to traverse the corner, thereby allowing the mobile sinks to be in range for longer than typically found on average. c) The 3x5 road grid of the Manhattan mobility model.

times (see Figure 7.13a) are relatively evenly spread throughout the entire area, with minor spikes existing where the radio range overlaps between two roads.

The average visit duration (see Figure 7.13b) is visibly higher close to the roads and lower further from the roads, this being due to nodes close to the road being in range for longer than nodes further away from the roads. Minor spikes of visit durations can be seen in the corners of the deployment areas due to the sinks having to traverse the road's corners while constantly being in range of the corner



(a) Spatial intervisit time distribution



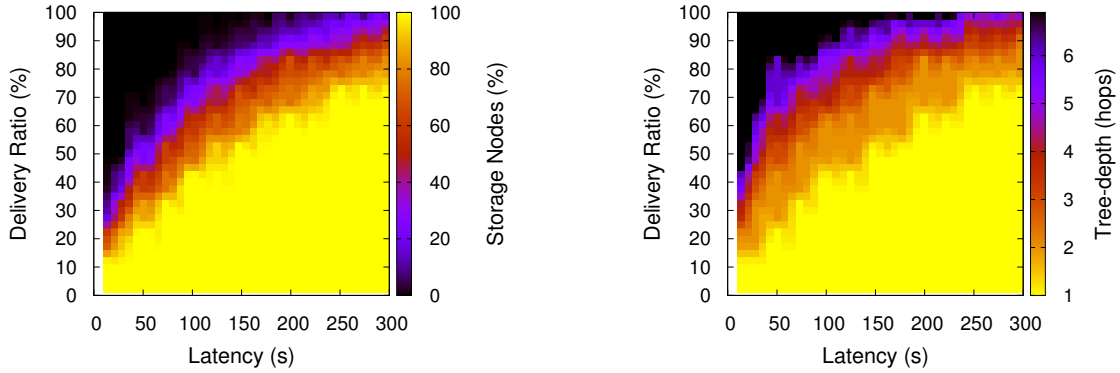
(b) Intervisit time to visit duration correlation

Figure 7.14: Trace statistics for the town suburb scenario. a) The minimal noticeable average inter-visit time is 30% of area being being visited within 100s. The coverage then progressively increases until 100% of the area is visited on average within 225s, resulting in a very uniform distribution. b) The average visit duration is strongly positively correlated to the average inter-visit time, with nodes with a low average inter-visit time often being visited for as short as 5-10s only. The majority of nodes is visited for 10s or longer, a sufficient amount of time to collect the data.

nodes.

Figure 7.14a shows that the majority of nodes are visited on average between 100-225s, 100% of the area being visited on average at least every 225s. This shows that the town suburb scenario has a relatively uniform inter-visit distribution, mainly incurred by the Manhattan mobility model road network which results in sinks visiting all areas on a regular basis.

The average visit duration is highly positively correlated to the inter-visit time (see Figure 7.14b), as nodes that are frequently visited are often only visited for very short periods. While the majority of nodes are visited for 10s or longer, the few nodes that have a short visit duration of 5-10s might struggle with uploading all their data to mobile sinks before the sink becomes out of range. This is likely to affect Adaptive P-scheme more when these nodes are the only selected storage nodes, as they are unlikely to be able to offload all their data.



(a) Adaptive P-scheme behaviour

(b) Adaptive R-scheme behaviour

Figure 7.15: Algorithm behaviour graphs for the town suburb scenario. a) The Adaptive P-scheme behaviour graph shows how the number of storage nodes increases sharply over a potentially small range when changing the data requirements. b) R-scheme also has a very short span during which the hop-count configuration changes from 1 hop to potentially not being achievable. For a delivery ratio of 80% the range of achievable latency scenarios is predicted to be above 100s.

7.2.3.2 Algorithm Behaviour

After having analysed general trace characteristics, we now assess the algorithm behaviour for the Adaptive P- and R-scheme in the given mobility scenario. The Adaptive P-scheme (see Figure 7.15a) shows a sharp increase of qualifying storage nodes as we select a more relaxed data latency or delivery ratio requirement. For a delivery ratio of 80%, we can expect to achieve latencies of 125s and higher, with nearly all nodes being selected as storage nodes from a latency of 250s and upwards. Consequently we can also expect a significantly reduced radio duty cycle for more relaxed latencies as fewer hot spots exist and the routing load becomes more balanced.

The Adaptive R-scheme behaviour graph (see Figure 7.15b) shows a sharp increase in required tree-depth configuration as we select more stringent data requirements. The algorithm behaviour for a delivery ratio of 80% allows us to expect Adaptive R-scheme to perform adequately from a latency of 100s upwards, with a

significantly reduced tree-depth configuration of $TD = 1$ from 250 seconds onwards. Unlike Adaptive P-scheme, we can only assume minor reductions in the radio duty cycle as R-scheme inherently is already well distributed, avoiding P-scheme's hotspot problems.

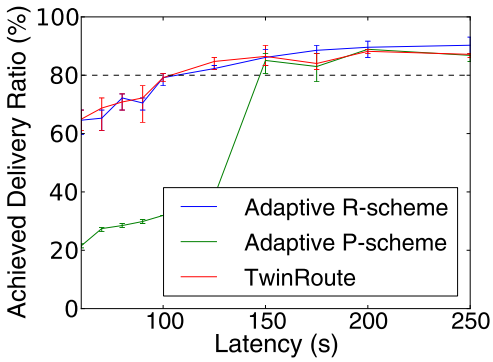
7.2.3.3 Experimental Results

After having analysed the town suburb scenario, we will now present the results from the experimental evaluation.

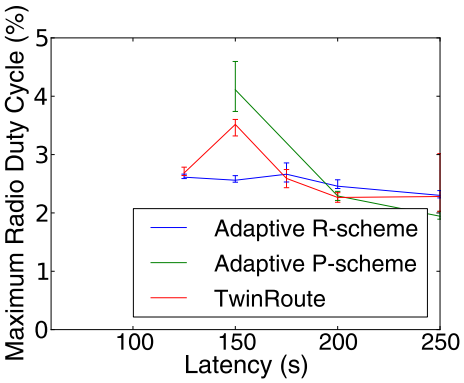
The data latency requirements achieved by TwinRoute and Adaptive R-scheme range from 125s upwards (see Figure 7.16a), while Adaptive P-scheme can only achieve a latency requirement of 150s upwards. This is due to Adaptive P-scheme relying on the selection of storage nodes, which for latencies up to 125s is unobtainable. For a latency of 125s and lower, TwinRoute can rely on its reactive component and thereby still achieves the data latency requirements.

The maximum radio duty cycle is slightly lower for Adaptive P-scheme than for Adaptive R-scheme as long as it can select a sufficient amount of storage nodes and consequently distributes the routing load evenly, for example, when the data latency requirements is very high (e.g. 250s). This changes dramatically when Adaptive P-scheme selects very few storage nodes and consequently suffers from the routing hotspot problem, resulting in a large spike in the maximum radio duty cycle for the latency scenario of 150s. Meanwhile, Adaptive R-scheme continues to distribute its routing load and achieves a more stringent latency requirement of 125s. TwinRoute follows the pattern of Adaptive P-scheme and to a lesser extent also suffers from the proactive hotspot routing problem. For very low latencies (125 sec), it selects no storage nodes and falls back to the reactive component and behaves like Adaptive R-scheme.

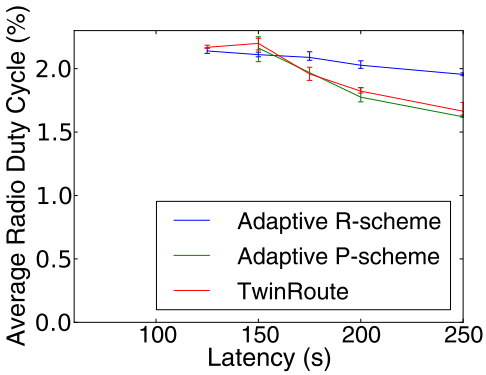
The average radio duty cycle draws a similar yet less exaggerated picture (see



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.16: Experimental results for the town suburb scenario with a delivery ratio requirement of 80% and a range of data latency scenarios. a) The data latency requirements are achieved by Adaptive P-scheme from 150s upwards, while Adaptive R-scheme and TwinRoute achieve these from 125s upwards and act more gracefully thereafter. This is due to Adaptive P-scheme relying on the presence of storage nodes, this not being the case for a latency of 125s and below. b) The maximum radio duty cycle is slightly lower for Adaptive P-scheme when it can select sufficient storage nodes, namely when the data latency requirement is relaxed. As soon as very few storage nodes are available, the routing hotspot problem becomes imminent and the maximum radio duty cycle peaks dramatically, as can be seen for the scenario with a latency requirement of 150s. Both TwinRoute and Adaptive R-scheme can rely on their reactive behaviour and can achieve lower latency requirements as they are not reliant on storage nodes. c) The average radio duty cycle is very similar for Adaptive P-scheme and TwinRoute.

Figure 7.16c) in which Adaptive P-scheme achieves a lower radio duty cycle and TwinRoute follows it very closely. This is due to TwinRoute relying on its proactive component for the bulk of data collection, however it also has at times a higher

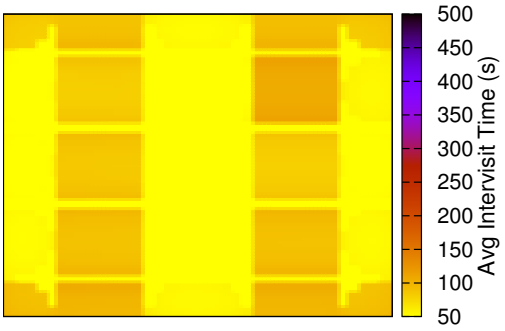
routing overhead due to the reactive data collection trees being built in areas that are distant from storage nodes.

