Principled tuning and structuring methods for sensornets

Jonathan Tate

Submitted for the degree of Doctor of Philosophy

University of York

Department of Computer Science

October 2009

Abstract

Wireless Sensor Networks, or *sensornets*, are an emerging class of information processing system. Unlike conventional computer networks, sensornets compose many independent *motes* into self-organising ad hoc networks. Motes are small, cheap computers, each equipped with independent power supplies, wireless communication capability, and sensors with which to passively monitor the physical environment in which they are embedded.

Distributed applications distil voluminous raw data about sensed physical phenomena into meaningful information with utility to end users. All nodes are equal peers, responsible for data production, processing, storage, delivery and consumption. Management is decentralised. Interactions with the real world imply real-time requirements.

Designing and configuring sensornets is difficult. Motes have limited resources, limited connectivity with peers, and are liable to fail. Unstable physical environments imply variation in sensor data volume and content, and variation in wireless connectivity. Countless configurations of application software, network topologies, middleware, and protocols are possible, with no guarantee that any combination is sufficiently performant and reliable.

This thesis aims to contribute toward understanding, and solving, the problems associated with designing large scale self-managing sensornets. We argue that properties of sensornet behaviour can be measured and quantified such that objective evaluation and comparison among the set of candidate configurations is feasible. Significant improvements in measurable attributes of sensornet behaviour can be obtained through appropriate design decisions in network protocol selection and logical configuration.

Existing protocols can be tuned for specific deployments, maximising performance while retaining confidence of correct behaviour accumulated from previous experience. Where protocol tuning cannot deliver the required behaviour as a consequence of inherent properties of protocols, logical networks overlaid on physical networks enable further improvement, providing a platform in which protocols can fulfil application requirements.

Contents

Abstract iii

List of figures vii

List of tables xi

Acknowledgements xv

Declaration xvii

1 Introduction 1

1.1 Structure of this chapter . . . 1
1.2 Background . . . 1
1.3 Issues . . . 2
1.4 Statement of hypothesis . . . 5
1.5 Organisation and structure . . . 6

2 Literature survey 11

2.1 Sensornets . . . 11
2.2 Organisation and structure . . . 17
2.3 Energy and resource usage . . . 28
2.4 Evaluating sensornets . . . 32
2.5 Routing protocols . . . 39
2.6 Real-time behaviour . . . 57
2.7 Summary . . . 66

3 Measuring sensornet behaviour 67

3.1 Protocol failure modes and weaknesses . . . 67

3.2 Protocols and their controlled factors . . . 71
3.3 Measurable attributes of solution quality . . . 76
3.4 Offline static protocol tuning results . . . 80
3.5 Online dynamic protocol tuning . . . 91
3.6 Summary . . . 103

4 Engineering methods for sensornet protocol tuning 105
4.1 The protocol tuning problem . . . 105
4.2 Tuning protocols by a Design Of Experiments approach . . . 108
4.3 Comparing tunings in training and deployment networks . . . 120
4.4 Summarising factor-response relationships . . . 135
4.5 Summary . . . 136

5 Evolutionary methods for sensornet protocol tuning 137
5.1 Characteristics of evolutionary search . . . 138
5.2 Experimental method . . . 138
5.3 Results . . . 144
5.4 Quality of evolving solutions . . . 151
5.5 Summary . . . 154

6 Protocol tuning robustness 157
6.1 Robustness analysis . . . 157
6.2 Network response metrics . . . 158
6.3 Spatial distribution . . . 160
6.4 Spatial density . . . 161
6.5 Node count . . . 167
6.6 Summary . . . 173

7 Managing sensornets using cellular logical networks 175
7.1 Protocols for self-configuring sensornets . . . 175
7.2 Intracellular timing synchronisation . . . 179
7.3 Summary . . . 204

8 Distributed state management 207
8.1 Short-term duty schedule coordination . . . 207

8.2 Medium-term duty schedule coordination . . . 208
8.3 Long-term duty schedule coordination . . . 230
8.4 Summary . . . 267

9 Intercellular synchronisation 269
9.1 Cellular sensornets . . . 270
9.2 Intercellular timing coordination . . . 274
9.3 Summary . . . 294

10 Evaluation and conclusions 295
10.1 Evaluation . . . 295
10.2 Future work . . . 299
10.3 General conclusions and closing remarks . . . 301

Abbreviations 303

References 305

Appendix 342

A Protocol definitions 343
A.1 TTL-Bounded Gossip . . . 343
A.2 Implicit Geographic Forwarding . . . 344

List of Figures

3.1 Transmit/receive final states 82
3.2 Delivery success rates 83
3.3 Delivery route characteristics: straightness 85
3.4 Delivery route characteristics: hopcounts 85
3.5 Coverage versus broadcasts 86
3.6 Path speed 87
3.7 Path latency 88
3.8 Coverage versus time 89
3.9 Energy efficiency characteristics 90
3.10 Potential energy savings 91
3.11 Source distance vs frequency 94
(a) All delivery attempts for unmodified gossip 94
(b) Unsuccessful delivery attempts for gossip variants 94
3.12 Destination distance vs frequency 95
(a) All delivery attempts for unmodified gossip 95
(b) Unsuccessful delivery attempts for gossip variants 95
3.13 Best-path hops vs frequency 97
(a) All delivery attempts for unmodified gossip 97
(b) Unsuccessful delivery attempts for gossip variants 97
3.14 Best-path time vs frequency 98
(a) All delivery attempts for unmodified gossip 98
(b) Unsuccessful delivery attempts for gossip variants 98
3.15 Total hops vs frequency 99
(a) All delivery attempts for unmodified gossip 99
(b) Unsuccessful delivery attempts for gossip variants 99

5.1 Comparing TBG solution quality using Factorial Design, SPEA2 and TA . 146
5.2 Comparing IGF solution quality using Factorial Design, SPEA2 and Two-Archive 149

6.1 Network response metrics under controlled spatial distribution 162
6.2 Reachability metrics under controlled spatial distribution 162
6.3 Network response metrics versus spatial density under TBG 165
6.4 Reachability metrics versus spatial density under TBG 166
6.5 Network response metrics versus spatial density under IGF 168
6.6 Reachability metrics versus spatial density under IGF 168
6.7 Network response metrics versus node count under TBG 171
6.8 Reachability metrics versus node count under TBG 171
6.9 Network response metrics versus node count under IGF 173
6.10 Reachability metrics versus node count under IGF 174

7.1 Influence of peer synchronisation timing feedback 186
7.2 Worst case initial state transitioning to equilibrium state 186
7.3 Normalised metrics for variant A 196
7.4 f_α versus epochs to equilibrium 198
7.5 n versus epochs to equilibrium 199
7.6 Adding node to stable system 200
7.7 Removing node from stable system 201
7.8 Lost pulses 202
7.9 Phantom pulses 203
7.10 Jitter 204
7.11 Clock drift 205

8.1 Finite State Machine for CDAP states 210
8.2 Synchronisation window management 214
8.3 Window size 224
8.4 Proportion of time spent in CDAP states after convergence 225
8.5 Proportion of time with cell coverage levels after convergence 226
8.6 Mean energy consumption rates after convergence 228
8.7 Dynamic schedule slots with duty period mutual exclusion 229
8.8 Dynamic schedule slots with managed duty period redundancy 230

8.9 Finite State Machine for ADCP states 234
8.10 Development of R_1 over time for values of p_ω 253
8.11 Epochs required to reach $R_1 = n$ stable state versus p_ω 255
8.12 Epochs required to reach $R_1 = n$ stable state versus l 256
8.13 Progress of R_1 as a function of time as ADCP replaces a failed node 259
8.14 Progress of R_1 as a function of time as ADCP removes a surplus node . . . 260
8.15 Average percentage of total time in ADCP states for first 10 epochs 262
8.16 Average percentage of total time in ADCP states for first 100 epochs 263
8.17 Standard deviation of proportion of time in $ACTIVE$ versus time 265
8.18 Standard deviation of proportion of time in $ACTIVE$ over different timescales 266

9.1 Routing between individual nodes and between cells in cellular sensornets . 273
(a) Routing between individual nodes 273
(b) Routing between cells 273
9.2 Intercellular phase error 279
9.3 Progressive adjacent cell phase alignment 281
9.4 Convergence of U_1 as a function of time (standard y-axis) 289
9.5 Convergence of U_1 as a function of time (logarithmic y-axis) 290
9.6 Epochs to stable synchronous equilibrium versus f_β 291
9.7 Epochs to stable synchronous equilibrium versus f_β (log-log plot) 292
9.8 Epochs to stable synchronous equilibrium versus degree of connectivity, a . 293

List of Tables

2.1 Complexity characteristics of proactive routing protocols 47

2.2 Complexity characteristics of reactive routing protocols 49

3.1 Normalised metrics for all variants 102

4.1 Phase 1: τ_α values for metrics $M_1 - M_3$ 124

4.2 Phase 2: p-values for controlled factors X_1-X_6 and interactions for metrics M_1-M_3 for protocol A in Ξ 125

4.3 Phase 2: p-values for controlled factors X_1-X_5 and $X_7 - X_8$ and interactions for metrics M_1-M_3 for protocol B in Ξ 126

4.4 Best-known protocol tunings: controlled factors 128

4.5 Best-known protocol tunings: measured responses 130

5.1 Phase 1: τ_α values for metrics $M_1 - M_{3a}$ 140

5.2 Best known TBG tunings 144

5.3 Network performance for best known TBG tunings 145

5.4 Best known IGF tunings 148

5.5 Network performance for best known IGF tunings 148

5.6 α, β and γ for combinations of MOEA and sensornet protocol 153

7.1 Convergence times for metrics 197

8.1 Guard conditions for CDAP Finite State Machine in figure 8.1 220

8.2 Proportion of time in protocol states 224

8.3 Proportion of time for cell coverage 226

8.4 Mean energy consumption rates (Watts) 227

8.5 Guard conditions for ADCP Finite State Machine in figure 8.9 248

8.6 Average proportion of total time in ADCP states for first 500 epochs 263

Acknowledgements

I would like to take this opportunity to thank those individuals and organisation without whom the production of this thesis and associated work would not have been possible. Their input and support has been of inestimable value and is greatly appreciated.

Firstly, I would like to thank my supervisor, Dr. Iain Bate, for the advice, opinions, and insight he has provided throughout this work. I would also like to thank Simon Poulding, Benjamin Woolford-Lim, and Prof. Xin Yao, with whom I have collaborated and co-authored publications as detailed in the declaration on page xvii. I must also thank Kelvin Lau for presenting my work at a conference I was unable to attend.

Moving from individuals to organisations, I would like to thank members of the Real-Time Systems group, the SEBASE project, and the anonymous reviewers of conference papers and journal articles, for their feedback and opinions. These have been of critical importance in refining ideas and identifying opportunities. I would also like to thank EPSRC and BAE Systems plc for funding the research described in this thesis.

Finally, but most importantly, I would like to thank my wife, family, and friends, for their support, encouragement, and inexhaustible patience.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2006 - 2009. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Some parts of this thesis have been published in conference proceedings and journals; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author. All papers for which I am the primary author were co-authored with my PhD supervisor, Dr. Iain Bate, who provided guidance and feedback.

- J. Tate and I. Bate. YASS: A scaleable sensornet simulator for large scale experimentation. In *Proceedings of the 31st Communicating Process Architectures Conference*, pages 411-430, September 2008. [287]

  The content of this paper, though not directly included in this thesis, was used throughout in designing and executing sensornet simulation experiments.

- J. Tate, I. Bate, and S. Poulding. Tuning protocols to improve the energy efficiency of sensornets. In *Proceedings of the 4th IET UK Embedded Forum*, pages 51-61, September 2008. [295]

  S. Poulding suggested the application of statistical model fitting techniques to identify significant factors and summarise findings in the multi-objective optimisation problem addressed in part of this paper.

  The content of sections 3.2-3.3 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Tuning complex sensornet systems using principled engineering methods. In *Proceedings of the 16th IEEE International Conference on the Engineering of Computer Based Systems*, pages 275-284, April 2009. [291]

  The content of sections 3.2, 3.3, 4.2 and 4.4 is derived, in part, from the content of this paper.

- J. Tate, B. Woolford-Lim, I. Bate, and X. Yao. Comparing design of experiments and evolutionary approaches to multi-objective optimisation of sensornet protocols. In *Proceedings of the 11th IEEE Congress on Evolutionary Computation*, pages 1137-1144, May 2009. [296]

  B. Woolford-Lim selected the Multi-Objective Evolutionary Algorithms and associated parametric values utilised in the relevant section of this paper, implemented these in software, and executed some simulation instances on University of Birmingham computation facilities. I developed the experimental design and network model, executed some simulation instances, designed the comparative analysis method, analysed the experimental results and extracted the conclusions. Further collaboration using an additional MOEA yielded the results presented in chapter 5. X. Yao provided editorial input and is the PhD supervisor of B. Woolford-Lim.

  The content of chapter 5 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Understanding behavioural tradeoffs in large-scale sensornet design. In *Proceedings of the 1st IEEE International Workshop on Quantitative Evaluation of Large-Scale Systems and Technologies*, pages 1085-1091, May 2009. [292]

  The content of section 3.1 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Energy efficient duty allocation protocols for wireless sensor networks. In *Proceedings of the 14th IEEE Conference on Engineering of Complex Computer Systems*, pages 58-67, June 2009. [288]

  The content of section 8.2 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Sensornet protocol tuning using principled engineering methods. *The Computer Journal*, 2009. Advance Access publication on 19 August 2009 at http://comjnl.oxfordjournals.org/cgi/content/abstract/bxp077 [290]

  The content of sections 4.1, 4.2, 4.3, and chapter 6, is derived, in part, from the content of this article.

- J. Tate and I. Bate. An improved lightweight synchronisation primitive for sensornets. In *Proceedings of the 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 448-457, October 2009. [289]

  The content of section 7.2 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Do sensornet protocol variants yield real benefits? In *Proceedings of the 16th IEEE International Conference on the Engineering of Computer Based Systems* (to appear), March 2010. [293]

  The content of section 3.1 is derived, in part, from the content of this paper.

- J. Tate and I. Bate. Maintaining stable node populations in long-lifetime sensornets In *Proceedings of the 15th IEEE Conference on Engineering of Complex Computer Systems* (to appear), March 2010. [294]

  The content of section 8.3 is derived, in part, from the content of this paper.

Copyright © 2009 by Jonathan Tate

The copyright of this thesis rests with the author. Any quotations from it should be acknowledged appropriately.

Chapter 1

Introduction

Wireless Sensor Networks, or *sensornets*, are an emerging class of information processing system. Unlike conventional computer networks, sensornets are composed of independent nodes embedded into their physical environment. Each node is an equal peer, and is responsible for data production, processing, storage, delivery and consumption. Difficulties arise owing to the large numbers of nodes required to support many intended applications, the limited resources available at each node, and the likelihood of node failures. This thesis aims to contribute toward understanding, and solving, the problems associated with designing large scale self-managing sensornets.

1.1 Structure of this chapter

This chapter introduces the subject matter considered in this thesis and defines the document structure. The research hypothesis is defined, providing a high-level description of the research questions to be addressed. This high-level problem definition is decomposed into a logical sequence of related subproblems, addressed in the remaining chapters of this document. Consideration is given to the composition of individual research units into a consistent whole, elaborating the overall direction taken during the research work.

1.2 Background

Wireless Sensor Networks, also known as *sensornets*, compose many autonomous *motes* into ad-hoc networks for distributed sensing and processing applications. Motes are small, cheap computers equipped with independent power supplies, wireless communication ca-

pability, and sensors with which to passively monitor their environment. Real-time interaction with the real world is not merely a factor to consider in sensornet design; it is the fundamental purpose of the sensornet.

Distributed applications distil voluminous raw data about sensed physical phenomena into meaningful information with utility to end users. Sensor-actuator networks also interface some motes with actuators to actively influence the environment, completing the control feedback loop. Interaction with the physical environment implies that sensornets have real-time requirements in addition to functional requirements. Typical applications include environmental monitoring and surveillance.

1.3 Issues

Sensornets are currently at an interesting point in their evolution. MIT Technology Review identifies sensornets as one of the *ten emerging technologies that will change the world* [245]. Some real-world deployments have been implemented but at relatively small scale. These small trials have validated the concept as workable and useful. However, despite considerable interest, wireless sensor networks have not yet made the transition from the laboratory to commonplace real-world usage. What is holding back these real-world deployments?

1.3.1 Resource limitations

Motes have greatly restricted resources, owing to the requirements for small physical size and low cost [104]. Advances in technology lower the cost for a given hardware capability, but in sensornet applications these advances may be translated into lower unit costs rather than increased per-mote capability [316]. Increasing the number of nodes allows larger applications to execute in-network and implies greater redundancy to deal with equipment failure, but also implies an increased volume of network traffic and a larger management and coordination problem [130].

Each mote can apply its limited local resources independently of other motes to address small localised problems, but cooperation between motes is essential to harnessing the total resources of all motes in the sensornet to tackle realistic-sized problems [216]. The distributed applications running on motes effectively pool computation and storage resources [9], but energy cannot be transferred between motes. It is often desirable to

consume energy at a similar rate across a sensornet to prevent the occurrence of network or sensing voids [222]. When sufficient motes have run out of energy for the sensornet application to stop functioning the sensornet is effectively dead [31]. This can occur even if the majority of motes continue to enjoy substantial unused energy reserves, which is likely where nodes near network gateways bear a disproportionate burden [51].

Wireless communication components are generally the most energy-hungry subsystem of sensornet motes [86]. Sensornet mote resources are so tightly constrained that any energy efficiency improvement is worthy of evaluation. Many small savings achieved over the sensornet lifetime accumulate into large savings, but it is not always clear how even small savings can be achieved. Energy efficiency improvements are possible both in sensornet hardware and software. Network lifetime can be extended by using radio communications more efficiently [281].

1.3.2 Design optimisation

There exists a trend toward increasing complexity in sensornet system design [297]. The hardware and software modules from which sensornets are composed, and the protocols which specify the bindings between these components, may in turn be composed from simpler subcomponents [83]. These compositions may deliver more sophisticated functionality intended to address specific in-network situations, or address specific perceived defects in simpler components, potentially delivering improved performance. This necessarily increased complexity can, however, make it harder to reason about, or to predict, the behaviour of such systems. Good engineering practice favours the selection of simpler systems and subsystems whose behaviour is readily understood and well established by previous experience [237]. Greater complexity also tends to imply higher costs, which is contrary to common design goals of sensornet systems [81].

One approach to sensornet design optimisation is to propose, design, test, and implement new custom protocols for each sensornet deployment [83]. While such an approach potentially allows for very efficient sensornet designs, this potential is unlikely to be realised in practice [297]. New protocols are unlikely to have been as thoroughly tested in real deployment environments as older, more established, and more general equivalents. This is of particular significance in sensornets; inherently destined for deployment in, and exposure to, the unpredictable real world, it is desirable to employ tried-and-tested protocols which have been shown to work well in previous work.

An alternative approach is to tune existing protocols to optimise performance, offering robustness to changing or unknown network conditions [111]. Given a model of the deployment environment, hardware platform, distributed application, and sensor data inputs, a protocol can be tuned for this specific deployment context. It is also useful to evaluate any change in network behaviour induced if the deployment context should vary from that for which the protocols were tuned [339]. A fragile network configuration that fails under moderately changed deployment conditions is unlikely to offer acceptable robustness [100]; it may be more desirable to select an alternative configuration that offers acceptable performance across a range of environmental conditions, rather than a tuning offering optimal performance under optimal conditions but poor performance under other conditions.

1.3.3 Managing sensornet systems

Network designers may lack confidence that a given design will function adequately in a given scenario if there is little historical data or experience upon which to draw. There are few experimental studies employing large numbers of sensornet nodes, and consequently little measurement of sensornet protocol scalability [102]. Until such time as there exists a large number of previous sensornet installations from which to draw experience, simulation offers a low-risk and low-cost environment in which to assess the viability of proposed solutions, and to improve the quality and relevance of any putative solutions. The results will assist in the identification and mitigation of risks for real deployments of large scale sensornets.

Sensornet motes typically have limited resources and limited communication capacity, so lightweight protocols and algorithms are favoured throughout [86, 120]. The designer should select the most efficient option which supports the distributed application, but these lightweight options may not scale well in significantly larger systems [100]. System cost can be measured in attributes such as algorithmic complexity, storage cost, energy cost, latency and so forth; the growth of these costs in network size or other system attributes may follow any number of relationships [244], but any cost growth of order greater than O(1) will eventually exceed the available resources.

The sensornet must be large enough to achieve the physical sensor coverage and processing requirements of the distributed application, but ideally no larger so as to avoid the inefficiency and suboptimal performance observed in unnecessarily large systems [244].

The large number of nodes anticipated in real sensornets implies that device failure, frequent network change, and task dynamics, must be managed as standard operating conditions rather than exceptional circumstances. The sensornet system must automatically self-reconfigure to cope with these changes, as it is generally impractical or perhaps impossible to manage this process centrally [87].

1.3.4 The problem

The correct functioning of a distributed sensornet application is dependent on the underlying network platform providing sufficiently performant services. These requirements are typically defined as quantified attributes of a *Quality of Service* (QoS) contract [5, 222].

Given a set of hardware motes of fixed capability deployed into the environment, and assuming that software is infinitely malleable, operators can compose software-controlled motes into any number of sensornet configurations [297]. However, it is not obvious whether any configuration is sufficiently performant to satisfy application requirements.

Acceptable sensornet configurations are those which, when deployed and measured, perform sufficiently well in each attribute specified in the QoS contract [111]. Exceeding these minima may allow the operator to extract greater value from the sensornet, or may provide a safety margin against unanticipated events.

It is desirable to maximise each measurable performance attribute. However, complex interrelationships may exist between controllable design factors and measurable performance attributes, such that obtaining sensornet configurations sufficiently performant in all specified attributes is non-trivial [8].