7.2.3.4 Summary

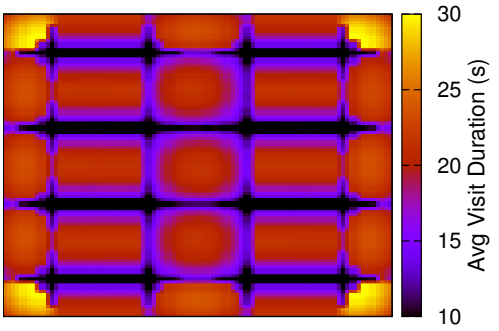
In this section we analysed and experimentally verified a town suburb scenario in which 3 sinks follow a Manhattan mobility grid. For relaxed data latency requirements in which Adaptive P-scheme and TwinRoute can select many storage nodes, they outperform Adaptive R-scheme which incurs a higher routing overhead. However, as soon as Adaptive P-scheme selects very few or no storage nodes at all, Adaptive R-scheme as well as TwinRoute outperform Adaptive P-scheme. For all latency scenarios where Adaptive P-scheme achieved the lowest radio duty cycle, the difference between Adaptive R- and P-scheme is relatively small. Consequently, it seems as if a reactive algorithm, such as Adaptive R-scheme or TwinRoute, is better suited for this scenario as it can more reliably achieve the data latency requirements without potentially suffering from the hotspot problem.

7.2.4 Town Centre

The town centre scenario is simulated by 6 sinks following the Manhattan mobility model at a speed of 5m/s, which can be found when people cycle or drive cars slowly through the town centre. The town centre scenario and the previously described town suburb scenario technically use the same mobility models, however with more sinks being present within the town centre scenario. The increase in mobile sinks leads to a higher sink visiting frequency, for which we will investigate how this affects the different data collection schemes.



(a) Average intervisit time heatmap



(b) Average visit duration heatmap

Figure 7.17: Spatial distribution of inter-visit times and visit duration for the town centre scenario. a) The spatial distribution of inter-visit times is relatively uniformly distributed with most nodes being visited on average every 50-100s. b) The visit duration is highly influenced by the location of the roads; nodes close to the roads have longer durations as they are longer in direct range of mobile sinks.

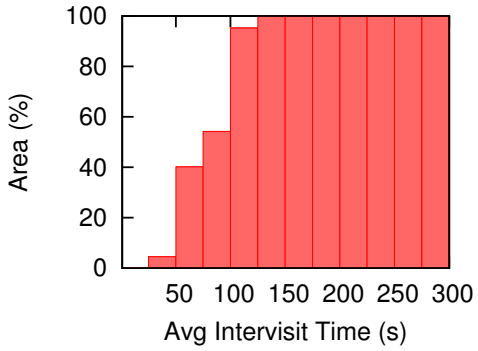
7.2.4.1 Scenario Analysis

The sinks, which follow the Manhattan mobility model, move on the same 3x5 road grid, as the one used for the town suburb scenario, which is depicted in Figure 7.13c.

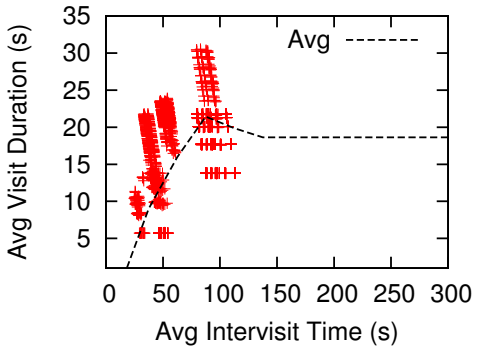
Figure 7.17a shows that the average inter-visit times range between 50-100s for the entire deployment area. The distribution of the average visit duration (see Figure 7.17b) is similar to the town suburb scenario, in that nodes closer to the roads have a longer visit duration due to the increased time they are in range to a mobile sink. The nodes in-between roads have low visit durations, which may not be sufficient to allow for direct mobile sink data collection.

Figure 7.18a shows that all nodes are visited on average within 125s. The Manhattan mobility model enforces regular visits to all regions while the increase of mobile sinks leads to a lower inter-visit time as the likelihood of sink visits increases with the number of sinks within the deployment region.

The average visit duration is, again, highly positively correlated to the inter-visit times (see Figure 7.18b). The majority of nodes are visited for 10s or longer,



(a) Spatial intervisit time distribution



(b) Intervisit time to visit duration correlation

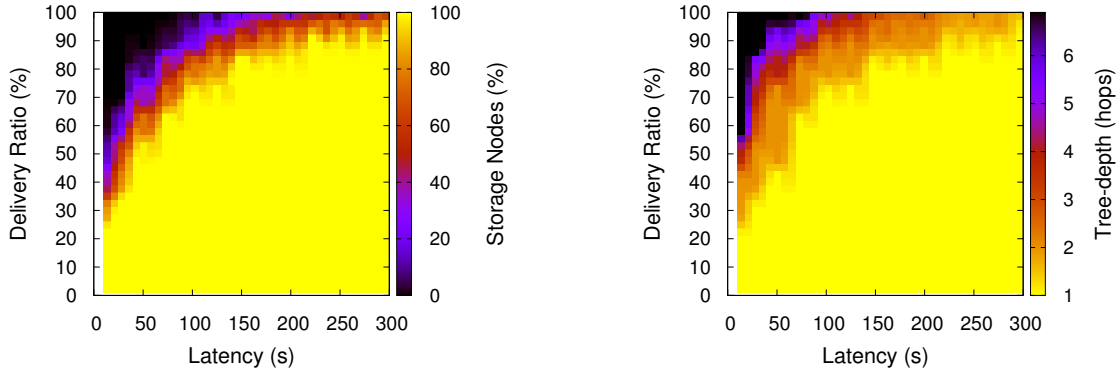
Figure 7.18: Sink visit statistics for the town centre scenario. a) The majority of nodes are visited on average every 50s to 100s, showing how uniformly the 6 sinks following the Manhattan mobility model visit the nodes within the deployment area. b) There is a strong correlation between inter visit times and visit duration, which has shifted to lower latencies. The majority of nodes have a visit duration between 10-30s.

however, nodes with a shorter period will not be visited long enough to allow for data collection. It is to be expected that Adaptive P-scheme will, in such a case, once more suffer from the hotspot problem and struggle to achieve the required delivery ratios.

7.2.4.2 Algorithm Behaviour

Figure 7.19a shows the behaviour of Adaptive P-scheme in the town centre scenario as we vary the data requirements for latency and delivery ratio. Notice that in this scenario, Adaptive P-scheme can achieve more stringent requirements than in the town suburb scenario. For a given delivery ratio, say 80%, the number of storage nodes increases very fast from one to all nodes as the latency requirement varies from 50 to 150 secs. Beyond this point, the algorithm behaviour remains constant, and degenerates to every node storing data locally and waiting for a mobile sink to directly offload its data.

The Adaptive R-scheme algorithm behaviour (see Figure 7.19b) has a similarly



(a) Adaptive P-scheme behaviour

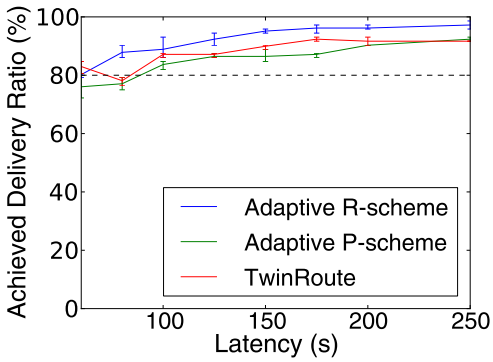
(b) Adaptive R-scheme behaviour

Figure 7.19: Algorithm behaviour for the town centre scenario. a) The Adaptive P-scheme behaviour graphs shows how the number of storage nodes increases sharply as we relax the data requirements. The sharp increase becomes especially apparent as the latency range of interest for a delivery ratio of 80% ranges only from 50-150s, a very short range. b) The Adaptive R-scheme behaviour graph shows a sharp decrease in required tree-depth, with a large amount of data requirements being satisfied with a 1-hop configuration.

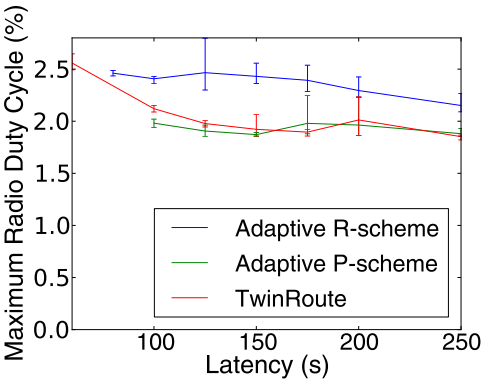
short band in which the configuration changes from the maximum 6-hop to the minimum 1-hop trees. For a delivery ratio of 80%, this happens in the latency range from 25-150s. The area in which 1-hop trees are built is furthermore significantly increased due to the higher visiting frequency achieved by the mobile sinks. Recall that building 1-hop trees means that nodes store data locally and wait for mobile sink visits to directly offload the data. Thus, Adaptive P-scheme with 100% storage nodes has the same behaviour as Adaptive R-scheme with 1-hop trees.

7.2.4.3 Experimental Results

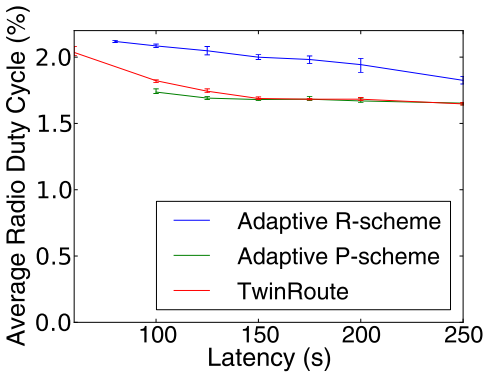
The data requirements are satisfied for all data collection schemes for latencies higher than or equal to 100s (see Figure 7.20a), with Adaptive R-scheme achieving a lower data delivery ratio of 80s and TwinRoute even achieving a data latency requirement of 60s. While Adaptive P-scheme does still select storage nodes for latencies of 80s and below, it does not always reliably collect data within the data latency threshold. It may be noted that when the data collection schemes fail to



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.20: Experimental results for the town centre scenario with a delivery ratio requirement of 80% for a range of data latency scenarios. a) All schemes achieve a latency requirement of 100s and upwards, with Adaptive R-scheme achieving a latency of 80s and TwinRoute achieving an even lower latency requirement of 60s. It may be mentioned that both Adaptive P- and R-scheme barely fail to achieve the lower latency requirements. b) The maximum radio duty cycle required by Adaptive P-scheme and TwinRoute is markedly lower than Adaptive R-scheme, due to the latter's reactive tree construction overhead. Adaptive P-scheme and TwinRoute have very similar performance. c) In terms of average radio duty cycle, the difference between Adaptive P-scheme and TwinRoute is negligible while Adaptive R-scheme requires a higher average radio duty cycle due to its route construction overhead.

achieve the data requirements, they do so marginally.

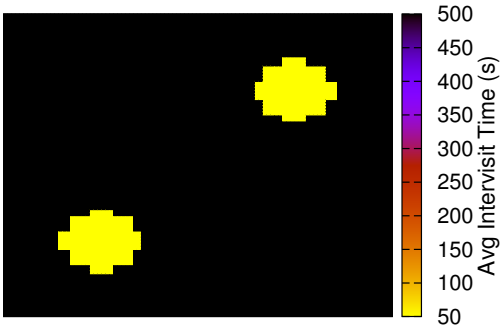
The maximum radio duty cycle of Adaptive P-scheme and TwinRoute is markedly reduced by 20-30% when compared to Adaptive R-scheme (see Figure 7.20b), this being attributed to the overhead incurred by the reactive tree construction.

Adaptive P-scheme and TwinRoute have negligible differences that fall within the expected simulation error. Their performance is similar because they both heavily rely on their proactive component. The result of this is that Adaptive P-scheme and TwinRoute are expected to have an extended time-to-failure than Adaptive R-scheme.

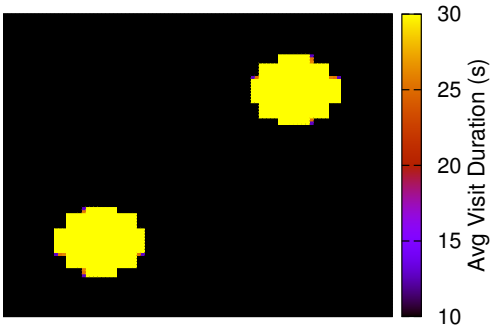
The average radio duty cycle draws a similar picture (see Figure 7.20c) in which Adaptive P-scheme and TwinRoute achieve a 10-20% lower radio duty cycle than Adaptive R-scheme, this also being attributable to the reactive route construction overhead.

7.2.4.4 Summary

In this section we investigated a town centre scenario in which 6 mobile sinks roamed the deployment area using a Manhattan mobility model. The higher number of mobile sinks leads to a significantly lower inter-visit time, thereby increasing the number of storage nodes or decreasing the required tree-depth for Adaptive P-scheme and R-scheme, respectively. Adaptive P-scheme reliably achieves a lower radio duty cycle for both maximum and average radio duty cycles, while Adaptive R-scheme can achieve a lower data latency requirement as it does not suffer from P-scheme's hotspot problem. However, TwinRoute can combine P-scheme's radio duty cycle efficiency with an even lower data latency requirement than R-scheme, as it can draw on the strengths of reactive as well as proactive components. In conclusion, Adaptive P-scheme outperforms Adaptive R-scheme for the majority of latency scenarios while TwinRoute in turn outperforms Adaptive P-scheme.



(a) Average inter-visit time heatmap



(b) Average visit duration heatmap

Figure 7.21: Spatial distribution of inter-visit times and visit durations for the ocean monitoring scenario with two fixed gateway nodes. The inter-visit times and visit durations are concentrated around the static gateway nodes. Due to the constant presence of the gateway nodes the inter-visit times are zero and the visit duration covers the entire deployment period.

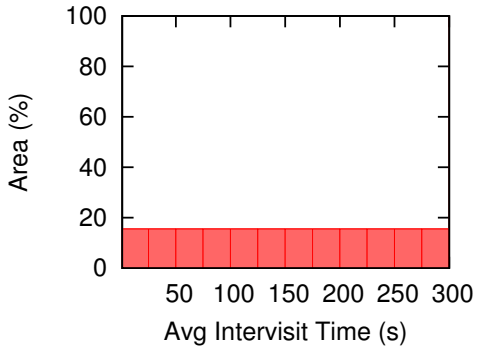
7.2.5 Ocean Monitoring

The ocean monitoring scenario consists of nodes that are attached to fixed position buoys in order to sense environmental data from the ocean. The data upload is conducted via two static gateway nodes that are connected to the outside world via long distance connections such as a satellite uplink. This scenario is typically used for fully static WSN deployments in which single gateway nodes are used to interconnect the WSN, without making use of mobile sinks.

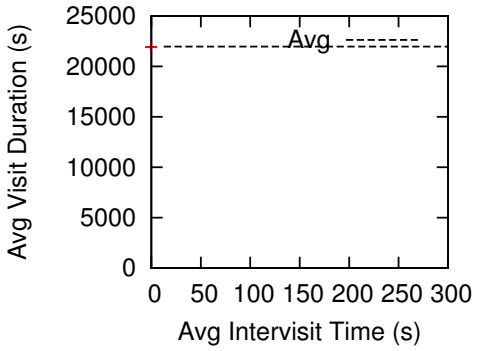
7.2.5.1 Scenario Analysis

Figure 7.21 shows that the nodes within range of the two gateway nodes have an inter-visit time of zero, and visit duration set to a maximum value. The only points from which data can be collected are the two gateway nodes, as is typical for a static WSN deployment that does not make use of mobility within the network. None of the remaining areas are visited by sinks.

Figure 7.22a shows that the two gateway nodes are always within reach of



(a) Spatial inter-visit time distribution



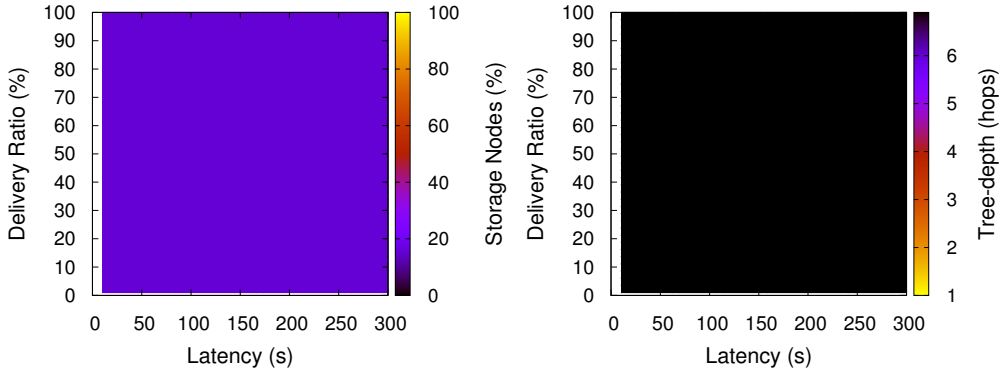
(b) Intervisit time to visit duration correlation

Figure 7.22: Sink visit statistics for the ocean monitoring scenario. a) The static gateway nodes are constantly in touch with 17% of the nodes and consequently have an inter-visit time of zero. As the scenario exhibits no mobility, the covered area remains at a constant 17%. b) The visit duration covers the entire span of the simulation, namely 22000s, which in the light of this immobile scenario is irrelevant.

17% of the nodes, for which the inter-visit time is zero. The visit duration is constant and only limited by the duration of the simulation, namely 22000 seconds (see Figure 7.22b), a sufficiently large amount of time to allow all data to be duly transferred.

7.2.5.2 Algorithm Behaviour

The algorithm behaviour of the respective schemes is relatively simple due to the static nature of the gateway nodes. Adaptive P-scheme simply selects the nodes within the vicinity of the gateway node as its storage nodes, for any set of data requirements. The nodes within the vicinity of the storage nodes constitute roughly 17% of the nodes. Adaptive R-scheme, however, is unable to collect all the data from the network as the maximum tree-depth (6 hops) is insufficient to reach all the nodes. This is to be expected as the reactive approach relies entirely on mobility which is notably absent in this scenario.

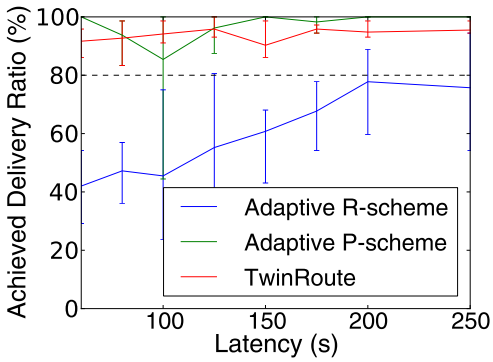


(a) Adaptive P-scheme behaviour (b) Adaptive R-scheme behaviour

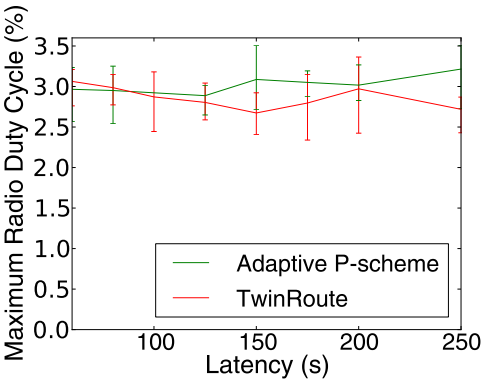
Figure 7.23: The algorithm behaviour for Adaptive P-scheme shows that a constant amount of storage nodes is likely to be selected, namely the sensor nodes in direct proximity to the gateway sinks. The same number of storage nodes will always be selected as this is independent of application requirements. The non-existence of an Adaptive R-scheme tree-depth indicates that the R-scheme is unable to achieve any data requirements as the hop-counts are too large to reliably reach the entire network.