Sensornet motes may be unreliable, inaccessible, and have limited resources [216]. Sensornets composed of many such motes must react to events on the ground in a timely manner [121]. It follows that manual sensornet management is impractical [254], and centralised management becomes less likely to prove successful as network size grows [87]. It follows that self-organising and self-managing sensornets are required for practical deployments of realistic scale [216].

1.4 Statement of hypothesis

Following from the discussion of subject matter above, the hypothesis of this thesis is defined as follows:

Properties of sensornet behaviour can be measured and quantified such that objective evaluation and comparison among the set of candidate configurations is feasible. Performance improvements in measurable attributes of sensornet behaviour can be obtained through appropriate design decisions in network protocol selection and logical configuration.

1.5 Organisation and structure

The remaining chapters of this thesis describe an integrated set of work elements, in a logically organised sequence of steps, which address the research questions defined by the hypothesis. A number of new results are presented in this thesis, each framed in terms of existing results described in the literature and within an integrated research plan. Chapter 3 demonstrates that interesting phenomena exist, and that it is possible to study these at reasonable cost. Chapters 4 to 6 demonstrate that these phenomena can be exploited to improve system performance for a given system, but this potential improvement is bounded, and system performance tends to decline with scale. Chapters 7 to 9 demonstrate that larger systems can be made to resemble smaller systems, so as to avoid this scalability problem.

1.5.1 Background

Before we begin to address the novel work presented in this thesis, it is important to define the underlying research problem and consider existing related work. Chapters 1 and 2 address these matters.

- Chapter 1 defines the research hypothesis from which the following work is derived, and the structure of the thesis in which this work is described.

- Chapter 2 presents a survey of related work discussed in the literature. Topics covered include the nature and composition of sensornets, the limited resources available to sensornet nodes, the requirement to use these limited resources with maximal efficiency, and the scale-dependent behaviour of sensornets. Subsequent chapters provide further analysis of the literature where appropriate.

1.5.2 Define the problem and methods for its measurement

A measurable and repeatable response in network behaviour can be induced by controlling protocol parameters. It is demonstrated that, for a very simple system with one controllable factor and multiple measured responses, the complexity of the factor-response relationship can be such that no single value of the controlled factor obtains the best observed value of all measured responses. This indicates that the protocol tuning problem is a multi-objective optimisation problem. Chapter 3 demonstrates that this optimisation problem can feasibly be addressed, in acceptable time and with acceptable computational cost, for systems of realistic scale by an experimental approach.

- Chapter 3 considers the experimental method and infrastructure required to address the problem of large-scale sensornet protocol optimisation. Physical experiments with realistic-scale sensornets are currently infeasible owing to the associated cost, overhead and complexity. Methods are presented with which to render feasible the analysis of large-scale sensornets with acceptable experimental cost by simulation. Definitions are provided for controlled factors associated with protocols and measured responses by which solution quality is measured. The requirements for meaningful and repeatable simulation experiments are discussed, alongside an analysis of energy efficiency improvements possible without changing network behaviour.

1.5.3 Optimise protocol tunings for a given network instance

Chapter 3 shows that sensornet protocol tuning is a multi-objective optimisation problem, with the potential to achieve significant measurable improvements in network behaviour, and which can be addressed in a realistic timescale. Chapters 4 to 6 demonstrate that the multi-objective optimisation problem of protocol tuning can be made smaller by a dimensionality reduction approach based on identifying statistically insignificant controllable factors. Reasonable defaults are obtained and assigned to insignificant factors, and experimental effort is concentrated on the remaining statistically significant factors.

Removing insignificant factors allows the experimenter to either complete an experiment suite of fixed size in shorter time, or to explore the problem in greater depth in fixed time. Search-based techniques based on principled experimental design and evolutionary approaches are used to explore the factor-response relationship. The impact on network performance of deploying into an environment dissimilar to the tuning environment is

explored.

- Chapter 4 discusses the tuning of existing sensornet protocols. Sensornet protocol tuning is formulated as a multi-objective optimisation problem based on sound Design Of Experiments theory. Objective measures of solution quality are defined. The optimisation problem is addressed by principled search methods based on Factorial Design.

- Chapter 5 addresses the optimisation problem using evolutionary search methods based on Multi-Objective Evolutionary Algorithms. The relative costs of applying principled and evolutionary methods to the optimisation problem are examined.

- Chapter 6 applies objective measures of solution quality to assess the robustness and stability of protocol tunings to changes in deployment networks.

1.5.4 Optimise a network instance for a given protocol tuning

Chapters 4 to 6, described above, demonstrate that when protocols tuned for a given deployment environment are deployed into a dissimilar environment the resulting performance may be acceptable, but not necessarily matching that attainable where the tuning and deployment environments are similar. Similarly, it is shown that some desirable network properties tend to decline with network size. It follows that methods to manage network size are desirable, to attain a logical network of the required size from a potentially larger physical network. Chapters 7 to 9 describe an integrated set of lightweight protocols to achieve this goal for a cellular sensornet. These are clustered sensornets [126] in which the geographic region covered by a sensornet is divided into cells, and all nodes located within the boundaries of a given cell form a cluster.

Within each cell a fixed number of nodes are selected at any given time to participate in the network from a potentially larger set of candidates, the remainder being kept available as spares to replace failing nodes. Within this set of participating nodes, the responsibility for handling intercellular traffic flows, computation tasks and sensing duty is fairly and equally distributed into equal timeslots. A biologically-inspired synchronisation primitive is defined with which to coordinate these activities throughout the sensornet, exploiting the *synchronisation* phenomenon between cells and the *desynchronisation* phenomenon within cells. A virtual network of the fixed required size is obtained, even if the physical distribution of nodes is irregular such that some cells enclose more nodes than others.

- Chapter 7 considers the application of protocols to optimise a given sensornet. A lightweight primitive for global synchronisation in the absence of a global clock is defined. This primitive forms the basis for a suite of protocols to coordinate sleep schedules in cellular sensornets, implicitly defining a smaller, more efficient virtual network within a larger physical network.

- Chapter 8 presents methods to achieve stable mutual exclusion and managed redundancy in unreliable networks. Different approaches are applied at different timescales. This enables the duty burden of nodes to be minimised, in turn enabling nodes to switch to low-power dormant states when their active participation is not required. By reducing the average rate of energy consumption at each node, the operational lifetime of the entire network is extended.

- Chapter 9 defines a model of multicellular sensornets and intercellular connectivity. Within each cell, separate instances of the protocols defined in chapters 7 and 8 execute. A lightweight hierarchical system management mechanism is defined, with which activity across the network of cells is synchronised.

1.5.5 Evaluation and conclusions

Having considered the novel contributions of this thesis, we can then consider whether these were successful in addressing the research problem, and whether further related research problems exist. Chapter 10 addresses these matters.

- Chapter 10, the final chapter, concludes the thesis. The hypothesis outlined in chapter 1 is revisited in the context of the novel contributions of the preceding chapters, from which supporting evidence is derived. Directions for further work are given, concluding with some general remarks.

Chapter 2

Literature survey

This chapter presents a review of the literature pertaining to sensornets, concentrating on the issues addressed by this thesis. The novel work presented in the following chapters builds upon the existing body of research literature discussed here. Significant further discussion of the literature appears in subsequent chapters when necessary or helpful.

2.1 Sensornets

Research in *Wireless Sensor Networks*, or *sensornets*, necessarily touches upon many research topics of Computer Science and some of Electronic Engineering, drawing upon the existing work in related fields. However, the unique characteristic of the sensornet field is the interplay and integration of these foundation subjects, yielding a distinct topic worthy of further study in its own right. From the amalgamation of existing work in such diverse subjects as *computer networks*, *energy management*, *wireless communications*, *distributed computation*, *data management*, and numerous others, emerges the topic of sensornet research with its own goals, challenges, and opportunities, distinct from those of the related work from which it is drawn. Sensornets are typically *highly scalable*, *energy-aware*, *geography-aware*, *data-centric*, *application-specific*, *self-configuring*, and *self-adapting*.

2.1.1 Terminology

Tilak *et al.* [300] provide a taxonomy of sensornet models. The following terms are defined:

- **Sensor** - an independently-operating device that implements the physical sensing of environmental phenomena and reports measurements through wireless communication. Typically consists of five components: sensing hardware, memory, battery,

embedded processor, radio transceiver.

- **Observer** - the end user interested in obtaining information disseminated by the sensor network about *phenomena*, phrased as *interests* to the network.
- **Interests** - a query regarding physical phenomena sensed by the network, addressed to the sensornet as a single entity rather than to a specific *sensor*.
- **Phenomenon** - the entity or issue of interest to the *observer* that is sensed and potentially analysed/filtered by the sensornet.
- **Measurements** - a set of discrete sample values taken from a single *sensor*.
- **Source** - the start-point of a communication within a sensornet, typically a producer of *measurement* data or *interests* data.
- **Sink** - the end-point of a communication within a sensornet, typically a consumer of *measurement* data or query response data.

2.1.2 Characteristics

Wong and Arvind [326] state that the defining characteristics of a sensornet are:

- **Data-centric**: The primary focus of a sensornet is to produce data about the physical environment in which the network resides by taking readings from sensors attached to network nodes. Often the processing of this data will occur at a central hub, although in-network processing is possible.
- **Inter-node communication**: Nodes can communicate with their neighbours within a physical distance limited by wireless communication methods, typically of the order of metres but almost always less than the physical diameter of the network (hence ruling out a fully connected point-to-point network model).
- **Mobility**: Sensornet nodes are largely static, although some nodes may be mobile, and hence the sensornet is largely a static network notwithstanding node failure or temporary disconnection for power conservation.
- **Data transfer**: Sensor data is produced at *source* nodes and is transported to *sink* nodes at which it is consumed or otherwise processed. Often the *sink* node is a central hub, although a fully decentralised peer-to-peer model is feasible.

Estrin *et al.* [87] state that a sensornet is also *application specific*. Rather than accommodating a wide variety of applications, a sensornet is typically designed and deployed with a specific task (or set of tasks) in mind. This allows the designer to optimise the network design to these specific goals, and implement application-specific in-network processing distributed across the nodes of the network. This is in contrast to the application-agnostic routers of a traditional data network, which merely forward data unchanged and hence cannot implement application-specific optimisations. In sensornets, data transmission is typically more expensive than data processing [193, 235].

Kahn *et al.* [150] also note that sensornet nodes must *consume extremely low power*, *operate in high volumetric densities*, *have low production cost and be disposable*, *be autonomous and operate unattended*, and *be adaptive to the environment*. These requirements influence sensornet hardware and software design such that both should be considered simultaneously. The extreme nature of the problem, extreme scale of deployment, and extreme resource constraints, have required researchers to seek new solutions rather than recycle existing solutions from related problem domains.

2.1.3 Applications

Sensornet applications typically involve the distribution of a number of sensor nodes in a geographic region, where each node gathers raw data about its surroundings. Nodes may be equipped with seismic, magnetic, thermal, visual, infrared, acoustic, radar or GPS sensors [7]. These raw data can be processed and interpreted in-network to monitor a wide range of ambient conditions including temperature, humidity, vehicular movement, lighting condition, pressure, soil composition, noise levels, presence or absence of certain types of objects, mechanical stress levels, speed, direction, or object size [87].

Akylidiz *et al.* [7] examined the published literature and reported the following list of proposed applications:

- **Military applications**: Sensornets can be an integral part of C4ISRT systems (command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting), performing tasks such as *monitoring friendly forces*, *battlefield surveillance*, *reconnaissance of opposing forces and terrain*, *targeting*, *battle damage assessment*, *nuclear/biological/chemical attack detection*.

- **Environmental applications**: *Forest fire detection*, *environment biocomplexity mapping*, *flood detection*, *precision agriculture*.

- **Healthcare applications**: *Telemonitoring of human physiological data*, *tracking patients and doctors*, *drug administration in hospitals.*
- **Home applications**: *Smart environment.*
- **Commercial applications**: *Environmental control in offices*, *detecting and monitoring car theft*, *managing inventory control*, *vehicle tracking and detection.*

Culler and Estrin [68] provide a classification of sensornet applications which is more generic, albeit rather more simplistic. They consider applications to be differentiated into the categories:

- **Monitoring space** e.g. *environmental monitoring*, *habitat monitoring*, *precision agriculture*, *indoor climate control*, *surveillance*, *treaty verification*, *intelligent alarms.*
- **Monitoring things** e.g. *structural monitoring*, *ecophysiology*, *condition-based equipment maintenance*, *medical diagnostics*, *urban terrain mapping.*
- **Monitoring the interactions of things with each other and the encompassing space** e.g. *monitoring wildlife habitats*, *disaster management*, *emergency response*, *ubiquitous computing environments*, *asset tracking*, *healthcare*, *manufacturing process flow.*

From the above, we see that even a high-level of classification of sensornet applications can encompass multiple dimensions, as we categorise a set of candidate applications against sets of assessment criteria. Care must be taken in assessing the results obtained in sensornet research to separate the application-dependent and -independent factors, and to avoid confounding of sensornet factors among environmental factors.

Many other proposals for applications of sensornets have been published, in numbers too great to list comprehensively. However, some of the more interesting work describes real deployments of sensor networks. Although of limited scale, they provide interesting insight into the strengths and weaknesses of the sensornet concept and the current capability of realising these networks in practice.

Habitat monitoring sensornets [49] were deployed to monitor the ecology of the Leach's Storm Petrel [282, 283]. Interestingly, the investigators observed that the node failure rate was much higher than expected, and that unexpected emergent behaviour was observed which could not be predicted without *building complete systems and deploying them in realistic conditions* [282].

Wark *et al.* [315] describes a Wireless Sensor-Actuator Network, monitoring and influencing livestock behaviour through per-animal actuators. Closing the sensor-actuator loop in this manner provides an interesting challenge for network modelling and simulation. Accurately predicting livestock reactions, even within short periods, requires far greater modelling detail than applied in purely observational longer-term studies [282,283]

Other proposed sensornet applications include the *Smart Kindergarten* project [37] to provide a rich problem-centric learning environment, monitoring underground structures for failure [181], detecting and tracking plumes of gaseous substances in the environment [266], detecting ruptures in pipelines [276], structural health monitoring [89, 159, 210], automated distributed surveillance [129], biomotion capture [19], tracking moving objects [85, 173, 264], and lighting control for media production [223]. This is not an exhaustive list, and countless other applications could be conceived.

2.1.4 Related topics

Active networks are computer networks in which *the switches of the network perform customized computations on the messages flowing through them* [298]. Such networks are considered *active* because *nodes can perform computation on, and modify, the packet contents*, and are considered by Tennenhouse *et al.* [298] to be *a natural step beyond traditional circuit and packet switching.* It is evident that sensornets are also *active networks* if any in-network filtering, aggregation or other processing is implemented.

Sensornets are similar to *Mobile Ad-hoc Networks* (MANETs) in that they are composed of small, low-resource computer nodes connected through a wireless network. Akyildiz *et al.* [7] consider MANETs to be *the closest peers to sensor networks*, noting that MANETs place far greater importance on mobility but may be composed of far fewer nodes. More significantly, sensornets are *data-centric* rather than *application-centric* [110], and are usually composed of non-mobile nodes with environmental sensors [86].

VANETs are Vehicular Ad-hoc NETworks, a specialisation of MANETs in a particular application domain. In a VANET, each network node is associated with exactly one vehicle or roadside station. This restricts the possible node movement, with which the network must cope, to a set of patterns associated with vehicles, which tend to travel along defined routes (e.g. roads) within certain speed boundaries dependent on the class of vehicle [213]. Greater predictability of behaviour makes it easier for an algorithm to determine the best action. However, if there is a one-to-one relationship between vehicle and network node,

it is important that reachability of all nodes is guaranteed.

Smart Dust is an emerging technology of wireless sensing networks, similar to a conventional sensornet [316]. They utilise nodes that are physically very small, on the order of millimetres, and are deployed in a correspondingly small physical area, typically of the order of metres [150] rather than the kilometres of sensornets [326]. Other than size, perhaps the most significant difference between a *Smart Dust network* and a conventional sensornet is the medium for wireless communications; *Smart Dust* nodes typically communicate by optical transmission rather than the radio-frequency communication of sensornets [150].

Specknets, or *Speckled Computing Networks*, are a dense and non-static wireless network of thousands of very small, resource-constrained nodes called *specks* collaborating to process data [326]. *Specknets* resemble sensornets, but are *application-centric* rather than *data-centric*, have shorter internode communication range (tens of centimetres), are highly mobile, and apply a fully decentralised peer-to-peer model dissimilar to the *sources* and *sinks* of sensornets. However, it could be argued that *specknets* are merely an atypical subset of sensornets rather than a separate discipline.

Distributed computing divides a computation task into smaller subtasks, allocates these subtasks to independent computing resources, then compiles the set of subtask results into a result for the initial task. Any sensornet which performs in-network aggregation or collaborative sensing could be considered a *distributed computing system*, under Stankovic's definition of *a collection of computers connected by a communications subnet and logically integrated in varying degrees by a distributed operating system and/or distributed database system* [273]. However, this definition implies a somewhat more thorough level of integration and coordination between computers than is necessarily evident in the highly adaptive environment of a sensornet, and does not require that the computers sense or influence their physical environment.

Distributed databases and *parallel databases* are similar to conventional databases in that they manage, archive and query data, but distribute the data and processing across multiple physical nodes [220]. The motivation for distribution varies, but is typically to make the dataset available at multiple physical locations, to enable processing of a dataset which is too large for any single node, to provide resilience to equipment failure through redundancy, or to enable load-balancing. Sensornets do not necessarily implement *distributed database* functionality, but some applications will require this. Barbará [26] considers that the main challenges in the wireless mobile context are *communication asymmetry*, *frequent*

disconnections and *power limitations*.

Real-time databases are databases which exhibit predictable behaviour, conform to explicit timing constraints, and apply time-cognisant protocols to handle *deadlines* and *periodicity constraints* associated with database activity [240]. A *temporal database* contains *time-varying data* [219] which has *temporal validity* [240]. Ramamritham [240] suggests that the time-dependent characteristics of *real-time* and *temporal* database behaviours are strongly linked, as most *real-time databases* are inherently *temporal databases*, whereas Özsoyoğlu and Snodgrass [219] perceive these to be distinct research topics that may be applicable to a given system but are otherwise separate. Necessary interaction between sensornets and their physical environments induce real-time requirements, hence many sensornet applications must maintain *real-time databases*.

Both *real-time* and *temporal* databases extend conventional database theory in the time domain, with *temporal database theory* considering the impact of time upon data, and *real-time database theory* considering the impact of time upon the database system which manages data. Although there is generally no requirement that the data managed by a *real-time database* be *temporal data*, this pairing is likely in sensornets producing and processing data for real-time consumption or controlling actuators. In addition to the general real-time scheduling theory reapplied to database tasks, *real-time databases* and *temporal databases* must consider the notion of *temporal consistency* [240].

Safety critical system design techniques seek to quantify, control, and minimise, the possibility of system failure. This is generally undertaken where the failure or malfunction of a system could cause or allow unacceptable risk or damage to people or other entities, and expose the system operator to financial or legal penalties [37]. Sensornets yield data about the real world which, if incorrect or incomplete, could lead the network operator to incorrect or inappropriate conclusions. Of more concern would be *Wireless Sensor-Actuator Networks* [6] which interact with and modify their environment in response to observations, typically without human intervention, and for which special consideration must be given to the potentially deleterious effects of system misbehaviour.

2.2 Organisation and structure

Sensornets, like any computer network, are systems in which a set of independent computing nodes produce, process and consume data, and around which data are distributed as required by the application. Unlike conventional wired networks, a combination of lim-

ited resources and coupling between physical environment and network imposes certain requirements and restrictions on sensornet structure.

2.2.1 Physical connectivity

Murthy and Garcia-Luna [209] summarise the effects of the *Physical* layer of the OSI [343] and TCP/IP [91] models, indicating that wired networks have *high bandwidth and topology that changes infrequently*, whereas wireless networks have *mobile nodes and limited bandwidth for network control*. They conclude that routing protocols widely used in wired infrastructure networks may not function effectively in sensornets because the underlying network is fundamentally different. It follows that the characteristics of sensornets must be considered in protocol design and evaluation.

Woo *et al.* [327] characterise inter-node communications of a sensornet node as a small number of high-quality links to near neighbours and a large number of low-quality links to other nodes. High-quality links exist to neighbours in the *effective region* which increases with transmit power. Beyond a certain distance, very distant nodes are occasionally able to successfully transfer packets. However, between these points is a *transitional region* in which average link quality falls off smoothly with distance, but individual pairs of nodes exhibit high variation. Asymmetric links are common in the transitional region.

2.2.2 Regulating access to the wireless medium

Interaction between sensornet nodes occurs through a shared wireless medium. As discussed in section 2.2.1, the characteristics of this shared wireless medium have significant influence on the nature and quality of connectivity that can be achieved between nodes. Assuming that interaction is possible between two or more nodes, there is a requirement to control access to the shared medium to ensure effective connectivity is possible. *Medium Access Control* (MAC) protocols [284], located in the *Data Link* layer of the OSI model [343] and the *Link* layer of the TCP/IP model [91], provide this control mechanism.

IEEE 802.11 [137] defines a standard MAC protocol and physical layer specification for *Wireless Local Area Networks* (WLANs). It operates in wireless networks, supports an *ad hoc* mode in which nodes communicate directly rather than through a base station, and supports multi-hop mesh networking. Dedicated hardware support for IEEE 802.11 is provided by some wireless networking chipsets, such as the CC1000 transceiver [54] employed in the popular MICA2 [65] mote platform. As IEEE 802.11 is a well-established

standard, supports the behaviours required in sensornets, and is well-supported in wireless networking chipsets, it is a reasonable choice for sensornets. However, it is not ideal for lightly-loaded sensornets. *Idle listening*, which occurs at nodes listening to a wireless medium which is not currently supporting meaningful activity, consumes almost as much power as actively receiving useful data [140]. This is wasteful; considerable energy savings could be achieved if nodes can power-down wireless communications modules during these inactive periods.