7.2.5.3 Experimental Results

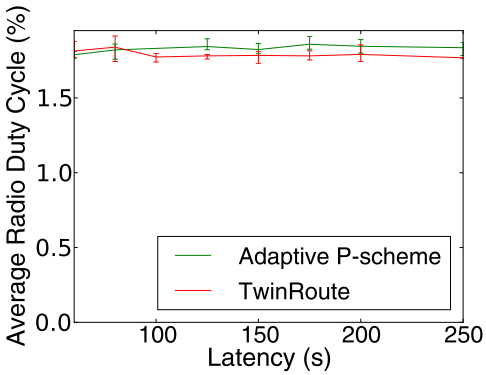
The experimental analysis shows how both Adaptive P-scheme and TwinRoute achieve the data delivery ratio requirements of 80% (see Figure 7.24a), while Adaptive R-scheme is unable to achieve them due to it being unable to reliably connect to all the nodes within the WSN. Technically, this could be addressed by either increasing the MAC-layer throughput, which would result in a higher radio duty cycle for all nodes, or by simply increasing the tree-destruction time which effectively results in Adaptive R-scheme acting like Adaptive P-scheme. This, however is not our aim. Reactive schemes are inherently limited in their tree-depth due to their reliance on fast response times and high throughput rates in order to rapidly build reactive trees and collect all the necessary data. Response times and data throughput rates are primarily limited by the MAC-layer, the maximum range of a reactive scheme thereby also being limited. Our aim is to show how Adaptive P-scheme can natively achieve the data requirements without a specialised con-



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.24: Experimental results for the ocean monitoring scenario with a fixed delivery ratio of 80% for a range of data latency requirements. a) Adaptive P-scheme and TwinRoute reliably achieve the data requirements for all latency scenarios due to their ability to fall back on the proactive storage nodes. In contrast to this, Adaptive R-scheme achieves significantly lower delivery ratios as the reactive nature of the data collection scheme is unable to reliably collect data from all nodes. b) Adaptive P-scheme and TwinRoute have similar maximum radio duty cycle characteristics, with TwinRoute having marginally better results. c) The Adaptive P-scheme and TwinRoute also have similar average radio duty cycles.

figuration, while Adaptive R-scheme can, if at all, only achieve it with significant tuning.

TwinRoute and Adaptive P-scheme show a similar behaviour with regard to their maximum and average radio duty cycle (see Figures 7.24b and c). The minor differences can be attributed to the margin of error to be expected for complex simulations, particularly in extreme cases with high throughput rates such as the

ocean monitoring scenario.

7.2.5.4 Summary

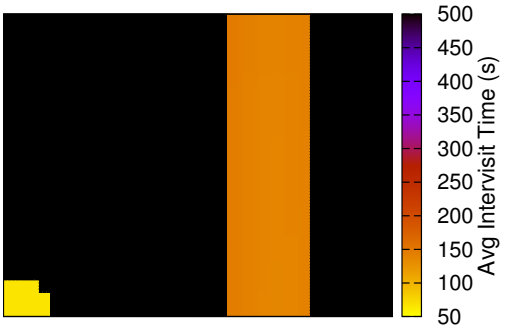
In this section we analysed and experimentally tested the ocean monitoring scenario in which two static gateway sinks were connected via a satellite uplink to the outer world. Due to the lack of mobility, Adaptive R-scheme was unable to achieve the data requirements though it could potentially have done so if specially configured for this scenario. In contrast to this, Adaptive P-scheme and TwinRoute were able to natively achieve the data latency requirements by simply acting like gateway nodes would in a standard static WSN deployment. It shows that for scenarios with little mobility, proactive schemes easily outperform reactive schemes, as reactive schemes typically assume sink mobility.

7.2.6 National Reserve

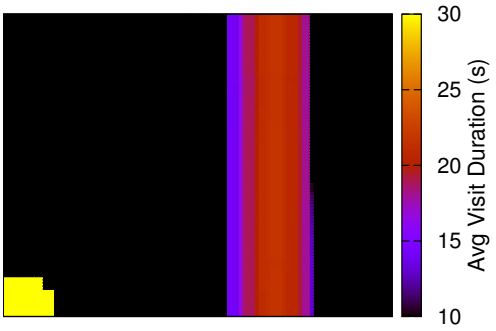
The national reserve scenario is made up from a single gateway node that is connected to the outside world via a 3G or similar uplink, while two mobile sinks roam a single pathway that traverses the national reserve at a speed of 5m/s, similar to cyclists or slow off-road vehicles. Consequently, two separate paths of data collection are available, namely the infrequently traversed pathway as well as the gateway node.

7.2.6.1 Scenario Analysis

Both the road as well as the gateway node are clearly visible with distinct visiting characteristics (see Figure 7.25). The average inter-visit times (see Figure 7.25a) for nodes close to the roaming path are constant throughout, which is to be expected as the path is regularly traversed by the roaming sinks. Figure 7.25b shows



(a) Average intervisit time heatmap



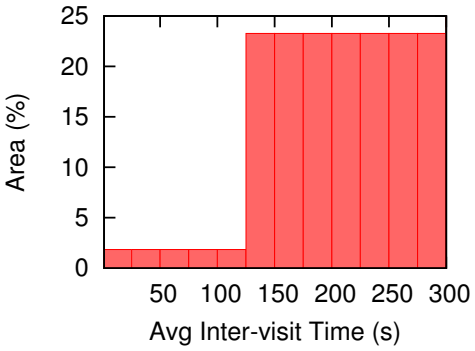
(b) Average visit duration heatmap

Figure 7.25: Spatial distribution of inter-visit times and visit duration for the national reserve scenario. a) The inter-visit times of the gateway node as well as the path are distinct, with the path having an inter-visit time of 125s. b) The visit duration is dependent on the distance of the node from the path, closer nodes having a longer visit duration as they are more exposed to the range of the mobile sink. The visit duration at the static gateway node is high due to its constant presence.

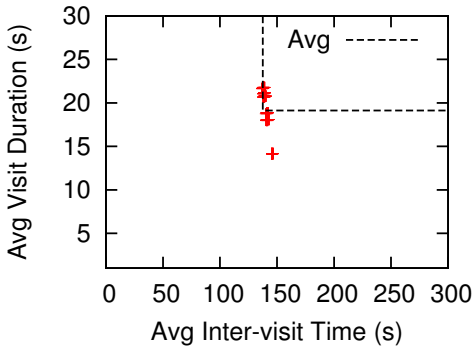
how the average visit duration is highly dependent on the node's distance to the pathway, closer nodes having a longer visit duration.

The highly concentrated visiting patterns, exclusive to nodes in range to the pathway and the gateway node, will require the data collection schemes to route their data towards the visited areas, something that Adaptive R-scheme might not reliably achieve. Furthermore, only the gateway node can guarantee timely delivery and will likely have to be the primary point of data collection as the pathway is being visited only infrequently.

Figure 7.26a shows that nodes close to the path are visited on average every 125s. Consequently, any scenarios with a data latency below 125s will have to primarily rely on data being collected by the static gateway node. Data latency requirements of 150s and higher can potentially also be handled by the pathway storage nodes, allowing for a more balanced and potentially more reliable data collection. The visit durations (see Figure 7.26b) are reliably within the 15-25s range, a sufficient amount to allow for timely data collection.



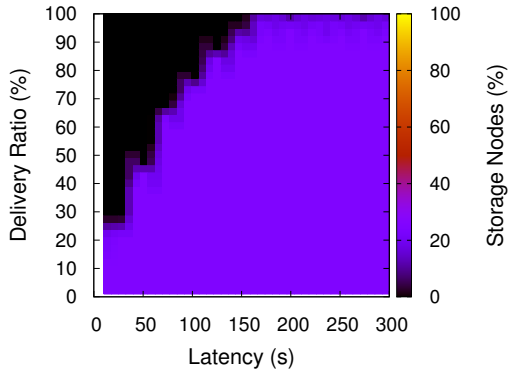
(a) Spatial intervisit time distribution



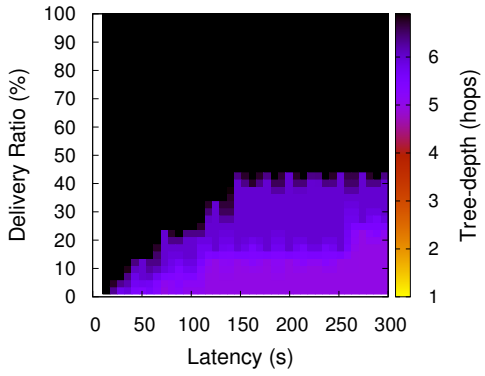
(b) Intervisit time to visit duration correlation

Figure 7.26: Sink visit statistics for the national reserve scenario. a) Nodes close to the path are visited on average every 150s, 26% of the area being covered by the path. With no further mobility present in the network, the covered area remains the same for all higher latencies b) The average inter-visit to visit duration correlation graph shows how the visit durations are repeatedly the same ranging from 15-25s. In fact, the visit durations are nearly indistinguishable due to them all falling on the same spots.

7.2.6.2 Algorithm Behaviour



(a) Adaptive P-scheme behaviourn



(b) Adaptive R-scheme behaviour

Figure 7.27: Algorithm behaviour graphs for the national reserve scenario. a) For Adaptive P-scheme, the pathway nodes' infrequent yet predictable visiting characteristics result in a sharp storage node selection characteristic. For all scenarios, few storage nodes are selected that are within range of the gateway node, while scenarios with data requirements that are met by the pathway sinks see a sharp increase in available storage nodes. b) Adaptive R-scheme suffers greatly from unreliably connecting to nodes that are far from the pathway due to it being restricted by the MAC-layer performance. Consequently, it is unlikely to achieve a vast majority of the data requirement scenarios.

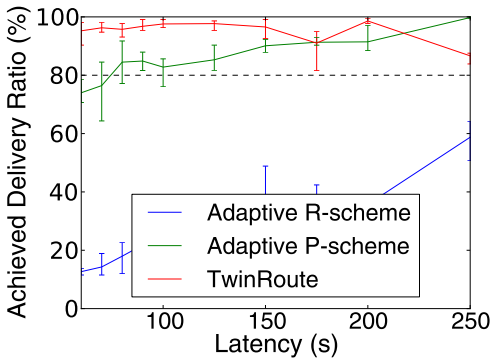
The Adaptive P-scheme behaviour graph (see Figure 7.27a) shows a sharp increase of predicted storage nodes once the data requirements are met by nodes close to the pathway. For all stringent data requirements that are not met by the pathway, only nodes within range of the gateway node will be selected as storage nodes. Consequently, we can expect Adaptive P-scheme to be vastly more efficient for scenarios in which the pathway nodes satisfy the data requirements, resulting in data being collected from the pathway as well as gateway sinks.

Adaptive R-scheme suffers greatly from the unreliable long range connection, largely due to it being restricted by the MAC-layer performance. In order to reliably reach distant nodes, the MAC-layer has to be extremely responsive and allow for high throughput. Our standard MAC-layer configuration is already configured to meet those requirements and has little head-room for improvement, other than using a MAC-layer that is constantly on, something that defeats the whole purpose of maximising network lifetime. Adaptive R-scheme is able to achieve very relaxed data requirements, however is likely to fail to achieve the vast amount of data requirement scenarios.

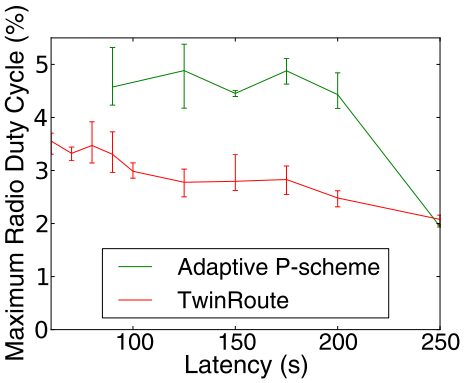
7.2.6.3 Experimental Results

Adaptive R-scheme fails to meet any of the data latency requirements (see Figure 7.28a) due to its inability to reliably collect data solely using the pathway sinks. In contrast to this, Adaptive P-scheme achieves nearly all latency requirements with minor exceptions for extremely stringent latency requirements below 80s. The failure can be attributed to Adaptive P-scheme's inherent hotspot problem of the nodes bearing the vast majority of the routing load, namely the nodes close to the gateway sink. TwinRoute achieves all data requirements, even the extremely stringent requirements that Adaptive P-scheme fails at.

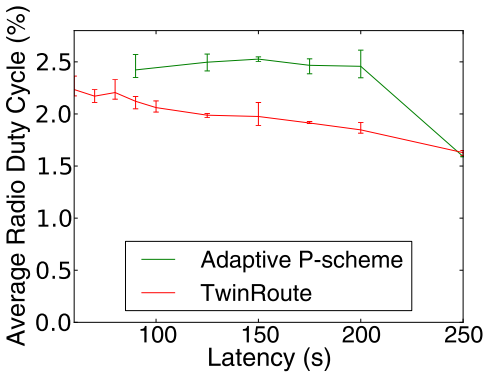
TwinRoute achieves a significantly reduced maximum radio duty cycle when



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.28: Experimental results for the national reserve scenario with delivery ratio requirements of 80% for a range of data latency requirements. a) Adaptive P-scheme achieves the majority of the latency requirements, however struggles with the more stringent latencies that are smaller than 80s. Adaptive R-scheme fails to achieve any of the latency requirements, as it is unable to reliably collect data using the infrequent pathway sinks only. TwinRoute achieves all data latency requirements, as it can rely on the gateway node for the majority of scenarios while collecting substantial amounts of data using the pathway sinks. b) TwinRoute handily outperforms Adaptive P-scheme with regards to the maximum radio duty cycle, as it significantly reduces Adaptive P-scheme hotspot problem surrounding the gateway node by having the pathway nodes collecting data using the reactive component. c) A similar picture is drawn for the average radio duty cycle in which TwinRoute beats Adaptive P-scheme for all latency scenarios.

compared to Adaptive P-scheme (see Figure 7.28b). This is due to TwinRoute utilising the reactive component to collect data that is distant to the gateway node, thereby significantly reducing Adaptive P-scheme's hotspot problem. For some scenarios (data latency 125-200s), TwinRoute reduces the maximum radio duty

cycle by as much as 50%, potentially doubling the time-to-failure. Similarly, TwinRoute requires a reduced average radio duty cycle (see Figure 7.28c) due to it reducing the average number of hops that a message needs to reach a sink.

7.2.6.4 Summary

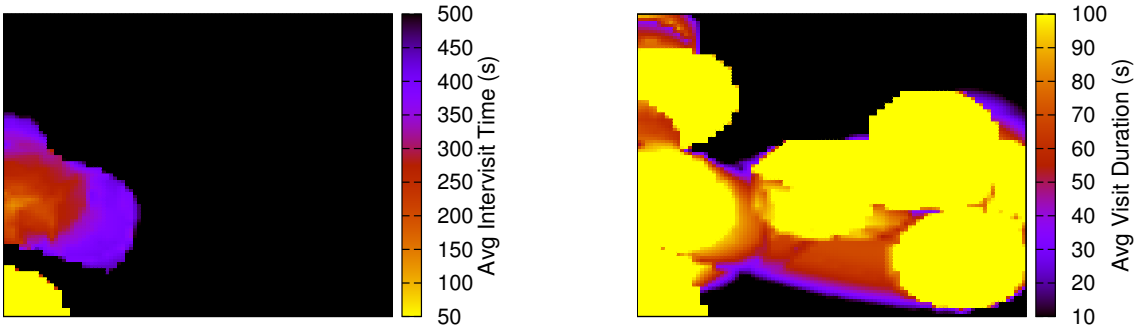
In this section we analysed and experimentally tested the national reserve scenario in which a static gateway node and two pathway sinks roam the area.

We found that Adaptive R-scheme is unable to reliably collect the data due to it being restricted by the MAC-layer responsiveness. Adaptive P-scheme can achieve many of the data requirements while having difficulties to achieve very low latencies or very high delivery ratios. This is due to Adaptive P-scheme suffering substantially from the hotspot problem and relying solely on the gateway node to collect the data.

TwinRoute achieves all the data requirements as it can rely on the static gateway node to collect the majority of data, while the roaming pathway nodes substantially reduce the hotspot problem by collecting data locally. This results in prolonging the expected time-to-failure by as much as 100%, a substantial gain achieved through combining reactive and proactive components.

7.2.7 Wytham Woods - National Park

In this section we scrutinise a national park scenario that consists of real mobility traces that were gathered during a WSN deployment in Wytham Woods, Oxfordshire (UK). The traces were gathered by attaching GPS devices to quad-bikes that were driven by foresters roaming the woods over a period of 6 hours. Their work conducted during the period mainly consisted of observing the woods and logging trees.



(a) Average intervisit time heatmap

(b) Average visit duration heatmap

Figure 7.29: Spatial distribution of inter-visit times and visit duration for the Wytham Woods scenario. a) The areas most frequently visited are restricted to cross-roads in the west side of the woods, as well as nodes close to the static gateway node. The vast majority of the deployment area is visited very infrequently, forcing the data collection to rely on the frequented areas only. b) Many areas are visited for extensive amounts of time, this being likely due to the foresters conducting extensive amounts of work.

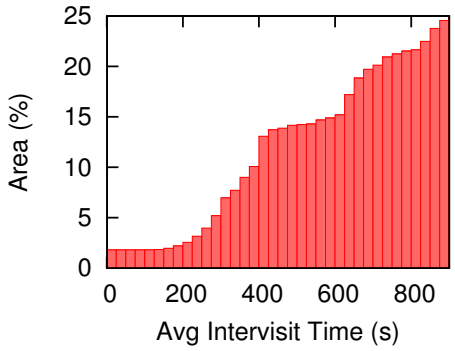
In addition to the mobile sinks carried by the foresters, a static gateway node was placed in the forest that provided a constant connection for uploading data.

7.2.7.1 Scenario Analysis

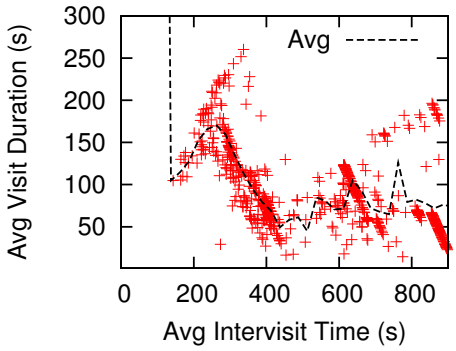
The Wytham Woods scenario has very few areas that have a low average inter-visit frequency (see Figure 7.29a). One such area consists of a cross-road at which the roaming foresters are likely to pass by when driving around the woods. Furthermore, the static gateway allows the surrounding nodes to be constantly visited. The data collection will have to rely on the restricted amount of area that is frequently in range of a mobile sink.