Two other IEEE standards, IEEE 802.15.1 and IEEE 802.15.4, relate to wireless MAC protocols which also use the 2.4GHz radio frequency used by IEEE 802.11, but were designed to support networks small computing devices similar to sensornet motes. IEEE 802.15.1 [138] defines a standard MAC protocol and physical layer specification for *Wireless Personal Area Networks* (WPANs). This standard was derived as a formalisation of version 1.1 of the *Bluetooth* protocol stack, though later versions of Bluetooth are not ratified as IEEE standards [33]. IEEE 802.15.4 [139] defines a standard MAC protocol and physical layer specification for *Low-Rate Wireless Personal Area Networks* (LR-WPANs). This standard is most commonly implemented as the lower levels of the *ZigBee* protocol stack, which adds a network layer, an application layer, an application framework, and a number of security services [342].

The IEEE 802.11 (WLAN) [137], IEEE 802.15.1 (Bluetooth) [138], and IEEE 802.15.4 (ZigBee) [139] standards share the same 2.4 GHz ISM band, but are not designed to co-operate. Shuaib *et al.* [261] demonstrate that significant reduction in network throughput is observed where two or more of these protocols are simultaneously active in a shared radio environment. In each combination the throughput of both protocols is reduced, but with non-trivial relationships. IEEE 802.11 is greatly more affected by Bluetooth than by ZigBee, and ZigBee interference has more effect on the IEEE 802.11 uplink than the down-link. The magnitude of the influence of IEEE 802.11 performance on ZigBee performance depends greatly on the extent to which the spectrum of the chosen channels of operation co-inside. It follows that designers of sensornets employing one or more of these standards must consider both the internal activity of the sensornet, and interference with the radio environment into which the sensornet is deployed. For example, the designer may select significantly different wireless communications configurations for sensornets deployed into urban settings, in which widespread IEEE 802.11 usage is common, and for sensornets deployed into rural settings, in which it is not.

Numerous MAC protocols have been described in the literature. Although all MAC protocols address a similar problem, the specific characteristics of any given protocol may render it more or less effective in a specific usage context. It follows that the task of selecting the most appropriate MAC protocol requires the network designer to carefully consider the strengths and weaknesses of each candidate in this context. For example, sensornet MAC protocols may prioritise good energy efficiency over other performance attributes. Comprehensive surveys of MAC protocols that are compatible with usage in sensornets are given by Demirkol *et al.* [76], and by Naik and Sivalingam [211]

Some MAC protocols have greater capability than others to function effectively in highly dynamic networks, in which network membership changes frequently or in which nodes are highly mobile. The mobility patterns typical of sensornets, in which node connectivity changes occur on a timescale of the order of seconds to minutes, are particularly difficult for MAC protocols to handle efficiently; the induced connectivity faults are *too frequent to be handled by repair mechanisms and too long lasting to be handled by retransmission* [66]. Reliable and unicast MAC protocols may aggravate the networking problems induced by this type of dynamic network behaviour [127]. The connectivity changes induced by node mobility may be misinterpreted as transmission errors. If the MAC protocol attempts to address these perceived, but non-existent, transmission errors, this may result in wasted bandwidth and energy.

2.2.3 Localisation

Localisation is required in sensornets to assign location context to sensor data, and to enable application-independent services such as *geographic routing* and *geographic storage* [301]; it is *a prerequisite to the utility of most sensor networks* [17]. If fitting a GPS unit to each node is infeasible, perhaps for reasons of cost, size, deployment area (e.g. underground) or energy consumption [117], a compromise is to have nodes infer their position from a small proportion of *seeds* which are aware of their location at all times [250].

Location inference mechanisms can be divided into the two categories of *range-based* and *range-free* [250]. *Range-based* techniques attempt to measure the distances between nodes, and include methods based on *received signal strength*, *time of arrival*, *time difference of arrival*, *angle of arrival*, and *Monte Carlo approaches*. *Range-free* approaches include *triangle intersection regions*, *hop-count distance estimation*, *Bèzier curve estimation*, and *radiointerferometric methods*. Euclidian distance bias, typically ignored in much

network research, should be accounted for in link generation when constructing sensornets or models of sensornets [285].

2.2.4 Mobility

The *mobility* model of a dynamic network in which node location varies with time warrants special attention in a sensornet. A tradeoff exists between energy consumption and accuracy. *Localisation* activity consumes energy and should be minimised to maximise network lifetime, but performing *localisation* too infrequently runs the risk of location data becoming stale and inaccurate and hence harming all network activities which rely on location [301]. The *Monte Carlo Localization* (MCL) algorithm [134] addresses this problem, and in fact relies on node mobility to function at all. MCL only works effectively with low node speeds, but the related *MSL* and *MSL** algorithms degrade more gracefully with node speed at the expense of greater computational complexity [250].

2.2.5 Data production

A sensornet is *a proxy for a continuous real-world phenomenon* which necessarily *samples that phenomenon discretely at some rate, with some degree of error* [110]. *Multi-resolution storage* stores information at different abstraction levels, using more storage where higher precision is required. *Data ageing* progressively decreases data precision and hence storage overhead over time, enabling *graceful degradation* assuming that newer data are more valuable to network operators than older data. High-quality query responses are produced for recent data, and lower-quality responses are still available for older data [101]. Application-dependent data ageing algorithm fine-tuning is recommended [101].

Shenker *et al.* [257] opine that *the overwhelming volume of observations produced by [network node] sensors is both a blessing and a curse*, producing unbounded volumes of data [75] with insufficient network and energy resources precluding forwarding all data to base stations for processing [257]. Localised algorithms perform *in-network processing* to minimise data flow volume [75].

Network snapshots exploit redundancy within sensor observations of nearby nodes [75], electing a small proportion of *representative nodes* queried through *snapshot queries* [161]. *Acquisitional Query Processing* [193] enables finer-grained control, collecting sensor data only in direct response to user queries, but consequently precluding retrospective data mining [75].

Methods exist to examine *self-similar* [195, 286] and *multifractal* [309] patterns in network *burstiness*, *flow density* and *bytes transmitted per unit time* [50]. Applications include detecting abnormal activity or failures, congestion control, and Quality of Service management by scheduling traffic appropriately [50, 176, 285]. Overlaying multiple *bursty* data streams counterintuitively increases *burstiness* rather than smoothing-out data flow, with repercussions for managing peak data flow periods [176].

2.2.6 Data distribution

Sensor data collected and aggregated within the network are important, whereas individual nodes are not [326]. It follows that sensornet data, not sensornet nodes, require unique identification [257]. *Named data* are categorised by *type* and *location* [257] attributes *external* to network topology, *relevant* to applications, *independent* of network topology, and utilised throughout network stack layers [125].

Data models define the data representation and semantics shared by all data model users [110]. Raw sensor observations occupy a 3-dimensional spatial-temporal datacube of coordinates x-coordinate, y-coordinate, and *time* [99] in planar sensornets. Non-planar networks, in which nodes occupy arbitrary locations in 3-dimensional space, imply a 4-dimensional datahypercube as the z-coordinate is added.

Shenker [257] describes network cost of canonical distributed data models for sensornets of n nodes:

1. *External storage*: Event descriptions delivered to external storage at cost $\mathrm{O}(\sqrt{n})$. In-network queries and responses cost $\mathrm{O}(\sqrt{n})$. Externally-generated queries and responses have zero network cost; all information is already external to the network.

2. *Local storage*: Event descriptions stored locally at producing node at zero network cost. Query flooding costs $\mathrm{O}(n)$ for internally- and externally-generated queries. Responses delivered to query source with cost $\mathrm{O}(\sqrt{n})$.

3. *Data-centric storage*: Event descriptions, labelled by *name*, stored at arbitrary network location at cost $\mathrm{O}(\sqrt{n})$. Query delivery and response both cost $\mathrm{O}(\sqrt{n})$ for internally- and externally-generated queries.

Data-centric storage is preferred for sensornets with many nodes, or detecting many events where many remain unqueried. *Local storage* is preferred where most detected events are queried [257].

2.2.7 Structure

Traditional wired networks often employ the *client-server* model, in which highly resourced servers service the requests generated by a larger number of lightly resourced clients [284]. This architecture works well for many applications, but may not be well suited to the sensornet domain. Sensornets are generally composed of large numbers of identical motes, deployed into the environment without fine control over location, which form self-organising ad hoc networks through a shared wireless medium [259]. Each node is generally responsible for similar sensing, storage and processing duties, such that each node can be considered an equal peer in the network. It follows that the selection of networking paradigms and protocols which are well suited to this structure is conducive to maximising performance and reliability [9].

The exact definition of *Peer-to-Peer* networks, commonly abbreviated to P2P, is debatable but the term generally is used to describe systems which *lack dedicated, centralized infrastructure, but rather depend on the voluntary participation of peers to contribute resources out of which the infrastructure is constructed* [252]. P2P systems allow the effective utilisation of computational resources spread across a network, and are potentially robust to failures of single nodes and hence are suitable for long-term storage of data [24]. It follows that P2P mechanisms are well-matched to the needs of homogeneous sensornets in which the identity of individual peers is less important than the identity of the information produced and managed by the distributed application [257].

Managing large numbers of equal peers may be difficult in the absence of centralised control structures, and where these peers are minimally resourced there may be insufficient capacity to support more sophisticated protocols and structures. In such circumstances it may be useful to reduce the size of the management problem by introducing hierarchy in the form of *clusters* [126]. Each cluster composes a number of independent nodes, located within a small geographic region, into a higher level structure which can be effectively addressed as a single entity. One node from each cluster is selected as the *clusterhead*, and is responsible for managing activity within its cluster, and managing interaction with other clusters [170]. Clustering may allow sensornets to achieve greater scalability and reduce energy consumption [318]. However, it is not guaranteed that clustered sensornets outperform non-clustered sensornets [310], and few comparative studies exist.

Within a cluster, the clusterhead is responsible for allocating responsibilities to non-clusterhead nodes [318]. These may include the collection of sensor data from the physical

environment, data storage, data processing, communication and interaction with nodes in neighbouring clusters. If all activity and data within a cluster is channelled through the clusterhead, it follows that this node will consume resources at a greater rate than non-clusterhead nodes. It may be desirable to provide nodes specifically intended for the clusterhead role which are equipped with greater resources, such as additional processing power, storage capacity, and energy reserves [334]. However, there remains the problem of ensuring that these supernodes are distributed evenly throughout the network, and forming clusters of lower resourced nodes around these supernodes.

If a non-clusterhead node should fail, the clusterhead can simply allocate tasks to other nodes within the cluster [143]. This enables the control mechanism to be decentralised, with decisions and actions implemented in close proximity, and without the overhead and delay of lengthy round trips to a central controller. Within a cluster, the clusterhead is a single point of failure [52]; if the clusterhead should fail, the entire cluster effectively fails and is removed from the network.

Clusterhead election protocols exist with which to replace failed clusterheads from the cluster population [143], or to cycle the clusterhead among the cluster population for wear balancing [186] or load balancing [170]. However, this is not possible if the clusterhead and non-clusterhead nodes have substantially different capability. It follows that homogeneous sensornets offer greater flexibility and reliability if a clustering approach is selected, if the correct functioning of any given node cannot be guaranteed throughout the sensornet lifetime.

An extension of the clustering concept is the notion of the *cellular sensornet* [338]. The physical region occupied by the sensornet is divided into *cells*. Any node located within the geographical boundaries of a given cell is considered to be a member of that cell. This avoids the requirement for complex and resource-consuming cluster membership assignment protocols, and avoids non-deterministic cluster allocation [118, 259]. However, this requires that nodes be aware of their location and the geographic cellular structure.

2.2.8 Scalability

The number of nodes deployed in a sensornet may be of the order of hundreds or thousands, or in extreme cases of the order of millions [7], hence *scalability* is a critical issue. Emergent properties may be manifest in large deployments that do not become evident in smaller deployments. It is infeasible to individually configure every node as there may simply be

too many, and more fundamentally their precise location is unlikely to be known before deployment [94]. Therefore, it may be useful to address sensornets at a level of abstraction higher than the individual nodes. Welsh and Mainland [317] describe *abstract regions*, defined in terms of radio connectivity or geographic location, as a high-level building block for sensornet programming; code written against *abstract regions* will be automatically *compiled down to the detailed behaviour of individual nodes.* Frank and Römer describe techniques to define *roles* which can be automatically assigned to nodes in a self-configuring sensornet, on the assumption that the behaviour of nodes will not be uniform throughout the network at all times [94].

The *Kairos macroprogramming system* [114] allows the programmer to express the desired *global behaviour* of the entire sensornet, automatically generating code to express the *local behaviour* of nodes. The *Regiment macroprogramming system* [215] similarly allows the entire sensornet to be programmed and tasked as a single entity, automatically *deglobalizing* high-level descriptions of required behaviour into node-level, event-driven programs. Systems such as *TinyDB* [193] allow a distributed database to be queried and programmed as a single entity at run-time, rather than compile-time as in *Kairos* or *Regiment.* Reprogramming deployed sensornets could use *application specific virtual machines* [178] such as *Maté* [177] to reduce the size of code to be distributed and interpreted in virtual machines on sensor nodes.

2.2.9 Standardisation

Standardisation of a computer system requires *the isolation and separation of the inter-dependencies which now exist between the various elements*, such as *inter-dependencies between programs and the characteristics of equipment components, between programs and the characteristics of other programs, and between equipment components and the characteristics of other equipment components* [206].

Initial exploratory work on sensornet standardisation was published by Culler *et al.* [67] and Toh *et al.* [304]. Standardisation of potentially any sensornet system element is possible to some extent, other than deployment-specific tasking and optimisation. Mirroring traditional platforms, greater standardisation is possible toward the lower layers of the network and application stack. Standardised APIs, protocols, and service description languages, enable interoperability within sensornet frameworks where specific deployment factors require selection of specific component instances from available alternatives.

The *ZigBee* standard [342] defines a layered model of interoperating protocols and interfaces for wireless networks in lightweight resource demands, security, and low power consumption are of the highest priority, and the typical network data loading is relatively light for extended periods. Design goals include scalability for systems of up to 65,000 nodes, ease of deployment, long battery life, robust security, and low cost [84].

Communication in ZigBee networks is implemented by RF transmission using the license-free and globally available 2.4GHz band. The IEEE 802.15.4 [139] standard defines a MAC protocol and physical layer specification for *Low-Rate Wireless Personal Area Networks* (LR-WPANs), which are used as the lower levels of the ZigBee stack. Above IEEE 802.15.4, ZigBee adds a network layer, an application layer, an application framework, and a number of security services [342].

ZigBee is of growing importance [27] as it is mature, and of the competing standards it currently enjoys the greatest vendor and end-product support [320]. As ZigBee is an *open standard* [23], it is possible for multiple manufacturers to supply compatible ZigBee devices. These devices can readily be composed into functioning sensornets, without modification, by application domain specialists. This interoperability support is a significant factor in the popularity of ZigBee as sensornets begin to find commercial applications beyond the research community [320].

Bluetooth [33] is perhaps the closest rival to ZigBee in terms of scope and applicability in typical sensornet applications such as industrial process control and monitoring [23]. Versions 1.1 and 1.2 of Bluetooth are ratified as IEEE standards [138], though later versions are not. Later Bluetooth standards are maintained by the Bluetooth Special Interest Group, as compared to the *open standards* approach of ZigBee [23].

Similar to ZigBee, Bluetooth defines a unified protocol stack for wireless networks of computing devices, communicating using the 2.4GHz band. However, there exist significant differences in the focus of Bluetooth and ZigBee. Bluetooth was designed to facilitate connections between devices used by humans within a 10m *Personal Operating Space* [23], whereas ZigBee is focussed toward industrial control applications [320]. ZigBee explicitly aims to *network* large numbers of devices in multi-hop networks, whereas Bluetooth aims to *connect* devices, often in single-hop pairs [23].

The performance characteristics of ZigBee and Bluetooth also differ significantly [23]. ZigBee can communicate over longer distances, though Bluetooth supports higher data rates. Bluetooth devices are constantly awake, whereas ZigBee devices are usually asleep.

It follows that Bluetooth device battery lifetimes are generally of the order of days, whereas ZigBee device battery lifetimes may be of the order of years. As these characteristics are of great significance in sensornets, it is possible to conclude that, whereas both Bluetooth and ZigBee provide a suitable standardised protocol stack, ZigBee is generally better suited in this context owing to its lower power consumption, greater range, and better support for multi-hop networks.

Sensor Webs are integrated systems of real or virtual sensors, interacting through a set of open and standardised protocols, producing data values which are subject to further processing prior to archival and presentation to users [263]. The *Sensor Web Enablement* (SWE) initiative by the Open Geospatial Consortium aims to standardise *the entire sensor web process of sensor description, sensor discovery, sensor tasking, and access of data observed by sensors* [263], for sensor systems in which the geographic location of sensors is a *critical parameter* [36].

SWE defines an open infrastructure, based on web services, with which end users can interact with a sensornet via the Internet to interact with multiple remote sensor sources [205]. The resulting web-accessible sensornets and archived sensor data can be *discovered, accessed and, where applicable, controlled using open standard protocols and interfaces* [36]. Unlike other sensornet standardisation initiatives, SWE applies only to the application layer. It assumes the existence of a functioning underlying infrastructure of mote hardware, and networking protocols, of the type considered in this thesis.

SWE provides specifications for interfaces, protocols and encodings [36] as a framework of service descriptions and XML schemas, but does not specify any implementation details. Any number of implementations of the SWE standards are possible, and all should be able to interoperate successfully. An example is the *Open Sensor Web Architecture* (OSWA) [58], which provides a middleware platform for sensornets, and is a complete implementation of the SWE specification.

Interoperability of standardised networking protocols is of concern when connecting two or more sensornets, or when connecting a sensornet to a dissimilar network such as the Internet. Some argue that sensornets are unlike the Internet and hence require specialised protocols to replace *Internet Protocol* (IP) [235] such as *Sensornet Protocol* (SP) [67], requiring gateways for network interconnection [311]. Others argue that IP can be successfully implemented in low-power wireless networks [271] if modified appropriately. For example, the *6LoWPAN* protocol is intended to carry IPv6 over IEEE 802.15.4

networks [208]. Hybrid approaches such as *Tenet* have been proposed [107] in which an IP backbone connects regions in which a specialised protocol is used, analogous to the Internet backbone connecting *autonomous systems*.

2.3 Energy and resource usage

An important feature of sensornets is that motes have highly restricted resource availability, owing to the requirements for small physical size and low unit cost. Limited availability of energy resources is particularly significant. When a mote runs out of energy it cannot participate in the sensornet. Eventually a state is reached in which there are insufficient active motes for the distributed application to function correctly; at this stage the entire sensornet is effectively dead [146]. Sensornet designs must therefore be sufficiently optimised to ensure correct operation for at least the specified operational lifetime.

2.3.1 Impact on sensornet lifetime

Energy awareness and *energy management* are themes running throughout most aspects of sensornet design and operation. The energy resources of nodes are typically small and cannot be replenished after deployment, so energy consumption is *the most important factor that determines sensor node lifetime* [239].

Dutta and Culler [81] identify the four main ways in which sensornet nodes consume energy as being *sensing*, *communication*, *computation*, and *storage*. As *each of these processes consumes a different amount of energy for each unit of useful work*, and battery-powered sensornets have finite energy resources, it follows that the useful lifetime of sensornets is a function of both the unit cost and usage frequency of each process type. In principle, any energy saving is desirable. In practice, however, it is generally useful to focus attention on the more expensive behaviours, as these may render insignificant the savings achievable from the less expensive behaviours.

Energy consumption is *the most important factor that determines sensor node lifetime* [239]. Network lifetime can be extended by using radio communications more efficiently, either by reconfiguring existing protocols or by creating new protocols that imply less traffic but are functionally equivalent [281]. However, in the latter case, there exists the requirement to prove that this functional equivalence holds true under all valid circumstances, and not merely in a handful of simplified or small-scale example scenarios.

Attaining an appropriate level of confidence may require lengthy and expensive field trials and the application of formal analysis methods. Where sensornets are required to perform as safety-critical systems there may exist legal requirements or mandated standards with which compliance is obligatory, and for which expensive compliance certification may be necessary [37].

2.3.2 In-network energy production

If nodes could *harvest* or *scavenge* sufficient energy from their environment, network lifetime could extend indefinitely [145]. In practice this is very difficult. There are practical limits on the amount of energy which can be extracted from the environment, owing to the physical composition of the region and the capabilities and efficiencies of electrical generation systems. Unless energy consumption is at least matched by energy production, nodes will necessarily deplete their resources and eventually fail. However, even if perpetual network lifetime is not possible, augmenting batteries with energy harvesting could increase sensor node lifetime sufficiently to fulfil application requirements.

The power output of generation facilities associated with per-node energy harvesting may vary over time. For example, it is obvious that solar cells are not effective during nighttime. Furthermore, there is generally no means to transfer energy between nodes, so as to balance energy reserves within a sensornet composed of motes with differing exposure to energy sources or equipped with different energy harvesting facilities. It may be possible to coordinate activity within a sensornet such that energy production and consumption is balanced at each node. This may be achieved by assigning tasks to specific nodes, scheduled to occur at specific times, using reference data on hardware platform energy usage and collected data on energy harvesting production [153]. However, this implies significant overhead in collecting and managing energy production data at each node, and requires the energy cost of each task to be highly predictable.

2.3.3 Energy impact of communications activity

Wireless communication is generally the most energy-hungry aspect of sensornet operation [86]. It follows that wireless communication represents an obvious target for optimisation, and as such is worthy of closer attention.

Energy efficiency improvements might be achieved by improvements in hardware or software. Possible hardware improvements include more efficient antenna designs [55],

improved support for component idling and performance-cost balancing [108], and *smart battery* technologies [171]. Modular sensornet architectures have been proposed that allow sensornet designers to take advantage of these advances without changing the entire hardware platform [82]. However, if sensornet designers select off-the-shelf COTS hardware, such as the popular MICA2 mote [65], there may be limited scope to alter the core hardware platform aside from connecting additional optional modules.

In this thesis we consider only software-driven improvements at the level of network middleware, assuming a fixed hardware platform and typical distributed sensing application. The software configuration can readily be changed following deployment of the sensornet into the physical environment of interest. *Post hoc* improvement of the mote hardware is often impractical after the sensornet has been deployed into the field [304]. However, some hardware platforms may be reconfigurable under software control, and hence can be modified in the field to function more efficiently.

Received signal power in sensor networks drops off rapidly with distance, as r^4 due to partial cancellation from ground ray reflection with short antenna heights [86, 235]. An interesting consequence of this non-linear relationship is that network routes containing many short-distance hops may be more energy-efficient than routes containing few long-distance hops, albeit at the expense of requiring more intermediate nodes to be awake to forward traffic [157]. Hop count, a commonly used cost metric in conventional protocols like *Routing Information Protocol* (RIP) [123], may be inappropriate in sensornets requiring better-suited replacements considering physical factors to be found.

Surprisingly, operating radios in *idle* listening mode often provides little power advantage over actively transmitting or receiving, and receiving can consume more energy than sending [239]. Observing the full potential benefit requires radio state to be managed harmoniously with network activity.

Reducing transmission power can reduce energy consumption directly, by consuming less energy per bit transmitted, and indirectly, by localising network activity and reducing collisions and contention [157]. This is significant as the major source of extraneous energy consumption is overhearing packets not intended for a given node [265].

Even small improvements in network efficiency can yield large gains in network lifetime. Sending a single bit of information 100m may consume more energy than 1000-3000 CPU instructions [68, 235], a cost incurred at both the sender and receiver node, and by any intermediate nodes in multi-hop paths which may grow as the network becomes larger.

Therefore, energy savings must be driven by energy-aware design throughout the network stack, rather than relying on improvements in hardware technology [239].

2.3.4 Improving energy efficiency through design

Raghunathan *et al.* observe that energy optimisation for sensornets is complex as it *involves not only reducing the energy consumption of a single sensor node but also maximising the lifetime of an entire network*, requiring dynamic tradeoffs between *energy consumption*, *system performance*, and *operational fidelity* [239], yielding *up to a few orders of magnitude* of improved lifetime.

Dutta and Culler [81] state that *abstractions that are consistent over the diversity of power management techniques remain elusive*, but effective generic techniques exist which can be implemented for specific subsystems, without necessarily being implemented across the entire system. The most common techniques are identified as [81]:

Duty-cycling Cycling power to a subsystem to reduce its average energy draw.

Batching Buffering multiple operations and executing them in a burst in order to amortize a high startup or overhead cost.

Hierarchy Ordering boolean operations by their energy consumption and invoking low-energy operations before high-energy ones when the desired result is a conjunction of the operations.

Redundancy reduction Reducing or eliminating redundancy through compression, aggregation, or message suppression.

Significant energy savings can be observed by identifying low-activity periods and rebalancing the energy-performance tradeoff, as commonly implemented when trading off CPU clock speed against power consumption and heat production [108]. This might be achieved by switching entire subsystems off for the majority of time, periodically switching them to high-cost, high-performance modes to process accumulated work in batches. For subsystems that are capable of operating at different performance levels, where lower performance is associated with lower energy cost, these subsystems could remain active at all times, with the sensor node operating system dynamically balancing the performance mode against the incoming workload. The former option may be preferable if sleep states offer a significant efficiency improvement over reduced performance states. However, the

latter option may be necessary if time-sensitive behaviour is incompatible with batching, for example where emergency signals must be delivered within deadlines shorter than the subsystem state transition latency [81].

Dong [78] defines three models of energy consumption against which sensornets can be optimised. The *packet-based model* assumes nodes consume energy only when transmitting. Energy consumption per node is directly proportional to the number of packet transmissions. Having fewer packet transmissions yields improved energy consumption. However, this model is unrealistic. Significant energy is consumed by nodes when not transmitting nodes, for example when idle. Nodes receiving packets can consume energy at a similar rate to transmitting nodes. The *time-based model* assumes that energy consumption per node is directly proportional to the time spent in active states. Having nodes spend more time in sleep states yields improved energy consumption. The *mixed model* combines the *packet-based* and *time-based* models. Unless sleeping, nodes consume energy at some base rate regardless of activity, with an additional component proportional to the number of packets received or transmitted.

2.4 Evaluating sensornets

In the broadest terms, a sensornet deployment can be considered successful if it fulfils the requirements of the network operator. However, determining whether this is true requires a mechanism with which to obtain appropriate measurements of the given sensornet system, and a requirements specification against which these measurements can be compared. It is also useful to evaluate sensornets at the design stage, to provide confidence that they will perform adequately when deployed. This also allows the comparative evaluation of different sensornet configurations, such that the best can be identified for selection and the underlying relationships between controllable factors and measurable metrics can be explored.

2.4.1 Design support

Software development for sensornet applications is *currently a cumbersome and error-prone task since it requires programming individual sensor nodes, using low-level programming languages, interfacing to the hardware and the network, only supported by primitive operating system abstractions*, revealing a *strong need for programming abstractions that*

simplify tasking sensor networks [248]. Römer proposes that *middleware* offers a solution [248]. Other significant work in improving the maturity and quality of sensornet software development includes the definition of *design patterns* [105], and support for *interface contracts* that can be checked both statically and dynamically [15].

Sensornet design and evaluation frequently requires *simulation* and *emulation*; large-scale *testbeds* or *field trials* are infeasible and costly [21, 280, 282]. Formal analysis of typical sensornets may [136] or may not [4, 242, 246] be feasible. Regardless of feasibility, it is rarely attempted [242]. Experimenters must determine acceptable accuracy-scalability tradeoffs [63, 69, 243]; simulation-derived results are meaningless if simulated behaviour does not sufficiently match real behaviour [47, 232] and are particularly sensitive to timing discrepancies [172]. Wireless communication models are usually the component with highest computational cost [212] but the greatest source of inaccuracy [40, 124, 162].

Discrete event simulations are well suited to computer network simulation [20]. *Simulation models* [21] are constructed, similar to those used in model checking [268], and executed in *simulation engines* [109]. Incorporating real application code, execution environments, hardware, network connections, or other real entities, into *simulation models* yields *emulation models* [109], improving accuracy but harming scalability [308]. Real and simulated entities interact directly in *hybrid simulations* [175, 198, 321].

2.4.2 Modelling sensornets

Models can be described in terms of *inputs*, *outputs*, *parameters*, *relationships*, *mappings to and from the real world*, *model limitations*, and *model reliability* [258]. *Prediction systems* [92] consist of *mathematical models* with *prediction procedures* for determining unknown parameters and interpreting results. Unfortunately, required model inputs are often not *measurable or even directly observable, in any useful engineering sense* [258].

Paulk paraphrases Box; *All models are wrong; some models are useful* [228]. Models are an *abstraction of reality* [258] or an *abstract representation of an object* [92]. Models cannot provide perfect representations of real entities, but can be close enough to yield meaningful and useful results. Calibration significantly improves model accuracy, but models derived solely from post-hoc analysis of particular data sets perform poorly [92]. Models should undergo both theoretical and empirical evaluation before considering results trustworthy [258].

Models may be designed specifically for each system, or fixed models reused for classes

of similar systems [92]. Translation between different models of a given system is frequently problematic and subjective, but useful results are possible if models display *tolerable correspondence* [119].

Govindan *et al.* [110] argue that sensornets require *novel data-centric routing mechanisms* and *reconsideration of traditional network and database interface layering.* Govindan [110] states *it might be necessary to collapse layers or selectively break abstraction boundaries for efficiency or robustness reasons* because *communication using [sensor node] radios requires significantly more energy than computation.* Their reasons for breaking abstractions and strict layering are laudable and understandable, and not without precedent; consider multi-layer network switching [251].

However, we should be cautious when discarding notions of abstraction and indirection found critically important elsewhere in Computer Science; *separation of a system into relatively independent modules with clearly defined interfaces is a hallmark of good design* [255]. Another interpretation, contradictory to Govindan [110], is that existing research has not yet discovered a *suitable* model or set of abstractions for reused across components, applications and networks, and which is sufficiently robust and efficient.

Traditional layered network models include the OSI model [343] and the TCP/IP model [91]. Govindan [110] modifies the OSI model, proposing a sensornet model with layers for *hardware devices*, *radio and MAC*, *packet delivery*, *local signal processing*, *data-centric routing*, *collaborative event processing*, and *applications.* Tilak [300] defines a higher-level, unlayered sensornet model of *network architecture*, *performance metrics*, *communication patterns and mechanisms*, *data delivery*, and *network dynamics.* Akyildiz [7] extends the OSI model with orthogonal planes of *power management*, *mobility*, and *task management*, representing issues cutting across all network layers. An obvious omission of these models is modelling of the environment in which the sensornet resides, including the physical phenomena to be measured by sensor nodes and other factors which are not to be measured but will nevertheless influence the operation of the network [7, 174].

Shenker models sensornet data as [257]:

- *Observations*: Single, raw, low-level sensor readings.
- *Events*: Constellations of related *observations* of defined *event type.*
- *Web of events*: Hierarchy of events, some defined in terms of other events.
- *Event notifications*: Messages describing *events.* omitting raw *observations.*

- *Tasks*: Instructions to nodes describing required data gathering.
- *Actions*: Activities undertaken by nodes upon detecting *events*.
- *Queries*: User or application requests to elicit *event* information.

Nested queries allow nodes receiving *primary queries* to issue *secondary queries*, allowing greater abstraction in end-user queries [75, 125].

2.4.3 Simulation

Simulation offers an environment in which experiments are perfectly repeatable and perfectly controllable. This is generally impossible in real world experiments, and is particularly acute in sensornet research where experiments must consider interaction with non-network entities. Anastasi *et al.* [12] state that *simulation is clearly a better choice than experiments when considering large networks, as controlling large testbeds is very hard*, and also state that *most of the research on ad hoc networks has been carried out by adopting simulation and analytical approaches only.* It should also be noted that the publications in which the *TTL Bounded Gossip* (TBG) [217] and *Implicit Geographic Forwarding* (IGF) [120] protocols are introduced and described rely heavily on simulation. It is therefore appropriate to use simulation in analysing and comparing these protocols, as implemented in chapters 3 and 4.

Investigators must determine acceptable accuracy-scalability tradeoffs [243]; simulation-derived results are meaningless if simulated behaviour does not sufficiently match real behaviour [232] and are particularly sensitive to timing discrepancies [172]. Wireless communication models are usually the component with highest computational cost [212] but represent the greatest source of inaccuracy [162].

Discrete event simulators are well suited to computer network simulation [20]. *Simulation models* [21] are constructed, similar to those used in model checking [268], and executed in *simulation engines* [109]. Incorporating real application code, execution environments, hardware, network connections, or other real entities, into *simulation models* yields *emulation models* [109], improving accuracy but harming scalability [308]. Real and simulated entities interact directly in *hybrid simulations* [175].

Numerous sensornet-relevant simulators and emulators exist. Unfortunately, no current examples offer total accuracy or reach desired scalability. *TOSSIM* offers cycle-accurate low-level emulation of Berkeley motes running TinyOS but very simplistic net-

work modelling [179]. More detailed modelling might be required where observable phenomena are very sensitive to minor variation in network conditions [162].

ns-2, the predominant network simulator in sensornet research [40, 179], uses highly-detailed network models [40] but is single-threaded [128] and scales only to around 100 simulated nodes [154,212]. Much of the popularity of *ns-2* can be attributed to the breadth of reusable libraries and protocol models developed by numerous researchers. However, the complexity stemming from this lack of focus and the underlying architecture can make working with or extending *ns-2* time-consuming and difficult [128]. *ns-2* was not originally designed for wireless network simulation, support for which must be added through extensions [40], and achieving highly realistic simulation results may require careful tuning of simulation models to a specific physical environment [142].

J-Sim [267] offers similar facilities to *ns-2*, also providing a component model which can be scripted and customised through the *Tcl* language. *J-Sim* is less widely used than *ns-2* but better suited to sensornet simulation as it was designed for this purpose, and has more detailed support for modelling physical environments and network-environment interaction. It also offers limited multithreading support within a single processing host.

The high computation cost of network simulation might be addressed by task parallelisation. Interentity communication across parallel simulation hosts [246] may negate some benefit of additional processors by Amdahl's Law [11, 302]. However, this is not to say that parallel processing has no role to play, merely that care must be taken to ensure that simulator designs work harmoniously with the characteristics of a given target parallel processing environment.

GloMoSim exploits parallel execution by multithreaded simulation, scaling to 10000 simulated nodes across 10 processors [340]. However, to achieve this scale *GloMoSim* consolidates many independent entities of the simulated system into single compound entities, necessarily sacrificing low-level accuracy for performance. However, as the number of processors increases, so does the overhead in dividing the problem and then recombining the separate elements. Interentity communication across parallel simulation hosts [246] negates the benefit of additional processors in network simulation [302] as a consequence of Amdahl's Law [11]. Simulating sensornet-scale networks of millions of nodes requires entity-concatenation [20] and layer-concatenation [40,241] to reduce the memory footprint [47], further sacrificing simulation accuracy.

Lehnert *et al.* propose a three-tier development strategy composed of simulation, emu-

lation and deployment on real mobile devices [175]. Evaluating self-organising applications for multihop ad-hoc networks may require more mobile devices than are available, or be impractical. Rather than attempt to build these large testbeds, a hybrid approach is proposed. Initial evaluation is performed by *pure simulation*. When the design is thought sufficiently stable, evaluation moves to *hybrid simulation* mixing cooperating real and simulated devices, and thence to a *pure hardware* stage for final design evaluation. A consistent programming interface and design model is maintained throughout these stages.

Although useful, simulation is not perfect. Simulation remains useful, however, where other approaches are impractical. The models of systems are abstractions of the real systems; inaccurate or incomplete models may give misleading results. Higher fidelity modelling and simulation tends to imply reduced simulation speed [147], implying a tradeoff between acceptable accuracy and performance. Furthermore, each simulation instance represents a single path through state space. Even large numbers of simulation instances generally cannot guarantee to cover all valid paths, and therefore may not identify some behaviours. *Model checking* approaches, by contrast, can guarantee coverage of all valid paths through state space, and explicitly identify when defined correctness criteria are violated in sensornet systems [218].

2.4.4 Formal analysis

When exploring the behaviour of a system it is possible to apply either *analytical techniques* to a formal model of the system, or *simulative analysis* in which experiments are performed with a simulation model of the system, or some combination of the two. Hybrid approaches have been proposed which combine elements of both model checking and simulation, utilising a shared system model. However, this approach assumes that a reasonable formal model of the system can be constructed. There is some debate as to whether this is [136] or is not [242] currently feasible for typical sensornets.

The most common usage of model checking in sensornet research is to verify the correctness of protocols [25] rather than to evaluate protocol performance, or behaviour within complete network systems. However, model checking is prone to *state space explosion* problems which render the required computation effort infeasibly large for problems of realistic scale. For example, a worst-case network lifetime evaluation considered networks of between 4-11 nodes [207], which are orders of magnitude smaller than systems containing thousands to millions of nodes expected of sensornets in the foreseeable future [7].

Ploennigs *et al.* [233] propose a hybrid approach in which a formal model of the system is defined for use in *analytical techniques* and from which simulation models are automatically generated for *simulative analysis*. Acknowledging that setting up the formal models is often difficult and time-consuming, an automated network model building approach is defined which exploits information from existing design tool databases which specify an abstract functional view. However, this approach implicitly assumes that these design tool databases contain the required information, which is not necessarily true for all design tools. If this assumption does not hold, the effort expended in adding this information to the design database may not be less than that required to construct the models by hand.

Sobeih *et al.* [268] extended the *J-Sim* simulator [267] by integrating a model checking framework, but without removing the existing simulation capability. Protocol-specific abstractions were implemented which significantly reduced the size of the state space in the protocol under consideration. Although these protocol-specific optimisations were successful in reducing execution time to an acceptable level, they lack generality and must be designed and implemented for each protocol. Furthermore, although the extended *J-Sim* enables reuse of simulation models and simulator code, there is no real integration between simulation and model checking; the model is evaluated by one method or the other.

A prototyping framework, based on the *Prototype Verification System* (PVS) theorem prover, aims to assist protocol designers [30]. An iterative prototyping phase is used to gain informal confidence in a protocol. A formal protocol definition is obtained by refining a set of general formal PVS models, reusing an extensible set of executable communications primitives. Simulation code is automatically generated from the formal definition. Executing this simulation code may or may not reveal defects in the protocol. The observation of incorrect behaviour indicates the existence of defects; the formal definition is amended, and the simulation repeated. However, the absence of observed incorrect behaviour does not prove that no defect exists. When simulation reveals no further defects, the correctness properties are checked by analysing the formal definition with a theorem prover. Again, there is no real integration between simulation and model checking beyond the reuse of the system model. However, unlike the extended *J-Sim* [268] this is a formal model and hence is better suited to describing formal properties.

2.5 Routing protocols

Routing is the set of activities through which paths are obtained through networks for the flow of data between producers and consumers. Traditional data routing mechanisms ignore the content of data, blindly forwarding packets from node to node. Although necessary for general-purpose networks, application-specific networks such as sensornets can exploit awareness of data flow content to enable various optimisations and cost reduction strategies.

2.5.1 Routing packets in sensornets

Layered network models such as OSI [343] and TCP/IP [91] abstract the routing mechanism from other network elements. In small fully connected networks, it may be appropriate to manage all routing activity at the *Data Link Layer*. In larger, more complex networks, the routing mechanism operates at the *Network Layer* to discover, and optionally maintain, multi-hop routes between sources and sinks. Scalability is a major concern, particularly for large networks; the algorithmic and storage costs often grow as the square (or worse) of the number of nodes [231].

Karp and Kung [156] note that the dominant factors in the scaling of a routing algorithm are the rate of change of the topology, and the number of routers in the routing domain. Sensornets typically contain large numbers of nodes, each of which is a router, assembled into an ad-hoc network with rapidly changing topology. Scalability is therefore likely to be a significant factor in the design and evaluation of a suitable routing algorithm. Most routing approaches for ad-hoc networks assume that the rate of topology change is *not high enough to make flooding the only alternative and not low enough to make the traditional routing algorithms effective* [80].

Protocols found to work well in traditional wired networks may not work equally well when transferred to a wireless network. For example, Biaz *et al.* [32] evaluated the performance of TCP in wireless ad-hoc networks, finding that TCP's inability to distinguish packet loss from link failures from losses caused by congestion, resulting in low network utilisation caused by transmission rate reductions in the absence of congestion. Observed network throughput fell to around 40% of expected throughput when a stable network topology was transformed into an unstable network topology by allowing nodes to move.

Krishnamachari defines *address-centric* and *data-centric* routing, and discuss how the

application designer selects the most appropriate [163]:

- *Address-centric routing*: Each source independently sends data along the shortest path based on the route that the queries took ("end-to-end routing")
- *Data-centric routing*: Sources send data to the sink, but routing nodes en-route look at the content of the data and perform some form of aggregation or consolidation function on data originating at multiple sources.

Krishnamachari [163] considers *data-centric routing* to be synonymous with *aggregation*, although it could be argued that it is not necessary to aggregate or otherwise modify data in transit in order to make routing decisions based on the content of the data. Traditional network protocols generally ignore packet payload, utilising only metadata contained in packet headers in routing decisions [91]. Data packets generated by lower network stack layers may be wrapped as payload within higher layer packets. Under this model, network routers never consider packet payload in routing decision-making, passively forwarding data through the network unexamined and unmodified.

Content-based routing [46] offers another approach to routing data within networks. The fundamental concept of *content-based routing* is different to that of *address-centric routing*, in which routing decisions are obtained by applying a set of rules to logical or physical addressing metadata, obtained explicitly from packet headers or implicitly from packet broadcast details. Under the *content-based routing* paradigm, routing decisions are obtained by applying a set of rules to the data payload of each packet; no metadata is used. The ultimate destination of message packets is unknown and undefined at the producer, and is instead determined by potential consumers expressing an interest in receiving messages fulfilling some specified criteria.

There are some similarities between *content-based routing* and *data-centric routing* as described by Krishnamachari [163]. The network paths followed by packets are implied by packet content. An application-specific association exists between the producers of a given type of labelled data, and the consumers which express an interest in receiving data of this type. However, there also exist important differences. *Content-based routing* approaches do not attempt to aggregate multiple data streams derived from multiple sources; each packet is routed individually toward potential consumers. The content of packets is inspected, but not modified, by the routing mechanism. Whereas *data-centric routing* mechanisms generally connect multiple data sources to a single data sink [140],

a multicast *content-based routing* approach generally connects a single data source to multiple data sinks [45].

The *Context and Content-Based Routing protocol* (CCBR) [66] is a *content-based routing* protocol for sensornets. A receiver-based approach to routing is implemented. Upon receiving a packet, each candidate relay node decides autonomously whether to forward this packet to one or more neighbours. The decision is independent of the packet sender, using a probabilistic scheme to limit redundant transmissions by favouring retransmission by nodes located further from the previous relay.

Each node collaborates loosely with its neighbours to distribute and maintain knowledge of the interests and locations of data sinks within the network. This collaboration with neighbours also enables the *context-aware* aspect of CCBR. Rather than require data interests to be expressed in absolute terms, data sinks can instead express interests in data which exhibit some property relative to other related data [66]. For example, a sensornet user may be interested in identifying geographic regions that are warmer than their surroundings, but without knowledge of the average temperature.

A key advantage of CCBR [66] over protocols such as *Directed Diffusion* [140] is that node mobility and population dynamics are addressed specifically in protocol design. Changes in network composition and data sink interests are reflected immediately in node behaviour, without the overhead of route repair mechanisms or the requirement to update all nodes participating in a data flow.

Active network routers examine and potentially modify passing data flows, thereby implementing part of the application [298]. All sensornet nodes are routers sharing responsibility for distributed application processing, so sensornets are *active networks*. *Active networks* may separate the distribution of data from the distribution of processing instructions to routers, or combine these with hybrid data-instruction packets called *capsules*. Router instructions may modify packet payload data, or simply instruct routers how to interpret payload data in routing decisions. For example, specific routers could be instructed to apply matching rules to packet payloads, determining next-hop by which rules hold true without changing packet payloads. Alternatively, more complex instructions could implement in-network processing of various types such as Krishnamachari's proposed aggregation [163].