In contrast to the few frequently visited areas, long visit durations are to be found throughout the deployment area (see Figure 7.29b). This is due to the foresters parking their vehicles (to which the mobile sinks are attached) and conducting extensive amounts of work over a longer duration.

The distribution of low inter-visit times is highly concentrated with only 2% of the



(a) Spatial inter-visit time distribution



(b) Inter-visit time to visit duration correlation

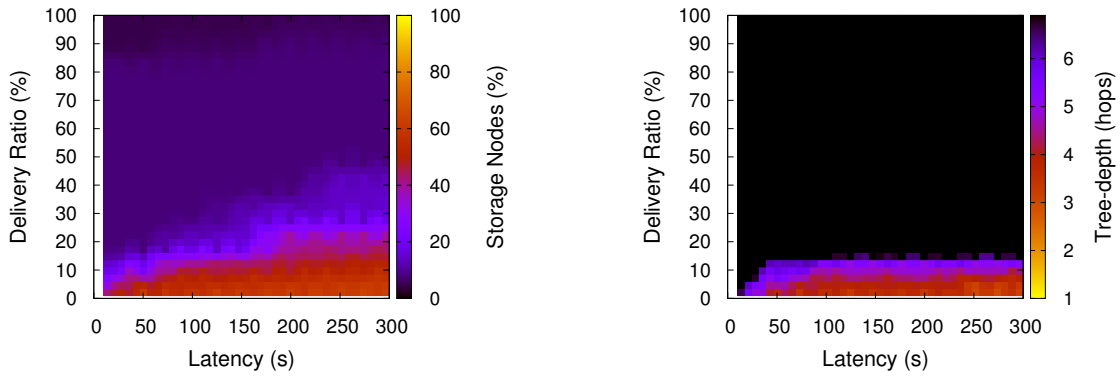
Figure 7.30: Trace statistics for the Wytham Woods scenario a) The static gateway node ensures that 2% of the area can constantly deliver their data, however, the sink visiting patterns are too unreliable to yield a significant amount of additional storage nodes even when the data requirements are relaxed. b) The correlation between inter-visit time and visit duration is highly random. The visit durations vary widely from 10s to as much as 300s, allowing for sufficient time to collect data.

nodes being constantly in touch with the static gateway node (see Figure 7.30a). Few additional nodes are regularly visited by the foresters with an additional 2% of the nodes being visited for an average inter-visit time of 250s. The Wytham Woods scenario will be very challenging for any data collection scheme due to the highly concentrated nature of the sink visits.

The correlation between inter-visit times and visit duration is low (see Figure 7.30b). The visit durations vary widely from 10s up to 300s allowing for sufficient time for data collection.

7.2.7.2 Algorithm Behaviour

The Adaptive P-scheme behaviour graph (see Figure 7.31a) shows how for the vast majority of cases less than 10% of the nodes are likely to be selected as storage nodes. These nodes are within the vicinity of the frequented cross-road or nearby the static gateway sink. For extremely relaxed data requirements, particularly low delivery ratios ranging from 0-30%, Adaptive P-scheme may select



(a) Adaptive P-scheme behaviour

(b) Adaptive R-scheme behaviour

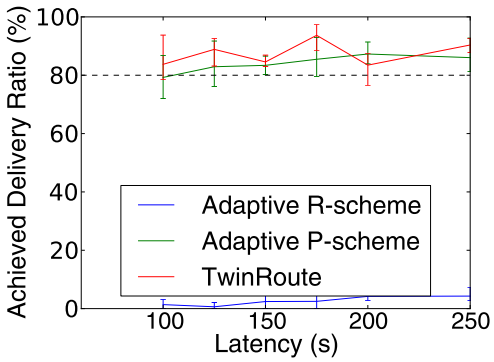
Figure 7.31: Algorithm behaviour graphs for the Wytham Woods scenario a) Adaptive P-scheme is predicted to only select less than 10% of the nodes as storage nodes for the vast majority of requirement scenarios and is likely relying on the few areas that are frequently visited by the roaming foresters. For extremely relaxed data requirements, particularly for low delivery ratios of 0-30%, a significantly higher amount of potential storage nodes is available, these scenarios however being somewhat unrealistic. b) R-scheme is predicted to severely struggle or even not be able to achieve the vast majority of data requirement scenarios. Once more, extremely low delivery ratios (0-15%) can be achieved, which again seem unrealistic requirements.

additional storage nodes in the deployment area. Scenarios in which the delivery ratio is set to such low levels are, however, unlikely to be selected in reality as the majority of data would potentially not be delivered, something most deployments try to avoid.

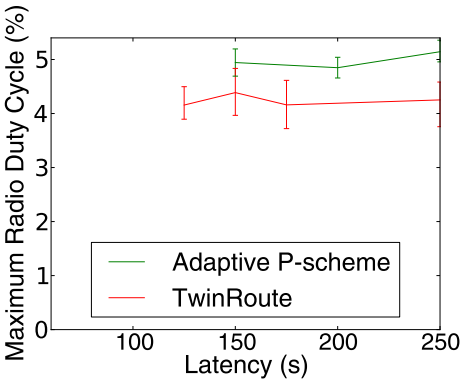
Figure 7.31b shows how Adaptive R-scheme is expected to fail to achieve the vast majority of data requirement scenarios, which is likely due to the restricted area that is frequently visited by mobile sinks. However, Adaptive R-scheme may achieve very low delivery ratio requirements in the range of 0-15%.

7.2.7.3 Experimental Results

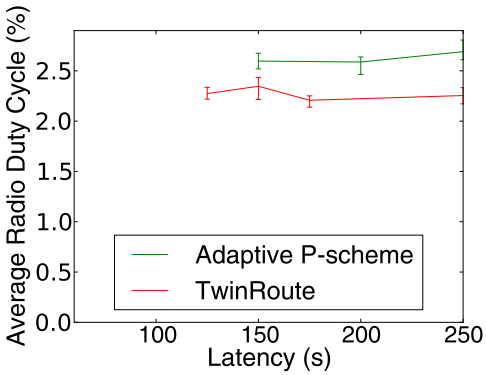
Figure 7.32a shows how Adaptive R-scheme fails to achieve any of the data latency requirements, due to insufficient sink mobility in the scenario. Adaptive P-scheme achieves data latency requirements above 150s, while marginally failing to do so



(a) Scheme performance comparison



(b) Maximum radio duty cycle



(c) Average radio duty cycle

Figure 7.32: Experimental results for the Wytham Woods scenario with a fixed delivery ratio of 80% and a range of data latency requirements. a) Adaptive R-scheme fails to achieve any of the data latency requirements as the mobility within the scenario is insufficient. Adaptive P-scheme achieves many of the requirements, however in an unreliable manner due to it struggling with the hotspot problem. TwinRoute, however, achieves the vast majority of latency scenarios due to its ability to circumvent the hotspot problem using its reactive component. b) TwinRoute achieves a markedly lower maximum radio delivery ratio than Adaptive P-scheme. c) A similar pictures is drawn for the average radio duty cycle that TwinRoute reduces by 15% when compared to Adaptive P-scheme.

for lower data latencies. Adaptive P-scheme fails due to it relying solely on the few storage nodes close to the gateway node that are overloaded. TwinRoute, however, achieves the vast majority of data latency requirements due to its ability to alleviate the hotspot problem using its reactive component.

TwinRoute achieves a lower maximum and average radio duty cycle than Adap-

tive P-scheme (see Figure 7.32b and c, respectively) as it can offload some of the data using the reactive component.

7.2.7.4 Summary

In this section we analysed and experimentally scrutinised a national park scenario that is based on real data from Wytham Woods. The scenario is characterised by very few areas that are frequently visited as well as many areas that are infrequently visited for long periods.

Once more, Adaptive R-scheme is unable to reliably collect data due to the sinks only visiting a small proportion of the area frequently. Adaptive P-scheme is close to achieving many of the data latency scenarios, however fails for the more stringent scenarios due to it suffering from the hotspot problem.

In contrast to this, TwinRoute achieves the vast majority of data latency scenarios. It can do so as it takes advantage of the scenarios' characteristics, with the proactive scheme collecting data in the frequently visited areas while the reactive component is utilised in areas that are infrequently visited for long durations.

7.2.8 Scenario Performance Classification

After having described, analysed and evaluated the performance of the algorithms, we will now analyse what scenario characteristics suit different data collection schemes.

In the previous experimental analysis we were able to analyse the scenarios according to two characteristics. Firstly, the inter-visit time which describes how frequently and reliably nodes are visited by mobile sinks in relation to the data requirements. Generally, a low inter-visit time also enables the WSN to achieve low latency requirements and vice-versa. Secondly, the spatial distribution of sink visits

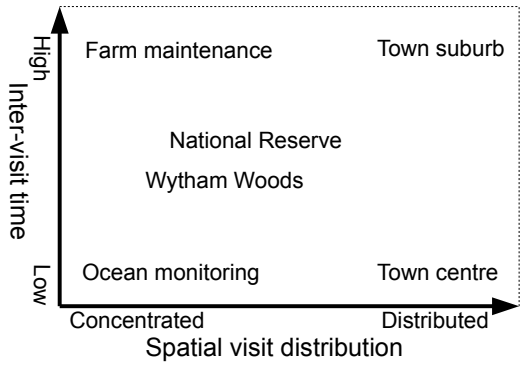


Figure 7.33: Categorisation of scenarios according to their characteristics with respect to the spatial distribution and inter-visit times of sink visits as found in the individual scenario analysis sections.