Krishnamachari [163] considers three data production scenarios:

1. *All sources send completely different information (no redundancy)*: Both address-

centric and data-centric algorithms will incur the same number of transmissions; no aggregation is possible.

2. *All sources send identical information (complete redundancy)*: Address-centric routing can be more efficient than data-centric routing, if the sink observes that duplicate information is arriving and issues an instruction to all but one of the sources to cease. However, this assumes much about the network and the application.

3. *Sources send information with some intermediate, non-deterministic, level of redundancy*: Address-centric routing can perform no better than data-centric routing, which reduces the overall number of transmissions by aggregating data.

It is evident that the choice of *address-centric* or *data-centric* routing is dependent on the nature of the network application; there is no single best choice for all scenarios. However, Krishnamachari *et al.* note that *data aggregation*, and hence *data-centric routing*, is a *particularly useful paradigm for wireless routing in sensor networks* as this can eliminate redundancy and minimise transmissions, thereby conserving precious energy resources.

2.5.2 Sensornet routing protocols

Virtually any computer network protocol could be implemented in a sensornet. However, owing to the characteristics of sensornets, some protocols are better suited than others to this environment. Lightweight protocols are generally favoured as small, low-cost mote hardware platforms have extreme resource constraints. Motes may not have globally unique identifiers [216], and it is reasonable to expect that the sensornet continues to function adequately following the failure of any individual mote. It follows that stateless protocols are better able to cope with an unreliable mote population, and low-complexity protocols may consume fewer resources. Advantageously, this also makes it more feasible for experimenters to gain an understanding of protocol behaviour in a given sensornet.

Tilak *et al.* [300] observe that, although sensornets are ad hoc networks, generic ad hoc routing protocols will *generally not be good candidates* for selection in this context because they commonly ignore power conservation, ignore routing table growth as networks grow, focus on end-to-end communications, assume globally unique node identification, and lack support for cooperative information dissemination required for in-network processing.

Stateful routing protocols require nodes to maintain at least partial knowledge of the network topology upon which to base routing decisions, whereas *stateless* protocols do

not. Maintaining routing knowledge incurs substantial expense in signalling traffic, power consumption and storage at each node [249]. This overhead may be justified by reduced latency as protocols can immediately make appropriate routing decisions. Simple *flooding*-derived protocols are stateless as decisions are independent of network topology [116]. Many *geographic* protocols are stateless as routing decisions are based on spatial, rather than logical, network structure, though some state information may be used to work around network voids [120]. *Proactive* protocols are stateful, and maintain partial network topology knowledge in advance of need. *Reactive* protocols are also stateful, but acquire network topology knowledge as and when required, perhaps caching this for future use.

The mechanism for routing information exchange is orthogonal to the subsequent use of this information in routing decisions. Most packet-centric routing algorithms with stateful nodes use either the *Distance-Vector* or *Link-State* approach [224]. *Distance-Vector* algorithms require nodes to maintain routing tables containing all known destinations, and for each destination a set of candidate routes. Each route is defined as a tuple of next hop node and associated cost. Nodes periodically broadcast their complete or partial routing tables to neighbours, which merge this knowledge with their own. Cost is often measured in *hop count*, though other options include *bandwidth*, *load delay* and *reliability* [79]. *Link-State* algorithms store considerably more information in routing tables. Each node periodically floods *link state advertisements* throughout the network, each describing possible connectivity between the originating node and its neighbours. Recipients merge incoming knowledge with their own *link state database*.

Both Distance-Vector and Link-State algorithms can create routing loops. Link-State routing loops are short-lived and disappear within the time for packets to traverse the network diameter [224], but Distance-Vector routing loops may be long-lived. Various techniques exist to reduce Distance-Vector routing loops [79], such as *split horizon* suppressing nodes advertising routes back to neighbours from which they learned the route, and *poisoned reverse* to advertise such *reverse routes* with an unreachable infinite cost.

These approaches generally attempt to find low-cost routes using links of fixed cost, rather than attacking the link cost itself. *Minimum transmitted energy* protocols select least-cost routes using the average energy consumed in transmitting packets between node pairs as the link cost [51]. *Maximum lifetime energy routing* protocols extend *minimum transmitted energy* protocols by introducing remaining node energy to the link cost function [51]. *Maximum lifetime data gathering* protocols implement energy load-balancing

across nodes [151]. *Energy-Aware QoS Routing Protocol* finds least-cost paths conforming to given end-to-end latency requirements [5].

A rich and diverse set of sensornet routing protocols have been proposed in the literature and implemented in industry. It is impractical to assess each extant protocol as too many exist; a comprehensive survey can be found in [7, 155, 336]. We therefore consider representative examples from the major classes of sensornet routing protocol.

2.5.2.1 Packet centric routing by network flooding

In classic flooding, a node wishing to send a packet broadcasts into the shared wireless medium. Each neighbour receives a copy, and then rebroadcasts the packet to its neighbours unless it has previously broadcast the same packet. Flooding converges when each node has received the packet, which occurs in $O(d)$ rounds in network diameter d [165]. Unbounded flooded messages can easily cover the entire network [116] which is wasteful if the source and destination are physically close.

Flooding is utilised by most non-geographical routing protocols [116]. Where the topology of a network changes very rapidly, it may not be possible for more sophisticated routing algorithms to react quickly enough to maintain up-to-date routing information. In these situations, flooding may be the only feasible routing strategy [148, 224].

Under ideal conditions, a flooded packet would ripple outward from the source in an orderly, uniform circle until all nodes had received at least one copy. However, despite its *appealing simplicity* [165], counterintuitively complex behaviour is frequently observed [102] owing to coupling and interactions between protocol stack layers and components. In particular, the Physical and Data Link layers induce non-uniform unpredictable behaviour which may leave some nodes or regions untouched. *Broadcast storms* [217] are particularly problematic with significant redundant broadcasts, contention, collisions, and high energy consumption, and *may result in significant power consumption and possibly a network meltdown* [121].

In experiments in a 185-node network [102], 5-15% of links were asymmetric with the proportion growing significantly with increasing distance at low power levels. Four types of unexpected behaviour were observed:

Straggler : Node that misses a transmission, even though it would be expected to receive a packet with high probability.

Backward Link : Link in which the recipient of the flood is closer to the base station than the transmitter.

Long Link : Link that is significantly longer than expected at given transmit power level.

Clustering : Number of nodes attached to a single point on the data gathering tree.

Counterintuitively, increasing radio transmit power does not necessarily improve performance [102]. With increasing power, the time taken to cover a given network proportion tends to decrease, but the time taken for the final 5% of nodes to receive the message was equal to the time for the preceeding 95%. Surprisingly, increasing transmit power increases the number of hidden terminals. Collisions are increased at the beginning of the flood, leaving pockets of *stragglers* and hence a larger number of backward links generated as the flood "rebounds". It follows that the simplistic strategy of increasing transmit power does not work, suggesting more sophisticated approaches are required.

Ni *et al.* [217] suggest that network composition and configuration influences the frequency and intensity of these effects, and thus offers a means of limited control. For example, multiple rebroadcasts of packets can be employed, with each rebroadcast increasing coverage but by rapidly diminishing amounts. An initial rebroadcast will yield anywhere between 0-61% increased coverage, falling to a maximum of 41% in the next rebroadcast. When the number of rebroadcasts is greater than 4, additional coverage will be below 0.05% in each case and additional broadcasts are hence of little utility. Contention grows quickly with host density. If multiple nodes have overlapping wireless communications coverage, all will attempt to begin rebroadcasting upon receiving a packet simultaneously, often resulting in all experiencing contention. Where two nodes attempt to rebroadcast, the probability of contention is approximately 59%. The probability is over 80% with 6 or more nodes with overlapping communications coverage.

GOSSIP protocols extend flooding by implementing probabilistic rebroadcast [217]. Upon receiving a packet, each node independently decides whether it will be rebroadcast with probability p, or silently dropped with probability $1 - p$ [253]. Only probabilistic performance guarantees and analyses are possible; unless $p = 0$ or $p = 1$, in a perfect network, no deterministic guarantees of delivery are possible. For $p \in [0, 1], p \neq 0, p \neq 1$, gossiping displays bimodal behaviour, in that *in almost all executions of the algorithm either hardly any nodes receive the message, or most of them do* [116]; values of $p \in [0.6, 0.8]$ often, but not always, ensures most nodes receive most packets. Appropriate

gossip probability selection is generally difficult, and may need to vary across nodes and time [169].

The existence of this *phase transition phenomenon* is contested by Sasson *et al.* [253], who suggest that bimodal behaviour as predicted by percolation theory is not observed in real networks in which packet collisions, packet loss and contention can occur. Instead, the success rate curve tends to become linear for networks of low average *node degree*, the number of possible communication partners, and tends to resemble a bell curve for networks of high average node degree.

Any protocol based on or utilising flooding can be modified to instead use a gossiping approach. Haas *et al.* [116] replaced flooding with gossiping in AODV and ZRP to yield AODV+G and ZRP+G. With network simulations of networks as small as 150 nodes, message traffic was reduced by up to 35% in AODV+G. No comparable figures were supplied for ZRP+G, but it is stated that this showed *significant improvement in all performance metrics*. On the other hand, the routes produced by the gossiping variants may be longer than those produced by the standard flooding variants. Also, probabilistic gossiping is non-deterministic; for example, arbitrary repeats of route discovery attempts may be required if probabilistic flooding fails to include the destination.

Epidemic Routing, as implemented by the *Epidemic Routing Protocol* (ERP) [307], extends flooding. Nodes have a bounded packet buffer; for a given packet, any node possessing a copy is a *carrier*. When a communication channel between any pair of nodes becomes available, they exchange one or more randomly selected packets, perhaps dropping existing carried packets to make space. Eventually, the packet may reach its destination. ERP does not assume a connected path ever exists between packet source and sink, and carriers of packets tend to remain as such for some time, allowing successful delivery where temporary network partitions occur. However, packets may be lost if buffers are not sufficiently large to contain all packets in transit at any given time, which is unrealistic for resource-constrained sensornets.

Other flooding variants include counter-bounded, distance-based and location-based types [217]. Energy-aware gossiping variants exist, which turn nodes off at random [133] to exploit the fact that overhearing irrelevant communications is a major source of energy consumption [265]. The physical topology of the sensornet strongly influences energy consumption [339], as exploited by the *Smart GOSSIP* variant [169].

Parameter	DSDV	CGSR	WRP
Time complexity (link addition/failure)	O(d)	O(d)	O(h)
Communication complexity (link addition/failure)	O($x = N$)	O($x = N$)	O($x = N$)
Loop-free	Yes	Yes	Yes (not instantaneous)
Number of tables per node	2	2	4
Frequency of update transmissions	Periodically and as needed	Periodically	Periodically and as needed
Updates transmitted to	Neighbours	Neighbours and cluster head	Neighbours
Routing metric	Shortest path	Shortest path	Shortest path
Critical nodes	No	Yes (cluster head)	No
Multicast support	No	No	No

Table 2.1: Complexity characteristics of proactive routing protocols

2.5.2.2 Table-driven proactive routing

Dynamic destination-Sequenced Distance Vector (DSDV) [229] is among the earliest and most widely referenced proactive routing protocols for ad-hoc networks. DSDV is based on the *Distributed Bellman-Ford* (DBF) algorithm [209] and attempts to find the shortest path, defined as smallest number of hops, through a graph representing a network. DSDV convergence is slow [209]; sequence numbers attached to routing table broadcasts reduce rather than remove the risk of stale routing data propagation. Routing table flux is damped by delaying advertisement of new routes, reducing convergence time [224].

Unlike many protocols, DSDV can work with layer 2 (Data Link Layer) or layer 3 (Network Layer) addresses [224]. For sensornets with non-hierarchical address structures this could be advantageous, as routing could be tightly coupled with Data Link Layer functions for efficiency, albeit at the expense of reducing protocol modularity. However, flat addressing schemes lead to substantial routing table overhead communicating topology updates, growing as O(n^2) in node count [230].

Royer *et al.* [249] compare the complexity characteristics of several proactive routing protocols in table 2.1, where d=network diameter, h=height of routing tree, x=nodes affected by topology change, and N=number of nodes in network. The protocols considered are DSDV [229], CGSR [53], and WRP [209].

From table 2.1 neither DSDV, CGSR or WRP is optimal in all criteria. CGSR extends DSDV and is therefore similar in some aspects. As time and communication complexity is similar for each, protocol selection must consider other factors. WRP is not immediately loop-free unlike DSDV/CGSR, but avoids the *counting-to-infinity* problem [91] in which two nodes repeatedly exchange route information, increasing the metric with each itera-

tion. WRP requires 4 routing tables per node whereas DSDV and CGSR require only 2, of smaller total size, suggesting a poor fit for resource-constrained sensornets, but routing table broadcasts contain only topology changes rather than complete tables and are thus smaller.

WRP is superior to DSDV/CGSR if energy and bandwidth minimisation is more important than per-node storage cost minimisation; in sensornets, the former is more likely than the latter. Selecting between DSDV and CGSR depends on network size; CGSR scales better in large networks, but energy cost is uneven across the node population. Poor scalability of DSDV, with periodic update overhead growing as $O(n^2)$ in network size, may preclude large networks functioning at all, whereas CGSR may work at the expense of some nodes failing before others.

2.5.2.3 On-demand reactive routing

Ad hoc On-Demand Distance Vector (AODV) routing is a *pure on-demand route acquisition* algorithm; nodes that do not lie on active paths neither maintain any routing information nor participate in periodic routing table exchanges [230]. Route acquisition consists of several stages:

1. Reverse path setup. A route request is flooded through the network, restricted by a *expanding ring* specified by packet TTL to minimise the flooding scope. Each node records the address of the first neighbour from which the route request was received, implicitly setting up the reverse path. Each node maintains a routing table for destinations of current routes, the entries of which timeout if not used.

2. Forward path setup. A route reply is forwarded from the destination to the source by unicast, establishing the forward path. This route reply does not contain the route itself; this information is distributed among the nodes, and the reply just indicates that the route exists.

3. Path maintenance. If a broken link is encountered in routing, the source is informed and must reinitiate the route discovery process.

AODV always favours newer routes over older routes, even if older routes are shorter [70], potentially inducing suboptimal route selection. In simulation it was found that performance, measured as packet delivery success proportion and latency, decreases substantially as the network grows. This effect became noticeable between the 100- and 500-node

Parameter	AODV	DSR	TORA	ABR	SSA
Time complexity (initialisation)	O($2d$)	O($2d$)	O($2d$)	O($d+z$)	O($d+z$)
Time complexity (postfailure)	O($2d$)	O($2d$)	O($2d$)	O($l+z$)	O($l+z$)
Communication complexity (initialisation)	O($2N$)	O($2N$)	O($2N$)	O($N+y$)	O($N+y$)
Communication complexity (postfailure)	O($2N$)	O($2N$)	O($2x$)	O($x+y$)	O($x+y$)
Periodic beacons	No	No	No	Yes	Yes
Multiple routes	No	Yes	Yes	No	No
Node route maintenance	Route table	Route cache	Route table	Route table	Route table
Routing metric	Freshest, shortest path	Shortest path	Shortest path	Associativity, shortest path	Associativity, stability
Multicast support	Yes	No	No	No	No

Table 2.2: Complexity characteristics of reactive routing protocols

sizes [230], primarily because of a greatly increased level of collisions. These collisions were a result of more nodes per unit area, a greater number of protocol control messages, and longer paths causing a greater likelihood for collisions during the hop-by-hop forwarding.

Royer *et al.* [249] compare the complexity characteristics of several reactive routing protocols in table 2.2, where d=network diameter, l=diameter of network segment, y=total number of nodes in directed path, and z=diameter of directed path. The protocols considered are AODV [230], DSR [148], TORA [224], ABR [303], and SSA [80].

Examining time and communication complexity details in table 2.2, two groups of protocols emerge; the *associativity-based* protocols ABR and SSA, and the *associativity-ignoring* protocols AODV, DSR and TORA.

Within the *associativity-based* group, the main difference is that SSA considers link stability and ABR does not. Links in sensornets are unlikely to be stable owing to unpredictable wireless environments, unreliable nodes, and power management policies switching off communication modules, suggesting that SSA is unlikely to perform well.

Within the *associativity-ignoring* protocol group, only AODV supports multicast; if required, AODV is the automatic choice. DSR broadcasts full routing tables whereas AODV distributes this information among neighbouring nodes; DSR may have greater routing overhead and scale less effectively in large networks than AODV. However, DSR can operate with unidirectional links commonly found in sensornets, whereas AODV cannot. TORA and DSR support multiple routes, unlike AODV, but there are few other reasons to select TORA over AODV.

2.5.2.4 Geographical routing

Geographical routing protocols exploit the *physical* network topology, rather than the *logical* network topology employed by non-geographical protocols. The physical distribution of nodes, and their communication capabilities, implies the potential connectivity of each node to neighbouring nodes. It follows that the set of nodes with which a given node can communicate is a superset of those nodes with which communication is permitted under any possible logical network structure overlaid on the physical network. Some protocols require nodes to know their absolute location, whereas others require only that nodes know the relative location of their neighbours. Knowledge of physical location is not unreasonable in sensornets, in which physical location data must be known in any case to label the data collected from the physical environment.

In the *Greedy Perimeter Stateless Routing* (GPSR) protocol [156], routing decisions compare the geographic position of routers against the stated geographic position of packet sinks. In addition to knowing its own location, each node learns the location of each neighbouring node but does not distribute this information throughout the network. GPSR scales better than non-geographical proactive and reactive protocols, as routing table size is independent of total network size and depends only on local node density.

To determine the most appropriate next hop for a given packet, GPSR consults its table of neighbour locations. A *greedy* routing decision is made in the first instance. The locally optimal choice of next hop is the neighbour geographically closest to the packet's destination. Nodes need only know their approximate geographical location, but more accurate knowledge tends to give better routing decisions in dense networks.

In some network topologies there exist regions in which greedy forwarding is impossible; packets must temporarily move *away* from the destination to circumnavigate *voids* before heading *toward* the destination again. Where packets arrive at non-destination nodes with no neighbouring nodes closer to the destination, *greedy mode* is swapped for *perimeter mode*, applying the *right-hand rule* to traverse the graph. Routing switches back to *greedy mode* as soon as possible, when the packet is geographically closer to the destination than the node where *greedy mode* initially failed.

The *Implicit Geographic Forwarding* (IGF) protocol [120] also selects from the set of next hop relay candidates by considering geographical information. However, the next hop is selected by comparing *angles* rather than *distances*. This is advantageous if nodes are not equipped with accurate positioning equipment, such as *Global Positioning System*

(GPS) hardware. Estimating neighbour *distance*, perhaps by measuring received signal strength, may be inaccurate and prone to error from a changing radio environment. Estimating neighbour *angle* can be more accurate, either by scanning with steerable directional antennas, or by comparing relative signal strength across an array of directional antennas.

A three-phase handshaking mechanism is employed in relay selection. Assume packet p currently resides at node N but is destined for node D. N broadcasts a short *Request To Send* (RTS) packet. All m neighbours $M_i = M_1 \dots M_m$ in communication range of N receive the RTS, and calculate the angle $\theta_i = \angle M_i N D$. If $\theta_i \leq \theta_t$, a threshold angle to restrict the set of relay candidates, node M_i transmits a short *Clear To Send* (CTS) packet containing θ_i. N waits for some time q to collect θ_i values, after which the node M_i with the lowest θ_i value is selected as the relay. N transmits packet p in time r, with the identity of the selected relay M_i in the header. Nodes for which $\theta_i > \theta_t$ can switch off radios to conserve energy for time $q + r$ after calculating θ_i. Nodes for which $\theta_i \leq \theta_t$ listen for the transmission of packet p; all such nodes not selected as the relay can switch off radios to conserve energy for time r. Only the single selected relay must remain active to receive data while N transmits the complete packet p.

As network node density increases, the shortest possible network path between any two points is an increasingly good approximation to a straight line between those points, the latter being the optimal solution [88]. It is assumed that short, straight paths are desirable, as they imply fewer hops; this tends to deliver packets with reduced latency, energy cost, and wireless medium utilisation [120]. IGF employs a similar void circumnavigation mechanism to that used by GPSR.

The locally optimal greedy relay selection is not guaranteed to be globally optimal. Assume a network of reasonable density such that at least two candidate relays X and Y exist. Assume $|NY| < |NX|$ such that X is near the maximum communication range of N and Y is very close to N, but $\angle XND > \angle YND$. IGF will always select Y as the next hop relay, whereas X is the globally superior choice as necessarily $|DX| < |DY|$. This can lead to suboptimal route selection with a greater than necessary number of hops. However, in a network of near-uniform density, it is unlikely that this locally greedy approach produces routes that are significantly worse on average than a hypothetical globally optimal approach.

Geographical Adaptive Fidelity (GAF) [331] is an *adaptive fidelity* protocol rather than a routing protocol. It exploits geographic information to identify sets of nodes which are

near-equivalent from a routing perspective. Routing redundancy is correlated with denser node deployment regions in which many node-node communication pairings are possible. All but the minimum number required to maintain a constant level of *routing fidelity* are switched off, reducing energy consumption without harming connectivity or performance. Modified AODV and DSR show 40-60% energy consumption reduction in trials.

2.5.2.5 Real-time routing

Sensornets are inherently real-time systems owing to necessary interaction with their physical environment through sensors and actuators. Hard real-time guarantees are difficult to achieve in the context of unpredictable wireless communication environments and nodes based on unreliable hardware platforms. However, soft real-time guarantees are achievable. He *et al.* [121] opine that *end-to-end deadline miss ratio* is the most important metric for sensornets with soft real-time requirements, and that routing protocols using only local network information without flooding perform better in this metric as networks become more congested.