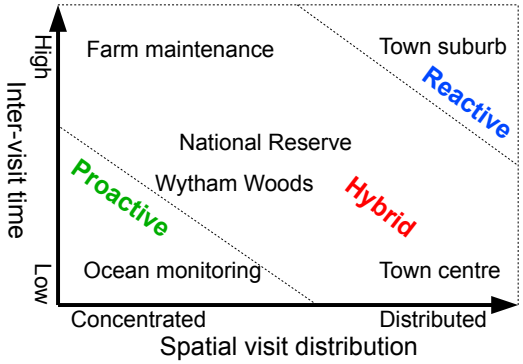


Figure 7.34: Proactive schemes perform well when few areas with low inter-visit times are found within the deployment area. Reactive schemes perform well when inter-visit times are high but relatively uniformly distributed through the whole WSN. Hybrid schemes outperform pure schemes in the remaining scenarios.

can range from concentrated, in which very few areas are visited, to distributed, in which the entire deployment area is visited by mobile sinks. The categorisation of previously discussed scenarios can be seen in Figure 7.33.

Based on our experimental results, Figure 7.34 shows in which scenarios different algorithms are preferred. It highlights how scenarios that have highly concentrated visits as well as low inter-visit times, such as the ocean monitoring scenario, suit proactive schemes as they are designed to utilise reliable pick-up opportunities. For such scenarios, reactive schemes are wholly inadequate as they rely on sink mobility for data collection, which is limited due to the concentrated visiting

nature of such scenarios.

On the contrary, scenarios with high inter-visit times and spatially distributed sink visits tend to suit reactive schemes, as they can easily react to the presence of sinks and adjust their routing behaviour accordingly, thereby still achieving the data requirements despite high inter-visit times. Proactive schemes, however, fail to perform well in such scenarios as they fail to select storage nodes due to the high inter-visit times.

Hybrid schemes can build on the strengths of the pure schemes and outperform these in most of the remaining scenarios, as can be seen in Figure 7.34. We have found that hybrid schemes perform particularly well in scenarios that exhibit some very frequently visited areas as well as a sufficient amount of random sink movement in some or all of the remaining areas. This is due to hybrid schemes being able to rely heavily on the proactive components, namely to reliably select storage nodes in frequently visited areas, while load-balancing significant amounts of data using the reactive component in the randomly visited areas.

We have shown how hybrid algorithms have significant strengths for the majority of scenarios due to their ability to combine the strengths of proactive and reactive components. Pure algorithms only perform well for a restricted set of scenarios due to their inability to deal with unsuitable scenario characteristics. Furthermore, hybrid algorithms can act like pure algorithms when required, allowing them to perform nearly as well as or better than adaptive pure schemes.

In conclusion, we have found that hybrid algorithms are more versatile and easily adapt to a large range of scenarios, by combining the strengths of pure schemes.

7.3 Summary

In this chapter, we first compared how adaptive versions of the data collection schemes perform when compared to optimally configured static schemes. Firstly, we illustrated how the deployment procedures for the adaptive algorithms are significantly simpler than the static versions. The required information prior to the deployment is significantly reduced as the adaptive versions automatically gather such information and configure the network accordingly. We have also shown how the adaptive versions perform as well as or better than the static versions for a range of scenarios. The adaptive versions performed particularly well when the WSN conditions, such as the sink mobility or data latency requirements, changed mid-deployment, as the adaptive versions reconfigured themselves accordingly.

Following this, we compared the adaptive (pure and hybrid) data collection schemes for a range of scenarios. We classified these scenarios according to their mobility characteristics, namely average inter-visit times as well as the spatial distribution of the sink visits. Furthermore, we analysed the behaviour of the algorithms for a large range of data requirements. It allows network engineers to identify boundaries of achievable data requirements as well as likely changes in algorithm behaviour when varying the data requirements.

After having experimentally explored a wide range of scenarios, we identified the characteristics that suited different data collection schemes. We can conclude that proactive schemes perform best in scenarios with spatially concentrated sink visits in combination with low inter-visit times. Such scenarios allow proactive schemes to identify reliable storage nodes within the areas that are frequently visited, leading to reliable data collection.

Reactive schemes perform well when the inter-visit times are marginally below the required latency threshold, and the sink visits are too distributed for proactive

schemes to select reliable storage nodes. In such scenarios, the reactive schemes constantly adjust their behaviour according to the mobile sink position, allowing them to achieve the data requirements.

For the majority of scenarios, hybrid algorithms perform as well as or better than the pure data collection schemes. The hybrid schemes perform particularly well in WSN scenarios that have a blend of characteristics, such that areas with concentrated sink visits can be utilised by the proactive components while other less frequently visited areas make use of the reactive component. We have also shown how hybrid schemes are vastly more flexible at adapting to a wide range of scenarios, simply because they can draw upon the strengths of the pure schemes and can utilise them according to the given WSN scenario.

Chapter 8

Conclusions

This thesis investigated delay tolerant WSNs with static sensors and mobile sinks with a particular focus on selecting or designing appropriate routing algorithms. We assume that the static sensor nodes have restricted capabilities with regards to their processing, storage, communication range as well as the energy available from their battery. The data generated at the sensors is assumed to be collected wirelessly by mobile sinks which have abundant processing and storage capabilities. Any data collected by the mobile sinks is uploaded to the end-user using a 3G or 4G uplink. We neither assume control over the mobile sinks, nor do we require that they are location-aware.

A plethora of data collection schemes have been suggested in the literature which can typically be divided into two classes: proactive and reactive. In proactive schemes, sensor nodes proactively forward their data to carefully selected storage nodes, from where data is collected opportunistically by roaming mobile sinks. Proactively pushing the data towards the destinations only incurs a small routing overhead, however it is less flexible and adaptable to changing data needs or sink visiting patterns. In reactive schemes, sensor nodes react to sink-initiated probes for data collection. Reactive routing can deal well with changing sink visiting

patterns due to its on-demand data forwarding approach.

Although a lot of previous work has focused on the design of energy-efficient routing algorithms for WSNs, little attention has been paid to the problem of optimising algorithms according to the characteristics and requirements of a specific application. One of our goals was to investigate the suitability of data collection algorithms with regards to the sink mobility characteristics as well as data latency and data delivery ratio requirements. Furthermore, we investigated the possibility of combining proactive and reactive schemes, thereby alleviating their respective weaknesses while drawing upon their strengths.

In what follows, we first summarise the thesis' main contributions and conclusions. Following this, we will lay out potential future work that has arisen from this thesis. Finally, we will conclude the thesis.

8.1 Contributions

In this section we describe the four main contributions of the thesis. Firstly, we detail how we designed, implemented and configured two adaptive data collection algorithms which can self-configure to suit a particular deployment and are shown to outperform static data collection algorithms. Secondly, we have described the design, implementation and configuration of a hybrid data collection scheme, named TwinRoute, which combines the strengths of pure data collection schemes. Lastly, we classified scenarios according to their characteristics and identified data collection schemes that suited the particular scenarios. Each of these contributions will now be described in more detail.

8.1.1 Adaptive Data Collection

We proposed novel adaptive versions of P- and R-scheme that self-configure themselves and adjust to change. The key assumption they make is that we can predict the future sink visiting pattern from the past. Consequently, nodes maintain sink visiting logs that are analysed accordingly in order to configure the network appropriately. Adaptive P-scheme and Adaptive R-scheme work in a distributed manner and require only local coordination between nodes. A sensitivity study was conducted for both algorithms in order to illustrate how their parameters impact their performance.

We have illustrated how the deployment effort of the adaptive versions is significantly reduced due to their self-configuring nature when compared to their static counter-parts. The information needed prior to the deployment is particularly reduced, as static versions require network engineers to obtain a significant amount of information to deploy such a WSN in an informed manner.

We have experimentally verified the algorithms' ability to adapt to change and have shown that they can perform as well as or better than the static versions. Adaptive P-scheme, in particular, manages to reduce the maximum radio duty cycle when compared to the static version by as much as 40% when the environmental conditions change, such as sink mobility or data requirements.

We believe that the proposed adaptive algorithms have significant potential to ease the cumbersome deployment of delay-tolerant WSNs with mobile sinks, that currently require much manual labour and a high level of information to be successfully deployed. Although the approach of predicting the future sink visiting pattern from the past is widely applicable, it may not be appropriate for all circumstances, particularly in scenarios that may involve frequent and dramatic change. However, we believe that in gradually-changing scenarios the adaptive algorithms will ease the deployment and use of delay tolerant WSNs with static sensors and mobile

sinks.

8.1.2 Hybrid Data Collection

The second key contribution of the thesis was the design, implementation and configuration of a hybrid data collection algorithms that combines aspects of both proactive and reactive approaches. The underlying assumption is that proactive algorithms have strengths when the mobile sink visiting patterns are predictable and frequent enough to select storage nodes, however struggle to deal with scenarios with random or frequently changing visiting patterns. In contrast to this, reactive algorithms exhibit strengths with changing mobile sink visiting patterns due to their on-demand data forwarding approach. Our proposed hybrid data collection algorithm, TwinRoute, combines the strengths of the underlying proactive and reactive routing components.

We empirically show how TwinRoute performs as well as or better than Adaptive P- or R-scheme for a very large range of scenarios, doubling the expected time-to-failure for some scenarios. Furthermore, we show how TwinRoute can adapt to a larger range of scenarios, while the pure adaptive schemes often only can achieve data requirements in scenarios that suit their specific needs. The flexibility of TwinRoute in combination with its self-configuring nature show that it is a broadly applicable choice.

8.1.3 Scenario Classification and Adaptive Algorithm Performance

The final major contribution was the analysis of different scenarios, and the study of which adaptive (pure and hybrid) algorithm best fits which scenario characteristics.

The comparative performance evaluation of the data collection schemes al-

lowed us to identify characteristics that suit proactive and reactive algorithms. We found that scenarios with concentrated sink visits with low inter-visit times suit proactive algorithms, as they enable the selection of reliable storage nodes. Conversely, we found that scenarios with relatively uniform spatial visit distributions with near latency inter-visit times suit reactive algorithms, as they can quickly respond to a change in mobile sinks location, while proactive schemes are unlikely to reliably select storage nodes.

Our empirical study also highlighted how hybrid data collection schemes significantly outperform the pure adaptive schemes by as much as 100% in scenarios with mixed characteristics, such as those in which some areas have concentrated reliable sink visits that proactive components can make use of, while other areas are accessed less often and make good use of reactive trees. Furthermore, we showed how hybrid schemes can adapt to a much larger range of scenarios while performing nearly as well as or better than the pure algorithms.

8.2 Future Work

After having described the contributions of the thesis, we will now highlight potential ideas for future work.

Currently, although there is a plethora of routing algorithms, very few studies have focussed on assessing their relative merits in different scenarios. This thesis has attempted to do so for tree-based routing algorithms, and for a particular type of network consisting of fixed sensor nodes and mobile sinks. Clearly, this effort must be extended to cover a larger repertoire of algorithms and consider more general network settings.

Another topic investigated but not included in this thesis (see ”Evolution and Sustainability of a Wildlife Monitoring Sensor Network” [46]) was the prioritisation

of data in proactive schemes such that high-priority data used more energetically expensive routes (e.g. through a 3G router) while low-priority data utilised energy efficient routes when possible. In the paper we showed how our simple approach resulted in a 57% reduction of the expected time-to-failure. Extending the priority-based approach to different data collection schemes, such as reactive or hybrid ones, is a topic for future work.

Many of the underlying parameters have to be manually configured in a lengthy time consuming trial-end-error process, in our case Contiki OS' network stack parameter, such as the MAC-wakeup frequency, as well as many data collection scheme parameters. Developing and testing adaptive configuration approaches for such parameters would alleviate many tedious deployment tasks while potentially optimising the network throughout its deployment cycle. Generic configuration approaches do exist [34], though have not been adopted or tested under real deployment conditions which typically raise many subtle and difficult issues. We found that many of these parameters were strongly tied to the sink mobility, namely sink speeds and visit durations, which could form a basis of an adaptive configuration approach.

This thesis investigated the algorithm performance with fixed sensor data generation rates and fixed sensor node layouts, however this could be extended by investigating bursty data loads as well as varying the sensor node topology. The impact on the performance of different algorithms is likely to differ and constitutes a topic for future work.

8.3 Summary

In this thesis we investigated the scenario dependent performance of proactive, reactive and hybrid data collection schemes in delay-tolerant WSNs with mobile

sinks. We introduced novel adaptive data collection schemes that are able to adjust to changing sink visiting patterns or data requirements, thereby significantly easing the deployment of a WSN. We have shown that the hybrid data collection scheme is useful and for the vast majority of scenarios performs as well as or better than the pure algorithms, offering a self-configuring and easily deployable choice for network engineers. An analysis of deployment scenarios was conducted according to specific scenario characteristics, such as the geographic distribution as well as the frequency of sink visits. We were thus able to determine which scenario characteristics favour the deployment of different data collection algorithms. Overall, this thesis has shed light on the limitations of existing static data collection schemes, has proposed novel adaptive and hybrid algorithms, and has revealed interesting links between scenario characteristics and algorithm performance.

Bibliography

[1] Neighbor discovery for ip version 6 (ipv6). http://tools.ietf.org/html/rfc4861.

[2] Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications. page P802.11, NOV 1997.

[3] K. Akkaya and M. Younis. Energy-aware routing to a mobile gateway in wireless sensor networks. In *Global Telecommunications Conference Workshops (GlobeCom)*, pages 16–21, 2004.

[4] Stephen A. Ellwood Andrew Markham, Niki Trigoni. Effect of rainfall on link quality in an outdoor forest deployment. In *Proceedings of the International Conference on Wireless Information Networks and Systems, Athens, Greece*, July 2010.

[5] Fan Bai and Ahmed Helmy. *Wireless Ad Hoc and Sensor Networks - A Survey of Mobility Models*, chapter 1. Springer, 2006.

[6] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. The important framework for analyzing the impact of mobility on performance of routing for ad hoc networks. *AdHoc Networks Journal - Elsevier*, 1:383–403, 2003.

[7] Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(3):55–66, 2001.

[8] Arnab Chakrabarti, Ashutosh Sabharwal, and Behnaam Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *The second International Workshop on Information Processing in Sensor Networks (IPSN*, pages 129–145, 2003.

[9] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 29–42, New York, NY, USA, 2006. ACM.

[10] Simon Duquennoy, Fredrik Österlind, and Adam Dunkels. Lossy Links, Low Power, High Throughput. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2011)*, Seattle, WA, USA, November 2011.

[11] Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Cecilia Mascolo, Bence Pásztor, Niki Trigoni, and Ricklef Wohlers. Evolution and sustainability of a wildlife monitoring sensor network. In *SenSys '10: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2010. ACM.

[12] Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Cecilia Mascolo, Bence Pásztor, Niki Trigoni, Ricklef Wohlers, and Kharsim Yousef. Evolution and sustainability of a wildlife monitoring sensor network: the wildsensing system. 2011.

[13] E. Ekici, Y. Gu, and D. Bozdag. Mobility-based communication in wireless sensor networks. *IEEE Communications Magazine*, 44(7):56–62, July 2006.

[14] S.R Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Global Telecommunications Conference(GlobeCom), Vol. 1*, pages 377–381, 2003.

[15] Jonathan W. Hui and David E. Culler. Ip is dead, long live ip for wireless sensor networks. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 15–28, New York, NY, USA, 2008. ACM.

[16] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[17] S. Jain, R.C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mob. Netw. Appl.*, 11(3):327–339, 2006.

[18] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 217–229, New York, NY, USA, 2003. ACM.

[19] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, New York, NY, USA, 1999. ACM.

[20] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Net-*

working, edited by Charles E. Perkins, Chapter 5, pages 139–172. Addison-Wesley, 2001.

[21] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM.

[22] Abdelmajid Khelil, Pedro Jose Marron, and Kurt Rothermel. Contact-based mobility metrics for delay-tolerant ad hoc networking. In *MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 435–444, Washington, DC, USA, 2005. IEEE Computer Society.

[23] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *1st Intl Conference on Embedded networked sensor systems (Sensys)*, pages 193–204, 2003.

[24] J.-Y. Le Boudec and M. Vojnovic. Perfect simulation and stationarity of a class of mobility models. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2743–2754 vol. 4, March 2005.

[25] HyungJune Lee, Martin Wicke, Branislav Kusy, Omprakash Gnawali, and Leonidas Guibas. Data stashing: energy-efficient information delivery to mobile sinks through trajectory prediction. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 291–302, New York, NY, USA, 2010. ACM.

[26] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[27] Chenyang Lu, Guoliang Xing, O. Chipara, Chien-Liang Fok, and S. Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 381–390, June 2005.

[28] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang. TTDD: Two-tier Data Dissemination in large-scale wireless sensor networks. *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on ACM MOBICOM*, 2003.

[29] J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Intl Conference on Computer Communications (Infocom)*, pages 1735–1746, 2005.

[30] J. Luo and J.-P. Hubaux. Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. In *Proc. of the 24th IEEE INFOCOM*, 2005.

[31] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux. Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In *DCOSS*, pages 480–497, 2006.

[32] Jian Ma, Canfeng Chen, and Jyri Salomaa. mwsn for large scale mobile sensing. *Journal of Signal Processing Systems*, 51:195–206, 2008. 10.1007/s11265-007-0118-x.

[33] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI), 2002.

[34] Andrew Markham and Niki Trigoni. The automatic evolution of distributed controllers to configure sensor network operation. *The Computer Journal*, March 2010.

[35] C. Mascolo and M. Musolesi. Scar: context-aware adaptive routing in delay tolerant mobile sensor networks. In *Intl conference on Wireless communications and mobile computing (IWCMC)*, pages 533–538, 2006.

[36] Moteiv. Tmote sky datasheet, 2006. http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.

[37] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, Nov. 2006.

[38] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[39] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, IEEE Workshop on*, 0:90, 1999.

[40] Elizabeth M. Royer, P. Michael Melliar-smith, and Louise E. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *In Proceedings of the IEEE International Conference on Communications*, pages 857–861, 2001.

[41] A.A. Somasundara, A. Kansal, D.D. Jea, D. Estrin, and M.B. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing*, 5(8):958–973, 2006.

[42] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, New York, NY, USA, 2005. ACM.

[43] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Hybrid push-pull query processing for sensor networks. *Proceedings of the GI Workshop on Sensor Networks*, 2004.

[44] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, 2000.

[45] David W. Macdonald Andrew Markham Cecilia Mascolo Bence Pasztor Salvatore Scellato Niki Trigoni Ricklef Wohlers Vladimir Dyo, Stephen A. Ellwood and Kharsim Yousef. Evolution and sustainability of a wildlife monitoring sensor network: the wildsensing system. *Accepted to appear in ACM Transactions on Sensor Networks*, 2011.

[46] David W. Macdonald Andrew Markham Cecilia Mascolo Bence Pasztor Salvatore Scellato Niki Trigoni Ricklef Wohlers Kharsim Yousef Vladimir Dyo, Stephen A. Ellwood. Evolution and sustainability of a wildlife monitoring sensor network. In *8th ACM Conference on Embedded Networked Sensor Systems (SenSys 2010), Zurich, Switzerland*, 2010.

[47] Z.M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting sink mobility for maximizing sensor networks lifetime. In *Annual Hawaii Intl Conference on System Sciences*, 2005.

[48] Ricklef Wohlers, Niki Trigoni, Rui Zhang, and Stephen Ellwood. Twinroute: Energy-efficient data collection in fixed sensor networks with mobile sinks. In

Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM), May 2009.