The *SPEED* protocol [121] provides per-hop delay guarantees by applying a distributed feedback control scheme within a geographic routing strategy. It supports three types of real-time communication services; real-time unicast, real-time area-multicast and real-time area-anycast. End-to-end delay is proportional to the source-destination physical distance. Each node requires knowledge of only immediate neighbours, minimising storage overhead. SPEED requires periodic beacons, with associated energy cost and network overhead, containing geographic location and *receive delay* as the round-trip time for a given neighbour to acknowledge previous beacons.

SPEED [121] is a stateless non-deterministic geographic forwarding algorithm. Packet relays are selected using only local information. Packets are dropped only when no downstream node can support the single-hop delay guarantee. The *forwarding candidate set*, X, of neighbours closer to the destination is established; the packet is dropped if $X = \oslash$. Otherwise, X is divided into Y for nodes with *send delay* $< D$ and Z for nodes with *send delay* $> D$ where D is the guaranteed single hop delay. If $Y \neq \oslash$ the relay is selected from Y by a probabilistic scheme where candidates with greater *speed* are more likely to be selected, where *speed* is the distance to that node divided by the *send delay*, analogous to a packet travelling at this physical speed. If $Y = \oslash$, back-pressure is applied to the upstream node; the packet may be dropped or forwarded to a node in Z according to a

different probabilistic scheme.

2.5.3 The case for protocol tuning

Many communication protocols are described in the literature, many of which accept a range of general and/or protocol-specific parameters to fine-tune performance for particular use cases [155]. It appears that rather more research effort has been devoted to creating new protocols than optimising the performance of existing protocols. Although these new protocols are valuable contributions to the field, a sensornet designer must balance the demands of multiple performance objectives, and as such it is rarely appropriate to create a new protocol to address a single factor in isolation [83].

The application of search-based protocol tuning methods offers the sensornet designer an excellent tool with which to optimise network performance for a given usage context. It is surprising, therefore, that this approach has received such little attention in the literature to date. Possible explanations include the large number of controllable factors, measurable responses, and test cases required for meaningful coverage.

To maximise the overall effectiveness and efficiency of complex systems it is insufficient to consider each influential aspect in isolation. Owing to the nonlinear relationship between distance and received signal power in wireless communications [86] the total energy consumed by the network in delivering a given packet is sometimes lower if the route contains many short hops rather than few long hops, despite the higher number of participating nodes and transmissions [157].

Sensornet designers must identify the most significant factors to avoid being swamped by unnecessary detail. Unfortunately, even identifying the relative importance of factors and their interactions is rarely trivial [93]. Discovering the best values to assign to these factors and understanding their impact on network behaviour tradeoffs is harder still. Tunable parameters are often defined without clear default values and may be defined over an infinite range.

2.5.4 Approaches to protocol tuning

The sensornet protocol tuning problem is not simple or idealised; it is a real-world problem with multiple inputs, multiple outputs, and multiple objectives. Complex interrelationships between factors are generally unknown *a priori* and hence cannot not be targeted during experiment design.

When exploring the behaviour of a system it is possible to apply either *analytical techniques* to a formal model of the system, or *simulative analysis* in which experiments are performed with a simulation model of the system, or some combination of the two. Ploennigs *et al.* [233] support the latter hybrid approach in which a formal model of the system is defined for *analytical techniques* and from which simulation models are automatically generated for *simulative analysis*. However, this assumes that a reasonable formal model of the system can be constructed. There is some debate as to whether this is [136] or is not [242] feasible for typical sensornets.

Mohan *et al.* [202] observe that little work exists on evaluating and maximising end-to-end performance of large, dense sensornets. A number of candidate protocol stacks were measured for a fixed configuration of sensornet and application. Rather than considering several tunings of a single protocol this experimental work considered a number of protocols, but only one configuration of each. A significant diversity of network performance was observed, though it was not always possible to determine why a given protocol behaved as it did. A more general observation was that delivery success tended to be higher for shorter routes than longer routes. This suggests an upper bound on the size of networks that can achieve a given QoS with a given protocol tuning.

With many controlled factors and measured responses it is generally difficult to understand the resulting complex interrelationships. Totaro and Perkins [305] apply a *systematic statistical Design Of Experiments* approach to evaluate and model the complex tradeoffs in designing Mobile Ad-Hoc Networks (MANETs). The underlying Design Of Experiments approach is based on well-known experiment design theory, which is not specific to MANETs [22]. It follows that this approach could be reapplied in the domain of sensornets. However, as MANETs and sensornets are different classes of network (see section 2.1.4), it would be necessary to amend the system model appropriately. Totaro and Perkins consider the impact of controlling network composition with a fixed network application and environment in MANETs. In contrast, in the remainder of this thesis we assume a fixed network design and a distributed application built on a tunable networking infrastructure in sensornets. We assume no prior knowledge of physical topology, which is significant as this strongly influences energy consumption [339].

To achieve an appropriate QoS it is necessary to consider each non-functional requirement throughout the protocol stack [239], influencing the design and operation of sensornet applications, networking protocols, network topologies and network tasking. Grenier and

Navet [111] illustrate how MAC protocols at the Data Link Layer can be fine-tuned to achieve certain real-time requirements while conforming to other application-dependent criteria. A search space is defined using the definitions of MAC protocols, and knowledge of parametric values thought likely to be near to good solutions. The search space is then explored by exhaustive search, using simulation to evaluate individual configurations. In chapter 4 we address the tuning of routing protocols at the next level of the OSI protocol stack, the Network Layer [343]. A network designer would ideally tune all layers of the protocol stack simultaneously, but in practice combinatorial explosion issues are likely to render this infeasible. Therefore, tunings of any given protocol or network layer must be robust to change in other layers.

2.5.5 Multi-objective optimisation

A significant issue with multi-objective problems is the difficulty in determining a well-defined ordering of solution quality for a given set of candidate solutions [59]. In single-objective problems, the ordering of solution quality is given simply by the ordering of the associated fitness values. However, with multi-objective problems there are multiple fitness values to consider. A solution with every fitness value lower than another is clearly superior, and is said to *dominate* the other solution. However, solutions may be lower in one fitness value but higher in another; this kind of solution is termed *non-dominated* and is harder to classify. In place of a single optimum value, multi-objective problems generally have a *Pareto-optimal front* along which all solutions are mutually *non-dominated*, and all other valid solutions are dominated by the Pareto-optimal front members.

Combinatorial explosion of possible solutions within a multi-variable problem renders exhaustive search impossible. Stochastic search algorithms explore the solution space non-exhaustively in reasonable time. Multi-Objective Evolutionary Algorithms (MOEAs) are stochastic generational multi-objective search algorithms. Using the *survival of the fittest* concept, they seek Pareto-optimal fronts by evolving progressively better solutions based on the relative *fitness* of previous solutions. Numerous algorithms of this type have been proposed including NSGA-II [73], SPEA2 [345], PESA [61], PESA-II [60], IBEA [344] and Two-Archive [236]. Details of the state-of-the-art can be found in a book by Coello *et al.* [59] or a recent survey paper by Guliashki *et al.* [113].

Jourdan and de Weck [149] were the first to apply MOEAs to a sensornet optimisation problem, in which the optimal physical location of a set of motes was determined. Two

objectives were defined, *maximal coverage* and *maximal lifetime*, with no fixed constraints. Although these early results were encouraging, they were derived from unrealistically small networks of just 10 nodes.

Quintão *et al.* [238] addressed the optimisation of dynamically controlled sensornet topologies with the competing objectives of *minimal energy cost* and *maximal network coverage*. Two approaches were compared; a linear programming formulation which yielded exact solutions with high computation overhead, and an evolutionary approach which yielded good solutions in acceptable time. Greedy evolutionary algorithm approaches were effective in finding local minima, but were often ineffective in looking beyond these local minima to better solutions elsewhere in the parameter landscape. We address this problem in our work with a two-phase approach in which an initial principled search of the problem space is used to direct the evolutionary algorithm to regions likely to contain global rather than local minima.

Molina *et al.* [203] address the sensornet layout optimisation problem, balancing the two competing objectives of *minimal energy cost* and *maximal network lifetime* with a constraint of *complete network coverage*. Two MOEAs are employed for this problem; NSGA-II [73] and IBEA [344]. Each MOEA found sets of non-dominated feasible solutions that were found to be efficient, but no statistically significant difference was observed between the algorithms. Yang *et al.* [333] also applied the NSGA-II algorithm to sensornet design, but instead sought to find the optimal configuration of adaptive antennas. These papers demonstrate that the application of multi-objective evolutionary algorithms to the domain of sensornet optimisation is appropriate and feasible, but address optimisation problems unrelated to that considered later in chapter 4.

Alba *et al.* [8] describe a process to optimise MANET broadcast strategies. Five tunable parameters were defined which characterise the search space, and three metrics of network performance were defined which characterise the solution space. Two problem formulations were considered; the first optimised against three objectives, and the second optimised against two objectives with a user-supplied constraint on acceptable solutions. A cellular multi-objective genetic algorithm called *cMOGA* was employed to generate a Pareto front of good candidate solutions from which a human designer selects. Although the technique is effective, the Pareto front is *very reduced* in the presence of constraints, and there is little guidance for designers on manually selecting the single best solution.

Bonivento *et al.* [34] describe an integrated system level design process for sensornets

consisting of a semirandom communications protocol called SERAN, a network initialisation and maintenance procedure, and a mathematical model for optimising parameters of these system elements. The user specifies application requirements in terms of end-to-end delay, and packet loss probability, and hardware platform, and an appropriate set of protocol parameters is produced by the model. Interestingly, the mathematical model is able to accurately characterise performance without requiring extensive simulations, which is useful where running simulation suites is impractical owing to lengthy runtimes. However, this relies heavily on specific characteristics of the SERAN protocol and a simplistic network model, and cannot be easily extended to optimise arbitrary extant protocols.

2.6 Real-time behaviour

Real-time computing attempts to ensure that scheduled tasks begin and end at the required time in a reliable and predictable manner. Sensornets and sensornet-hosted services interact with their physical environment, both observing and influencing. End-users may depend on sensornets delivering information on the physical environment in a timely manner. The physical environment operates in real-time, therefore sensornets are necessarily real-time systems.

Real-time systems are those in which correctness *depends not only on the logical result of the computation, but also on the time at which the results are produced* [42]. In other words, the system must produce results at the *right time*; it is not sufficient to produce results *quickly* [43]. *Real-time jobs* are single units of work that become available for execution at a *release time* and must complete by the absolute *deadline* or, equivalently, within the relative *response time* from the *release time*. A sequence of related *jobs* constitute a *task* [185]. In a *periodic task* the jobs are released with regular *period*, whereas no such constraint applies to *aperiodic tasks*. *Sporadic tasks* are unlike *periodic tasks* in that their minimum *release times* and maximum *execution times* are unknown *a priori* [185].

2.6.1 Real-time sensornets

Real-time sensornet behaviour is considered by Stankovic [274], who observes that *sensor networks operate in the real world, hence timing constraints are important*, and that although some sensornets are not timing-sensitive *many sensor networks will have explicit real-time requirements related to the environment.* Some real-time requirements may ap-

ply within a network node, for example taking accelerometer readings at known intervals allows an estimate of speed to be obtained which will be incorrect if the intervals are inaccurate. Other real-time requirements will apply across multiple nodes, for example requirements to inform a base station of an observed event within a given period of the event occurring.

Stankovic observes that real-time guarantees are particularly difficult to achieve in sensornets [274] due to large scale, non-determinism, noise, and unreliability of nodes and networks. Attempts to support real-time guarantees in sensornets generally focus on scheduling the transmission of packets, and are discussed in [121, 121, 180, 188, 189] and build on previous work discussed in [168]. Liu *et al.* [187] discuss the use of imprecise computation to enhance dependability of real-time systems, such that imprecise results are accepted if precise results cannot be calculated within the deadline. If solution quality increases monotonically with respect to time the algorithm is allowed to iterate until it terminates or the deadline is reached. In a *data-centric* sensornet, it is also necessary to consider real-time scheduling of time-sensitive network traffic [2, 269].

Real-time constraints are of particular importance where sensornets deployed to gather data for safety-critical applications, or control applications where the sensornet manipulates the environment through a feedback loop of sensors and actuators. Online dynamic desynchronisation to schedule periodic actions of nearby nodes evenly throughout time, for example to share data sampling duties or schedule sleep cycles, can be implemented using the *DESYNC* method described by Degesys *et al.* [74].

2.6.2 Time granularity

Designing and evaluating a system whose correctness depends on time-sensitive behaviour requires an appropriate model of time, with sufficient detail and flexibility to encompass the necessary properties. Although the flow of time in the real world is, of course, completely independent of the properties of any given system, when reasoning about these systems we must select a time model which is amenable to analysis. This acknowledges the fact, first discussed by Newell [214] in the context of human cognition, that too fine or coarse a granularity of time measurement unit tends to obscure the issue under consideration.

For example, in the sensornet domain, *microseconds* may be an appropriate unit when measuring the execution time of CPU instructions. However, the *microsecond* would be inappropriate when measuring ambient temperature, as the the latter is unlikely to

vary more significantly than the physical sensor resolution over this timescale. Taking samples every *microsecond* would result in enormous volumes of sensor data which would overwhelm the storage, processing, and communication, capacity of motes, without offering any improvement in measurement accuracy. More importantly, these ambient temperature measurements would appear constant at this timescale, whereas at coarser timescales, such as the *second*, these measurements may vary. If the sensornet operator is interested in variation in ambient temperature within timescales of relevance to human experience and cognition [214], the *second* is meaningful whereas the *microsecond* is not.

It follows that a hierarchical time model is required to capture the requirements and behaviours of real-time systems for which a single abstraction of time would be suboptimal or inappropriate across the entire set of subsystems and concerns. Burns and Baxter [41] discuss the concept of *time bands*. The system time model is stratified into a hierarchy of time bands. Each band represents a different granularity of time measurement, from finest to coarsest granularity. A geometric progression of time unit size is observed as we ascend the hierarchy, where the measurement units of two adjacent bands differ in size by a factor of 10.

A well-defined system model is based on this hierarchy, such that different aspects of system behaviour are defined with appropriate time granularity. The model consists of the following elements [41]:

Bands A granularity of time measurement of relevance to the system.

Clocks Measures progress of time within a *band*, with each *tick* representing one discrete time unit of granularity appropriate to the *band*.

Activities An item of work undertaken by a system element, whose duration is an integral number of *ticks* within the single *band* within which it is defined and bound.

Events An *activity* with zero duration. Examples include *clock ticks*, and the *start-* and *end*-points of *activities*.

Precedence Relations Defines the order in which two *events* or *behaviours*, potentially within different *bands*, must occur.

Behaviours A set of *activities* and *events* within a single *band*, partially ordered by *precedence relations*, giving rise to parallel and serial composition.

Mappings A means of relating *behaviours* in one *bands* to those in another *band*.

Under the Burns-Hayes model, we reason about the time-sensitive behaviours of the system within each time band in isolation, perhaps applying different methods and criteria in each. We can then reason about the relationships between time bands, where the behaviour modelled within one time band potentially influences that modelled within another time band, as defined by the elements of a well-defined mapping rather than the low-level details [41]. This is conceptually similar to the Object Oriented Programming principle of *encapsulation*, which is *the process of compartmentalizing the elements of an abstraction that constitute its structure and behaviour; encapsulation serves to separate the contractual interface of an abstraction and its implementation* [35]. It is expected that imprecision may exist between bands [41], but whereas this might lead to imprecise behaviour it can also lend robustness to a system if the design affords a sufficient margin for error.

2.6.3 Synchronisation

Many sensornet tasks and data flows are at least approximately periodic [44], typically as a consequence of periodic interaction with the physical environment. It follows that sensornets require synchronisation mechanisms to ensure that these activities occur at the correct time at the correct nodes, and that tasks or sequences of actions spanning multiple nodes occur in the correct order. The network designer is responsible for exploiting the selected synchronisation primitive to achieve some desired behaviour.

A rich and diverse body of literature exists on the scheduling of periodic tasks in general systems; a comprehensive survey can be found in [185]. The periodic nature of sensornets suggests a *cyclic schedule* rather than a *priority-driven* or *deadline-driven* approach [42]. A *distributed* algorithm is necessary without a central controller to enforce shared schedules. *Dynamic* algorithms are required where motes are mobile or unreliable.

An alternative to online state scheduling protocols is to specify schedules at deployment time using traditional scheduling algorithms. However, sensornets do not have central controllers to enforce or arbitrate shared schedules, and algorithms intended for reliable environments may function poorly or may impose too much coordination overhead in unreliable sensornets of limited resources. *Hard real-time* requirements may be unsupportable. A comprehensive treatment of hard real-time scheduling algorithms and approaches can be found in [43]. *Hard real-time* requirements may be unsupportable in fundamentally unpredictable environments.

Caccamo *et al.* [44] propose a hybrid scheduling approach for multicellular sensornets. A *Frequency Division Multiplex* (FDM) strategy allocates different channels to adjacent cells by map colouring. Within each system-wide epoch an *Earliest Deadline First* (EDF) algorithm, distributed and replicated exactly at each node in a cell, allocates a proportion of equal-length frames to intra- and inter-cellular traffic. Traffic between adjacent cell pairs is managed under strict geographic *cyclic executive*.

A key weakness of this algorithm is its fragility. The EDF schedule must be replicated exactly at all nodes in a cell. If two adjacent cells contain different numbers of nodes, perhaps due to natural failure, their allocated inter-cellular frames will not overlap. The configuration requirement of one *router node* per cell creates single points of failure, and is perhaps an unrealistic requirement of sensornets deployed into unknown or hostile environments. However, it is based on well-understood and readily analysable scheduling algorithms; if the prerequisites can be met and nodes do not fail, it can support hard real-time application requirements. This property is comparatively rare in sensornet design.

PalChaudhuri *et al.* [221] define a protocol for clock synchronisation which is *adaptive* to the needs of a distributed application. It supports *relative synchronisation* where network nodes minimise the relative difference between local clocks, and *external synchronisation*. It can operate in single-hop or multi-hop networks. The overhead is relatively high; during each synchronisation iteration each node requires $O(n^2)$ bidirectional data packet exchange with all neighbours, and execution of a linear regression calculation. This cost is justified if the application requires nodes to collaborate at a *specific* time, rather than the lesser requirement that they collaborate at the *same* time.

Synchronisation of *chaotic* systems is non-trivial, but careful design may yield *self-synchronising* systems without need of external forced coordination [77]. Systems whose behaviour naturally converges on desired behaviour require less management, with consequent savings in energy, network bandwidth, and processing overhead. *Direct Sequence-Code Division Multiple Access* (DS-CDMA) [275] applies these principles within the *Data Link Layer*, but future work could reapply these principles within other layers or combinations of layers.

A biologically-inspired *synchronisation* phenomenon in which a closed system of oscillators, interacting in accordance with a set of simple rules, spontaneously achieves a mutually synchronised condition was proposed by Winfree [324, 325]. This phenomenon

has been studied comprehensively in the literature and is well known in multiple scientific disciplines [278]. Comprehensive surveys by Mirollo and Strogatz [201], and later Strogatz [277], illustrate the great diversity of biological processes and systems in which this phenomenon occurs in nature, and the development of analytical models with which the mechanism has been formalised and studied. Later work in this thesis exploits this phenomenon, as the mathematical foundation for networking protocols in chapter 7, but does not attempt to develop the mathematics further.

The original model of this phenomenon is due to Winfree [324]. It is nonlinear and difficult to solve in its generalised form [1], though a solvable version was later found by Ariaratnam and Strogatz [16]. An alternative model of the phenomenon due to Kuramoto [167] is easier to work with, and is exactly solvable despite its nonlinearity, and hence was generally favoured as the basis of following work in this domain [1]. Earlier work by Adler [3] preceded Winfree in the domain of stable electrical oscillator couplings, but considered only a single nonlinear oscillator, though this was later generalised by York [337] to describe systems of multiple similar coupled oscillators.

Under the *synchronisation* phenomenon described by the Kuramoto model [167] a closed finite system of periodic oscillators converge to a steady *equilibrium state*. Each oscillator in the system has identical period, but starts with arbitrary phase, and is prone to drift over time in the absence of external stimuli. System level coordination is an emergent property of independent agents implementing simple rules. Oscillators are pulse-coupled with their peers, and it is these interactions which determine the nature of the equilibrium state.

The Kuramoto model has an *order parameter* which quantifies the degree of synchronisation of the closed system of oscillators as time progresses toward ∞ [201]. The long-term behaviour of the system is entirely deterministic. If the *order parameter* ≈ 1 then all periodic oscillators fire simultaneously in the steady state. If the *order parameter* ≈ 0 then the firing times of periodic oscillators are evenly spaced throughout time in the steady state. The former steady state is a *coherent state*, whereas the latter steady state is a *incoherent state*. If the *order parameter* is somewhere between these extremes then a *partially synchronised* state exists in which periodic oscillator firing times are neither fully coherent nor incoherent.

Systems which tend toward, and have reached, a stable *coherent state* are referred to as *synchronised* systems [201]. Systems which tend toward, and have reached, a stable

incoherent state are referred to as *desynchronised* systems [226]. In a *synchronised* system the start and end times of two or more periodic isochronal events are identical. By contrast, in a *desynchronised* system these events are organised to maximise the inter-event period, which is equal for all pairs of events and their immediate successors. Note that this is not to be confused with an *unsynchronised* system, in which each event occurs periodically but without any inter-event coordination. *Desynchronisation* is usually defined within single-hop systems, but can be abstracted to a standard vertex colouring problem and applied within arbitrary topologies [152].

Wang and Aspel [314] construct primitives for synchronising oscillator systems connected through a wireless medium using the *synchronisation* phenomenon. It is observed that these primitives converge rapidly without global clocks, adapting automatically to changing oscillator populations. Unlike the *Phase-Locked Loop* (PLL) and *Delay-Locked Loop* (DLL) approaches, which offer similarly predictable and lightweight synchronisation behaviour, there is no requirement to maintain continuous contact between peers in the wireless medium.

Lucarelli and Wang [190] define a decentralised algorithm which is the first to exploit the *synchronisation* phenomenon to coordinate the timing of network activity. A sensornet of arbitrary logical topology applies a variant of the synchronisation-seeking algorithm defined in [201]; it is not required that the network graph is fully connected. Each sensornet node acts as a periodic oscillator but propagates its synchronisation signal only to nodes that are one hop away in the network topology. Over time, the entire system converges on a *synchronised* state.

DESYNC-TDMA [74] is a TDMA algorithm to *perfectly interleave periodic events to occur in a round-robin schedule* in a fully-connected network, and is the first to exploit the *desynchronisation* phenomenon to coordinate the timing of network activity. Each node acts as a periodic oscillator. Synchronisation signals are exchanged with peers defined by physical connectivity rather than logical network topology. TDMA timeslots are defined in terms of these synchronisation signals. The relative phase of signals measured within cyclical epochs is used to dynamically correct perceived error. Rapid convergence on a stable limit-cycle is guaranteed under ideal conditions, but disproportionately lengthy restabilisation periods result from small signal timing perturbations or network errors.

Kang and Wong [152] also employ the *desynchronisation* phenomenon as the foundation for a TDMA mechanism. *M-DESYNC* aims to address the hidden terminal problem

while maximising slot utilisation and hence network throughput. The main difference between DESYNC-TDMA and M-DESYNC is that the former allocates timeslots inflexibly to specific nodes, regardless of need, whereas the latter allocates more time to nodes with more neighbours and provides a priority-based mechanism for conflict resolution.

Christensen *et al.* [57] suggest that similar approaches can be applied in self-configuring systems of highly mobile robots. The physical topography of the implicit network can change very quickly owing to the high mobility of nodes. These self-organising strategies are particularly beneficial in highly dynamic and unpredictable situations, such as *Vehicular Ad-Hoc Networks*, where less agile approaches would struggle to maintain coordinated schedules. Superior throughput and noise levels of inter-robot communication are achievable if a shared communications channel access is coordinated through a *desynchronisation* mechanism than is achievable using a randomised allocation strategy.

Many other sensornet synchronisation approaches exist; a detailed survey by Sundararaman *et al.* can be found in [279].

2.6.4 Coordinated duty allocation

In a typical sensornet it is rare for all nodes to perform useful work at all times; usually a significant subset of nodes are waiting to execute pre-scheduled tasks or waiting for sensors to detect activity in the physical environment. Energy efficiency can be improved by carefully managing node state, placing some subset of the network in low-energy inactive states when not required to actively participate [87]. However, finding the optimal sleep schedule requires global knowledge of all node tasks and schedules to be maintained. Sensornets generally have insufficient resources to support the communication, computation and storage overheads of these optimal algorithms, with energy cost exceeding the resultant savings [86].

Low-level approaches minimise energy costs by identifying periods during which node subsystems are not fully utilised [108]. If components consume less energy when running at less than 100% capacity it is often beneficial to off-load activity from busy periods to less-busy periods, or to work speculatively in idle periods to minimise periods running at 100% capacity. The latter is intended to minimise periods during which 100% capacity operation is required, and to switch off unused resources to reduce the burden on non-renewable resources such as batteries.

The POCSAG protocol [306], used to distributed messages to pagers, defines the min-

imal set of periods during which devices must listen for possible broadcasts of relevant messages on a shared channel. The inverse of this is the much larger set of periods during which a device can switch off to conserve energy reserves. However, this implicitly defines a non-zero lower bound on message latency, typically orders of magnitude greater than actual message transmission times. System designers are forced to define these tradeoffs and design compromises if a minimum QoS is to be guaranteed [10].

Moving to sensornet protocols, the *Random Asynchronous Wakeup* protocol [225] implements a randomised and distributed algorithm under which nodes make local decisions on whether to sleep or remain awake. Within each time frame each node is awake for a randomly chosen fixed interval. When forwarding packets an integrated routing protocol selects from a set of equivalent next-hop locations with probabilistic guarantees that at least one of these will be awake. However, as there is no coordination between nodes there is no guarantee that any forwarding candidates are awake, and if more than one is awake this redundancy wastes energy.

Similar functionality is provided by the *Asynchronous Random Sleeping* scheme [135] which is principally useful where no inter-node coordination is possible. However, such scenarios might be considered unusual as sensornets generally execute distributed and cooperative sensing and processing applications.

The *Probing Environment and Adaptive Sleeping* protocol [335] implements an adaptive sleep policy in which nodes sleep for an exponentially distributed duration then wake and transmit a probe message. If any nearby nodes happen to be awake they transmit a reply message. If any such reply message is received the node is not required at this time and sleeps again; otherwise, it remains awake until it fails or runs out of energy. A significant weakness is that when a node fails there is zero local network coverage until some other nearby node wakes with indeterminate delay. If the failed node has accumulated significant data this cannot be replaced by that of other nearby nodes.

The *Lightweight Deployment-Aware Scheduling* algorithm [330] aims to improve network energy efficiency by switching off redundant nodes without access to accurate location or directional information. Observing that nodes require up to 11 active neighbours to provide a 90 percent chance of complete redundancy, LDAS allows network designers to tradeoff sensing redundancy against energy consumption. This protocol is most appropriate and efficient in networks where most nodes are required for physical sensing rather than for distributed data processing.

An alternative view [222] is that application-aware traffic scheduling, rather than network coverage, holds the key to maximising energy efficiency and hence network lifetime. The *Multi-Sensornets* approach applies a genetic algorithm to balance nodes' energy consumption in a distributed data fusion application. Peer nodes coordinate dataflow schedules such that the time during which they are required to be awake is minimised, allowing nodes to safely sleep at other times without disrupting network coverage. However, the resulting schedules are application-specific and do not consider sensing duty requirements.

An alternative to online state scheduling protocols is to specify schedules at deployment time using traditional scheduling algorithms. However, sensornets do not have central controllers to enforce or arbitrate shared schedules, and algorithms intended for reliable environments may function poorly or may impose too much coordination overhead in unreliable sensornets of limited resources. *Hard real-time* requirements may be unsupportable. A rich and diverse body of literature exists on the scheduling of periodic tasks in general systems; a comprehensive survey can be found in [185].

2.7 Summary

This chapter contains a survey of the relevant literature pertaining to the research hypothesis defined in section 1.4. The novel work which follows in the subsequent chapters of this thesis is set in the context of the literature discussed here, and is an extension of the existing work and results.

Chapter 3

Measuring sensornet behaviour

Sensornets are typically large systems, composed of many independent entities. Marshalling the pooled resources toward some shared goal is a non-trivial problem. Mastering the explicit and emergent complexity requires that we consider realistic scale systems, which implies we must make the problem tractable. This chapter considers methods for the objective measurement of sensornet behaviour, as required for meaningful analysis and comparative evaluation of candidate sensornet configurations.

3.1 Protocol failure modes and weaknesses

A consistent thread running through sensornet research is that they must operate under highly-constrained resource availability. Whether implemented using specialised discrete motes, or as functionality piggy-backed on existing equipment, it is vital that resource usage is kept to an absolute minimum. This maximises the lifetime of networks whose power sources cannot be replenished, and minimises the cost of necessary hardware.

Of course, it is equally important that the sensornet keeps pace with the real world with which it interacts, and has sufficient redundancy to recover from individual node failures. Contradictory requirements lead us to realise that multi-objective optimisation is essential to ensure that a reasonable compromise can be found (see section 2.5.5). To date there is a relative absence of studies examining sensornet behaviour at the scale necessary to provide confidence of their verity and applicability.

We measure the empirical response of network performance metrics to changes in network protocol configuration. By examining the consequent relationships, and similarities between such relationships, we can build models that give us insight into the tradeoffs and

compromises inherent in tuning and optimising a protocol. We show that these interrelationships are surprisingly complex even where only one parameter is controlled. We also categorise and measure types of suboptimal behaviour.

3.1.1 Measurement by simulation experiment

In this section we consider an experimental method with which to tune a networking protocol against multiple competing objectives through simulation, as implemented in sections 3.4 and 3.5. We consider the simulated network, the simulation tool, and the simulation environment.

Some interesting effects and behaviours may only become evident in sufficiently large networks. Preliminary simulation experiments, implemented to establish the characteristics of a suitable sensornet for the experiments discussed in section 3.4, considered networks of variable numbers of similar nodes distributed with constant spatial density. Qualitatively different behaviour was observed in networks of 200 nodes and of 500 nodes, with additional features and points of inflection appearing in plotted curves. Increasing node count further to 750, 1000 or 2000 nodes did not yield further features. We conclude that a test network size of 500 nodes is sufficient, giving a node spatial density of approximately 1.5×10^{-7} $node\ m^{-3}$ throughout the sensornet.

Measurement of network behaviour influenced by protocol tuning would ideally take place in physical testbed networks of realistic scale and composition. Unfortunately, economic and logistical factors preclude the construction of test networks on the order of hundreds of nodes for the experiments described in this thesis. All experiments were therefore implemented using the YASS sensornet simulation tool [287].

3.1.2 Experimental details

Three simulated networks were defined. Simulated nodes were based on the MICA2 sensornet mote [65] with a MAC layer based on IEEE 802.11 and radio range of around 150m, although this detail is largely irrelevant as any similarly-equipped nodes will yield similar behaviour. Each network contained 500 simulated nodes. Preliminary experiments showed that this network size is sufficient to reveal features in parameter-response relationships not evident in smaller networks. Networks of more than 500 nodes could be substituted with equivalent results, but with an increased simulation cost overhead.

Each network was identical in all regards other than node spatial distribution. By

averaging or otherwise compositing results across these three test networks we ensure that the characteristic quirks of any given network do not exert undue influence. Node spatial distribution within a bounding volume was random and even, with constant spatial density measured in units of *nodes per cubic metre* ($node\ m^{-3}$). A density of 1.5×10^{-7} $node\ m^{-3}$ was employed throughout the experiments.

This spatial density was selected with reference to the MICA2 radio range [65] such that the average degree of connectivity between a given node and its immediate neighbours was approximately 20. This degree of connectivity is typical of sensor networks [120], is within the *ad hoc horizon* limit of 10-20 nodes collaborating independently without hierarchical or external control [115], and facilitates energy efficient node cluster sizes [312].

In the simulated application each node serves as a packet source and packet sink, utilising the unicast paradigm throughout. Each node generates packets periodically with a single randomly selected destination to model a general distributed and decentralised process control application. Simulated packets have length randomly selected in the interval $[128, 1024]$ bits, including header. With the MICA2 radio having a transmit speed of $38.4 Kbs^{-1}$ [65] this gives per-packet transmit times in the interval $[3.33 \times 10^{-3}, 2.67 \times 10^{-2}]$ seconds.

When packet transmission begins the local wireless medium is occupied for some duration in this interval. Nodes implement *Carrier Sense Multiple Access* (CSMA) such that, if attempting to broadcast a packet, an exponential backoff procedure is implemented should the nearby wireless medium be occupied. A waiting node will implement up to 8 sense-wait cycles, doubling the wait period on each iteration, before giving up and dropping the packet. Note that although this greatly reduces packet broadcast collisions it does not avoid the *hidden terminal* problem [98], which is faithfully recreated in the simulation environment.

3.1.3 Computing resources

Simulation of large networks is a computationally intensive task [128]. Exploring parameter landscapes sampled at many points to understand behavioural tradeoffs and compromises implies a greater computational cost, as the evaluation of each sampling point implies the execution of at least one large simulation. In practice, the cost is higher still; each parameter landscape sampling point may be evaluated several times in several simulated networks.

The obvious remedy to this problem is to distribute the work among multiple computation hosts. Division of work might be achieved by running multiple simulations in parallel, or distributing a given simulation between multiple hosts. We apply the first approach as each test case can be executed in isolation from the remainder of the test suite. This avoids the significant coordination overhead implicit in the second approach; as any sensornet node can interact with any other node, it is notoriously difficult [340] to divide the problem into smaller subproblems with low mutual dependency.

Ideally, the computation work associated with each simulation could be divided between an arbitrary number of processing hosts to exploit high performance multi-purpose servers, low cost single-purpose resources such as hosts implemented using FPGAs, and the unused capacity of end-user workstations. The high resource demands of the current generation of network simulation tools implies that this will be difficult unless this goal is considered throughout the software design, and it will be difficult to retrofit existing simulation tools [128]. The YASS simulator used in these experiments was designed with this goal in mind but does not yet offer support for division of a simulation instance between multiple hosts.

Given finite resources and a potentially infinite search space, there is necessarily a tradeoff between that which we would like to evaluate by experiment, and that which we can realistically evaluate within reasonable bounded time. We implemented a principled search method which sampled that parameter space at a finite set of points. Our method does not require the individual simulation experiments to be conducted in any particular order. It is not necessary to wait for all planned experiments to complete before analysing data; it is possible to use the partial set of completed simulation instances to obtain preliminary results to assess whether it is worth continuing to completion. Of course, the more data points that are available for analysis, the greater the accuracy of results.

The experiments implemented for this section required over 100 days of computation time. Subsets of the computation job set were packaged for execution and managed automatically by suitably prepared scripts. The allocation of computation job sets to computers could be managed automatically by tools such as *BOINC* [13] or *Sun Grid Engine* [106] to any desired level of granularity. Owing to the lack of interdependency between any given pair of simulation experiment instances, however, it was not necessary to employ these tools.

The YASS simulator used for these experiments is implemented in Java [287]; simula-

tions can be executed on any platform capable of hosting a Java Virtual Machine without the need to recompile, and without consideration of problematic issues such as architecture endianness. The set of simulation experiments was divided among a heterogeneous set of computation hosts. The majority of these experiments were assigned to a set of high performance hosts specifically intended for heavy workloads, but other hosts were utilised including normal end-user workstations and laptops. Hosts employed the Linux 2.6.26.5 and 2.6.26.5-x86_64 kernels, OpenSolaris 2008.05 and Microsoft Windows XP. Simulation results from all hosts were combined into a single results set for analysis.

3.2 Protocols and their controlled factors

In this section we consider two protocols designed for sensornets, both of which implement a low-state lazy binding approach. Lightweight protocols remain relevant to the extreme resource constraints of small, low-cost motes and have the additional benefit that their complexity will not obfuscate the results of the methods proposed. For similar reasons the protocol chosen should be stateless, making no assumptions about the nature of the application, to avoid bias.

3.2.1 Protocol selection

Sensornets are an emerging technology which has already enjoyed some commercial success. However, deploying a sensornet in real-world applications remains a difficult, expensive, and error-prone challenge [28]. A consequence of this difficulty, and the cost of mote hardware [48], is that few sensornets of the scale considered in this thesis have been deployed into real environments. Despite the abundance of high quality protocols described and examined in the literature, no single sensornet routing protocol has yet attained the status of a *de facto* standard, as one might consider IP a *de facto* standard in commodity wired networking [79].

As sensornets are inherently application-specific [131] it is possible that a similar level of homogeneity is never attained. This is not necessarily a negative observation. If a sensornet is to be deployed into a specific environment, and is to be a self-contained system which does not interoperate with other sensornets, the network designer is free to select the most appropriate hardware, middleware and application software components and optimise the composition without regard for generality. Nevertheless, it is generally

desirable to reuse existing network components rather than create new examples, unless no suitable candidates exist.

We therefore select protocols which have been examined thoroughly in the literature and are intended for similar usage contexts. We consider two network routing protocols; *TTL Bounded Gossip* (TBG) [217] and *Implicit Geographic Forwarding* (IGF) [120].

3.2.1.1 Comparing TBG and IGF

Firstly, consider the similarities between TBG and IGF. These protocols are *stateless* and therefore do not consume storage resources to maintain routing tables or details of the underlying network. Furthermore, these protocols do not incur the energy and bandwidth overheads associated with distributing and maintaining this information, and do not suffer from problems arising from expired or redundant information. These issues are particularly relevant in highly dynamic or unreliable sensornets. The TBG and IGF protocols have low computational complexity and require little working space in memory, and are therefore ideal for sensornets composed of motes with few computational and energy resources.

Secondly, consider the differences between TBG and IGF. Perhaps the most significant difference is that IGF is geography-aware, whereas TBG is geography-ignorant. Sensornets are tightly coupled with the physical environment in which they are embedded, with data production, processing, consumption and storage generally being defined in terms of geographical position rather than node logical identity [257]. It follows that protocols which can exploit geographical context, which is typically unavailable in generic networks, are well-matched to geography-aware sensornet applications. Furthermore, IGF has a three-phase handshaking mechanism with which to deterministically allocate the packet relay role to a single node from a set of candidates, unlike TBG in which the packet relay role is allocated probabilistically to any number of the available candidates. If exploited correctly, this may enable efficiency improvements to be obtained by reducing redundancy and duplication of effort, at the expense of increased complexity.

3.2.1.2 Tuning TBG and IGF

If implemented carelessly these simple protocols can be highly wasteful, and hence represent an excellent opportunity for tuning. For example, unbounded flooded messages can easily cover the entire network [116] which is wasteful if the source and destination are physically close. More complex protocols often incorporate simple protocols during early

discovery phases or to maintain information. Gains achieved by optimising these simple protocols implicitly improve the performance of the more complex protocols in which they are incorporated.

For networks implementing either of the TBG or IGF protocols there parameters which are defined independently of any given network configuration, but can be tuned by a network designer to achieve a desired behaviour or to implement some resource usage tradeoff. Some tunable parameters are specific to a given protocol, but others are common to several protocols.

In selecting these protocols we make no claims as to their merit for any given sensornet application. More specifically, we do not claim that when optimally configured they necessarily offer superior performance to other recent and more complex alternative protocols. However, the methods described in this thesis can be reapplied without modification to the comparison of any arbitrary set of candidate protocols.

3.2.2 Protocol-independent controlled factors

Some tunable parameters are specific to a given protocol, but others are common to several protocols. In this section we define controlled factors $X_1 - X_5$ which are common to both *TTL-Bounded Gossip* and *Implicit Geographic Forwarding*, and may interact with other shared parameters and protocol-specific parameters. We define our experiments to explore as much of the parameter space as is possible. For each parameter $X_1 - X_5$ we limit our search to a subset of the defined range within which a measurable difference in response was expected. In general it is difficult to predict useful ranges by examining protocol definitions [295], so a set of preliminary experiments was implemented to find suitable ranges by a trial-and-improvement method [22].

X_1**: Seen packet buffer size** The number of packets received or transmitted by a node of which knowledge is retained. Nodes do not retransmit a previously-transmitted packet if the latter is held in this cache. A new packet displaces a randomly-selected cached packet if the buffer is full. Measured in *packets*. Defined in the range $[0, \infty)$ for integral values only. Search range is $[1, 10]$.

X_2**: Waiting packet buffer size** The number of packets which can be simultaneously enqueued for transmission or retransmission. Packets are consumed from the queue head and added to the queue tail. If the queue is full when a new packet is added,

a randomly-selected enqueued packet is dropped. Measured in *packets*. Defined in the range $[1, \infty)$ for integral values only. Search range is $[1, 10]$.

X_3**: Initial backoff** Before beginning transmission of a packet the sending node will sense the wireless medium. If the medium is clear transmission begins immediately, otherwise an exponential backoff strategy is applied in which the nth term is the nth power of this base value. Measured in *seconds*. Defined in the range $(0, \infty)$. Search range is $[0.1, 1]$.

X_4**: Packet lifetime** The maximum permitted time for a packet to remain in transit. If the lifetime is exceeded before reaching the destination, the packet is dropped. Measured in *seconds*. Defined in the range $(0, \infty)$. Search range is $[0.1, 10]$.

X_5**: TTL** The total number of node-node hops permitted for packets traversing the network. If this TTL is exceeded prior to reaching the destination, the packet is dropped. Measured in *hops*. Defined in the range $[1, \infty)$ for integral values only. Search range is $[1, 10]$.

Other networking protocols may be influenced by a different set of factors, which may or may not intersect the above set. However, any networking protocol for which there exists a set of quantitatively-defined factors can be explored using this process.

3.2.3 TTL-Bounded Gossip protocol

TBG [217] is the first protocol under consideration. This protocol is ignorant of energy, network topology, and the host application, ensuring no bias in the results produced. Flooding and gossiping protocols of this form are commonly used within more complex protocols [217] to establish delivery routes or maintain awareness of network status, widening the scope of our results to all such protocols.

The protocol makes no demands of a node wishing to broadcast a packet, either for packets newly created by the application or when forwarding packets. When a packet is broadcast, each recipient makes an independent probabilistic decision whether to rebroadcast the packet to its neighbours, if it is not to be dropped or consumed. The packet thus radiates outward from the source node, hopefully arriving at least once at each intended destination.

A detailed definition of the TBG protocol is given by algorithm 7 in appendix A. In

addition to the protocol-independent controlled factors defined above in section 3.2.2 an additional controlled factor must be specified.

X_6**: Gossip rebroadcast probability** The probability that upon receiving a packet, which is not to be consumed or dropped at the recipient, a given node will enqueue the packet for later retransmission to its neighbours. Unitless. Defined in the range $[0, 1]$. Search range is $[0, 1]$.

3.2.4 Implicit Geographic Forwarding protocol

IGF [120] is the second protocol under consideration. This protocol is ignorant of energy, network topology, and the host application, ensuring no bias in the results produced. Backtracking support, which is intended to work around network voids, is disabled as the networks considered in this thesis do not have any voids. This reduces the complexity of the protocol without affecting the results. As the backtracking mechanism has no tunable parameters it is of little interest for the protocol tuning work considered in this thesis.

Unlike flooding-derived protocols, IGF implements a three-phase handshaking sequence to moderate data packet broadcast. Consider a packet p with source A and destination D, currently at node S. Node S broadcasts a short *Request-To-Send* (RTS) received by neighbouring nodes $N_i \in N_{neighbours}$. Each RTS recipient N_i considers its geographic position relative to S and D. The geographic location of the three nodes $\{D, S, N_i\}$ defines the triangle $\triangle DSN_i$. The angle $\angle DSN_i$ is that subtended at vertex S of this triangle by the vectors $\overrightarrow{SD}$ and $\overrightarrow{SN_i}$.

If the angle $\angle DSN_i < \theta$ (where θ is a controlled factor X_7), node N_i broadcasts a short *Clear-To-Send* (CTS). $\angle DSN_i$ is trivially $0°$ if $N_i = D$. If S receives one or more CTS replies, it selects the node N_i offering the smallest $\angle DSN_i$ and selects this as the next recipient. Packet p is then broadcast with this choice added to its header. All neighbours $N_i \in N_{neighbours}$ except the selected N_i can safely ignore p. When the selected N_i receives p it sends a short Acknowledgement (ACK) to S, completing this stage of the process. The process repeats, with the previous N_i becoming the new S, until the packet arrives at D or a node N_i for which there are no suitable forwarding candidate neighbours.

The current simulated time is given by τ from each simulated node's internal real-time clock. Function $SEQ(\pi)$ extracts the IGF sequence number from packet π. Function $ANGLE(\sigma)$ extracts the CTS angle stored in CTS packet σ. Function $SENDER(\pi)$ obtains the identity of the last sender of packet π (not necessarily the original source).

Function $DEST(\pi)$ extracts the destination specified in packet π. Function $RELAY(\pi)$ extracts the selected next-hop relay node from packet π if defined for π.

A detailed definition of the IGF protocol is given by algorithms 8 and 9 in appendix A. Algorithm 8 defines the required behaviour for a packet relay candidate or destination N_i. Algorithm 9 defines the required behaviour for a packet sender S. In addition to the protocol-independent controlled factors defined above in section 3.2.2 two additional controlled factors must be specified.

X_7**: CTS threshold angle** When node N_i receives a CTS message from S, it will not send an RTS unless $\angle DSN_i < X_7$. This factor is intended to prevent many low-quality or poorly located forwarding candidates sending RTS messages, and prevents packets being forwarded in the opposite direction to the destination if $X_7 < 90$. Measured in *degrees*. Defined in the range $[0, 180]$. Search range is $[5, 85]$.

X_8**: State timeout base** Complete IGF cycles imply several wait/timeout periods. To minimise the search space we define all as multiples of a single parameter X_8, such that $CTS_WAIT = X_8$, $DATA_WAIT = 2X_8$, and $ACK_WAIT = X_8$. Measured in *seconds*. Defined in the range $(0, \infty)$. Search range is $[0, 1]$.

3.3 Measurable attributes of solution quality

To assess the relative or absolute quality of a given candidate protocol tuning we must first make measurable its behaviour in a representative application context. Each metric should correspond to some desirable notion of solution quality. For a given candidate solution, specified by a set of input controlled factor values for a given network protocol, we define the quality of this candidate solution in terms of a set of network response metrics derived in [295].

Three aspects of solution quality were measured; *performance*, *reliability*, and *efficiency*. Sensornets are inherently real-time systems [274] and hence sufficient *performance* is required to deliver packets by deadlines. Data are more important than hardware entities in sensornets [257] and hence sufficient *reliability* is required to ensure data are not lost in transit. Wireless communication is usually the largest energy consumer [235] and hence sufficient *efficiency* is require to ensure sensornets continue operation over the required timespan. This list of solution quality aspects is not intended to be exhaustive. Sensornet

designers can define other aspects of solution quality, and define suitable metrics with which to measure these, if appropriate to the intended application.

For each aspect identified as being significant to a greater notion of quality, a metric was defined against which to measure the extent to which a given protocol configuration satisfies this notion of quality when implemented within a simulated network. This set of metrics was reduced to the minimal set deemed sufficient to capture the characteristics of interest to minimise redundancy and experimental overhead [22]. We define our metrics in terms of SI units.

For each metric $M_1 - M_3$, lower values imply more favourable behaviour. A value of zero represents a perfect solution in a given metric, although in practice this may not be attainable; the optimal value may be somewhat higher, but the value is not known in all cases. Where metrics are defined *per metre*, this is to normalise results in the size of the network. This is essential in order that results be comparable between networks of different node count, node distribution in the network space, or physical size. Where metrics are defined *per packet*, this is to normalise results in the volume of traffic handled by the network to enable fair comparison between relatively busy or quiet networks, a property which is not a controlled factor but for which we must account.

3.3.1 Performance

Network performance is defined in terms of *normalised latency*, which is the average time taken for a packet to traverse unit distance within the network. This is important because in most real-world applications it is not sufficient for a network to guarantee that a packet will eventually be delivered. In real-time applications, such as a typical sensornet application, it is important that packets are delivered within a given deadline. Knowledge of the average latency per unit distance allows the network designer to calculate the physical speed at which data traverses the network.

M_1**:** *Latency per metre:* Mean time for a packet to travel 1 metre. Measured in $m^{-1}s$. Defined in the range $(0, \infty)$.

3.3.2 Reliability

Network reliability is defined in terms of *packet delivery*. Ideally, every packet generated by the simulated sensing application and queued for delivery at the source node would eventually reach the destination node within the delivery deadline. The source node and

destination nodes are not interested in how this is achieved, or the route taken through the network; these are details that are delegated to the network middleware.

M_2**:** *Packet delivery failure ratio:* Proportion of packets created at source nodes by the simulated application which the network attempted to deliver, but were lost before reaching their intended destination. Unitless. Defined in the range $[0, 1]$.

3.3.3 Efficiency

Network efficiency is defined in terms of average *energy* consumed by the network to deliver a packet from source to destination. It is generally impossible to define the energy consumed in delivering a specific packet so instead an average is obtained for all packets successfully delivered. Packets which are not successfully delivered nevertheless induce the consumption of energy during the delivery attempt until all potential delivery branches are truncated prior to reaching the destination.

M_3**:** *Energy per packet per metre:* Mean energy for 1 packet to travel 1 metre. Measured in $Jpacket^{-1}m^{-1}$. Defined in the range $(0, \infty)$.

3.3.4 Comparing quality of candidate solutions

The metrics M_1 to M_3 defined above are all mutually independent and may be targeted as individual objectives by sensornet designers. However, real sensornet designs are likely to require an acceptable compromise between multiple competing objectives. It is therefore necessary to define a mechanism by which the relative quality of two or more candidate solutions can be compared to determine which offers the best compromise.

Assume we have n controlled factors X_1-X_n and m metrics M_1-M_m. A candidate solution $S_\alpha = \{X_{\alpha 1}, \ldots, X_{\alpha n}\}$ maps to a set of metrics $T_\alpha = \{M_{\alpha 1}, \ldots, M_{\alpha m}\}$. The mapping of $S \mapsto T$ is not known *a priori* but instead is evaluated experimentally for specific values of S. A perfect solution $S_{perfect}$ would yield a set of metrics $T_{perfect}$ such that $\forall M \in T_{perfect} \bullet M = 0$. Although $S_{perfect}$ does not necessarily exist, we define the quality measure E in Equation 3.1 of any given candidate solution S_α based on the Euclidean distance [88] from the point in solution phase space defined by T_α to the point $T_{perfect}$.

$$E = \sqrt[2]{\sum_{i=1}^{m} w_i (s_i M_i)^2} \tag{3.1}$$

It is important to remember that minimising the E metric is *not* the objective of a *single-objective* optimisation process; it is merely a tool with which to select a single best compromise solution from the set of high-quality candidates identified by the *multi-objective* optimisation process. Some network performance attributes may be of greater importance than others to a sensornet designer. We therefore define weighting w_i for metric M_i such that a larger weighting value indicates a greater importance attached to the network performance attributes quantified by a given metric.

Each of the metrics M_1-M_m may be defined over a different range, so it is inappropriate to compare the absolute measured values directly. We define a normalising factor s_i for metric M_i such that all possible values of $s_i M_i$ are found in the range $[0, 1]$, noting that the the ideal value of any given metric is also the lowest possible value, 0. It is only meaningful to compare two E values if all normalising values s_i are equal for each E. If for a given metric M_i is defined over a finite range then the value of s_i is well-defined and does not vary between network configurations under consideration. However, if a given metric M_i is defined over an infinite range then there does not exist a single well-defined value of s_i. Instead, we define s_i in the context of a given set of experimental results by setting $s_i = 1 \div MAX(M_i)$ where $MAX(M_i)$ is the largest value of metric M_i observed.

In the experimental work that follows we set all $w_i = 1$ to give equal weighting to all metrics, and set all s_i using the second definition above as some metrics defined in section 3.3 are defined over an infinite range. As we consider 3 metrics M_1-M_3 it follows that all values of E are defined in the range $[0, \sqrt{3}]$ where 0 is the solution quality deriving from the theoretically perfect solution and $\sqrt{3}$ is the solution quality deriving from the worst quality solution considered in the set of all experiments. For the experimental results we present later we scale all E values by a constant factor of $1 \div \sqrt{3}$, mapping the range $[0, \sqrt{3}]$ to $[0, 1]$ for ease of comparison while remaining equivalent to the original.

Sensornet designers may need to find solutions conformant to specific constraints in one or more metrics. For example, if the sensornet application requires at least 50% of packets to be delivered successfully, then candidate solutions for which $M_2 \geq 0.5$ are unacceptable. This can be achieved by extending the solution quality function defined in Equation 3.1 to assign infinite E-values to candidate solutions which do not satisfy some arbitrary set of requirements.

Assume a boolean function $reject(T_\alpha)$ applies the set of relevant tests to the output metrics T_α corresponding to candidate solution S_α, returning $false$ if acceptable and

true if not acceptable. We extend Equation 3.1 using Iverson notation [160] to obtain Equation 3.2 which assigns $E = \infty$ for unacceptable solutions, and leaves E unchanged for acceptable (though perhaps sub-optimal) solutions. Candidate solutions S_α which do not meet specified constraints and hence have $E(T_\alpha) = \infty$, but which might otherwise have good E-values, are evaluated during the search process but are never selected.

$$E = \sqrt[2]{\sum_{i=1}^{m} w_i(s_i M_i)^2} + [reject(T_\alpha)]\infty \tag{3.2}$$

3.4 Offline static protocol tuning results

In this section we consider the various dissimilar types of relationship which exist between network protocol configuration and network performance metrics. Each experiment uses the network configurations defined in section 3.1.2.

Protocol configurations were evaluated in which only a single parameter, the static gossip rebroadcast probability p, was changed. 20 values of the parameter p were evaluated, distributed evenly in the interval $[0, 1]$. Graphs were plotted of p against observed metric values. In packet-centric metrics, where multiple copies arrive at the destination we consider only the first.

It is important that the data extracted from simulation experiments are valid, and meaningfully reflect network behaviour over an extended period. This was achieved by running simulations for a reasonably long period of simulated time prior to extracting values for metrics. Taking the measurements at this stage ensures that they accurately reflect the long-term behaviour of the simulated system.

To determine the required simulation length, a series of preliminary experiments was performed using a representative simulated system. The value of each metric of interest was sampled periodically during simulation execution. When the simulation was complete, an analysis was performed of the change in metrics over time as the series of measurements converged on their limiting values. For each metric, after a certain period of simulated time had passed, variation in measured values over an extended period was no greater than $\pm n\%$ where n represents the acceptable measurement error.

When the condition holds true that the sequence of measured value fall within, and remain within, the acceptable measurement error, we consider the measurement sufficiently stable to be representative of long-term behaviour. The simulation time required to reach

this condition was measured, doubled, and rounded upward, yielding a requirement that each simulation experiment execute for 600 simulated seconds.

Each simulation was allowed to execute for 600 simulated seconds, after which the simulation was terminated. The value for each metric of interest was then extracted, reflecting the entire period of 600 simulated seconds. This includes the initialisation stage at system startup. This is valid and appropriate because network traffic generation, and the set of rules defining network behaviour, is constant and unchanging throughout each simulation experiment.

Multiple related plots have been consolidated within figures in section 3.4 to allow easy comparison of related network measures. Where measured response values have incompatible units or scales, the observations have been normalised for each observation type such that the observed value range $[min, max]$ is mapped onto the range $[0, 1]$.

For each measured effect we have the objective of maximising the desired behaviour, measured by one or more metrics. It is shown that no single value of p is ideal in all metrics. It follows that a multi-objective protocol optimisation mechanism is required to obtain an acceptable compromise solution. Chapter 4 defines and applies suitable multi-objective optimisation methods.

3.4.1 Local versus global traffic effects

Figure 3.1 traces A and B show that the probability of successful point-to-point transmission, and to a lesser extent reception, declines with increasing p. As network utilisation increases, so does congestion and the possibility of overlapping transmissions interfering.

An exponential backoff mechanism is implemented to regulate attempts to utilise the shared wireless medium. As congestion increases, nodes generally must wait longer for the wireless medium to become clear for transmission. This exponential backoff may be preempted at a given node, to receive a packet of interest to that node. If some other node begins a packet transmission during a waiting period, and the transmission extends over the next potential transmission time, another iteration of the backoff mechanism is required (the complement of trace A). Eventually, the attempt to transmit a packet may timeout completely, if the backoff procedure is exhausted (trace C).

As all nodes share a wireless communications medium, as the number of concurrent transmissions within that medium increases so does the opportunity for packet reception at receiving nodes to experience data corruption. Where multiple packet transmission

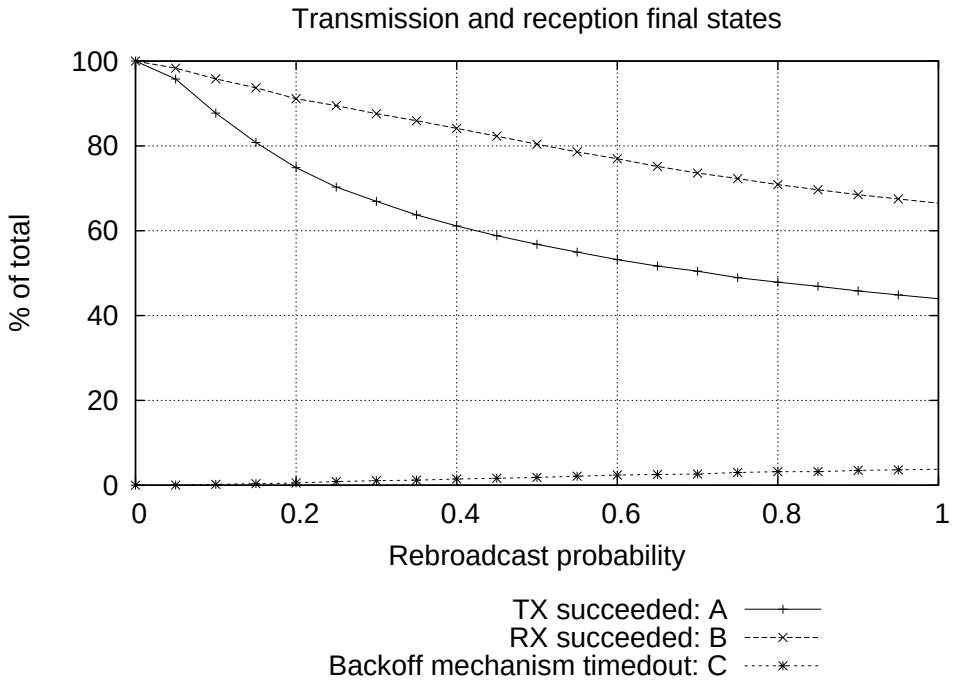


Figure 3.1: Transmit/receive final states

signals overlap and are received simultaneously, these signals may be of sufficient strength to interfere and interact beyond the recovery capacity of error detection and correction mechanisms. If this occurs, the listening node may be unable to successfully receive any of the overlapping packet transmissions, and any ongoing packet reception attempts will fail (the complement of trace B). The *hidden terminal problem* is a common source of this issue in wireless networks [98].

As network activity increases with p we find that all individual network actions become less reliable, an effect which must be countered by protocols which can recover from failed individual actions. Flooding and gossiping can achieve this goal, albeit somewhat crudely by simply repeating many redundant copies of each attempted action.

In figure 3.2 trace C we observe the anticipated bimodal behaviour in which either most nodes receive a packet, or very few do, along multi-hop delivery paths. Increasing p increases delivery probability, with a sharp transition at the critical probability p_c [116]. Increasing network size brings a sharper transition. For $p > 0.6$ the probability of a packet being delivered is steady at around 92%.

In contrast, traces A and B indicate that individual node-to-node pairwise message exchanges become less reliable as p increases as a result of increased network utilisation, and hence increased congestion and interfering transmissions. Single-hop communications

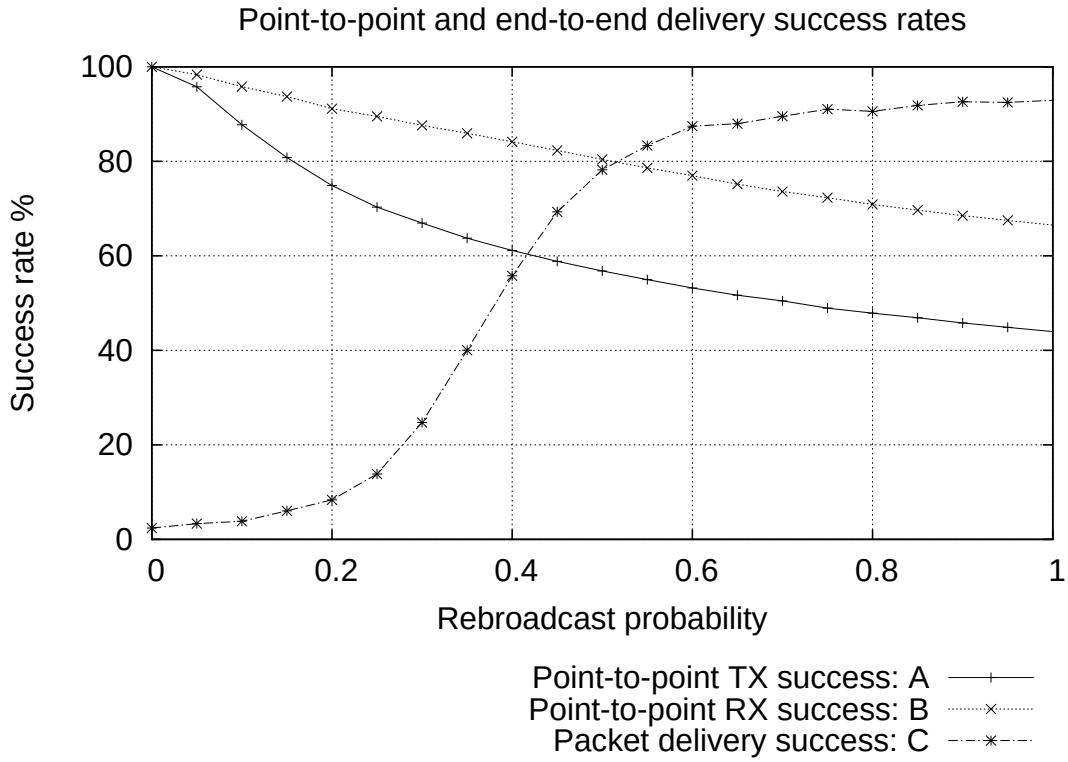


Figure 3.2: Delivery success rates

become less reliable, and yet the multi-hop communications composed of single-hop communications become more reliable.

This would seem paradoxical as the success probability of any given multi-hop path is the product of all single-hop component probabilities. However, as p increases the number of potential delivery paths explored by packet copies increases rapidly. The success probability of each potential path decreases, but the rapid growth of path count greatly outweighs this effect. Sensornet designers must consider the expected distribution of path lengths in determining suitable p. Where long multi-hop paths dominate a high p is beneficial, but where single-hop or short paths dominate then a lower p may work better.

3.4.2 Route optimality and coverage

Figures 3.3 and 3.4 illustrate a number of characteristics of packet delivery routes with respect to rebroadcast probability p. Trace A shows *direct path distance*, the average Euclidian distance between source and destination for delivered packets. Trace B shows *path straightness*, which indicates the closeness of the actual physical route traversed by the packet to the ideal straight path. If the Euclidian distance between source and destination

is x, and the sum of the Euclidian distances of each node-to-node hop along the actual packet delivery path is y, then route straightness $z = {}^{x}/_{y}$. Ideally, $z = 1$. Trace C shows *network distance*, the average number of node-to-node hops along the routes traversed by delivered packets.

The differing shapes of traces $A - C$ indicate that fundamentally different trends in observed factors are induced by variation in a single controlled factor, p. Note that traces $A - C$ illustrate different network properties. For example, trace A illustrates the *physical* distance traversed by packets, whereas trace C illustrates the *logical* distance traversed by packets. Although it is meaningless to directly compare values of incompatible measures, it is useful to compare the trends, to consider whether variation in a given controlled factor improves or worsens different measures of network performance.

As stated in section 3.1.2, each node generates packets with randomly selected destination. It follows that the average Euclidian distance between source and destination for *attempted* packet delivery is independent of p. However, the experimental results indicate that the average Euclidian distance between source and destination for *successful* packet delivery is dependent on p. Trace A in figures 3.3 and 3.4 shows that, as p increases from 0, the average physical distance d_a between the source and destination of delivered packets initially increases, then levels out at around p_c where $p_c \approx 0.4$ for this network. For $p < p_c$ node pairs separated by distance $d > d_a$ are very unlikely to successfully exchange messages owing to packet propagation dying out early. Network applications requiring packets to travel long distances must select a suitably high p; applications in which packet delivery paths are always physically short (e.g. hierarchical aggregation) may cope with lower p.

Figure 3.3 trace B shows that the physical path followed by packets from source to destination is often far from the ideal straight line. In the degenerate case where $p = 0$ each path is perfectly straight as each consists of exactly one node-to-node hop. As p increases to p_c it becomes possible for node pairs located more distantly to exchange packets, though the delivery paths become less straight (and less efficient) as few nodes rebroadcast, and those that do so need not be located along the optimal straight path. However, as p increases beyond p_c it is more likely that a node choosing to rebroadcast will lie on, or near, the straight path. As packets encounter delays at each hop, it is more likely that shorter, straighter routes with fewer hops will induce less delay and hence deliver packets earlier than more tortuous routes. This effect is reflected in trace C in figure 3.4.

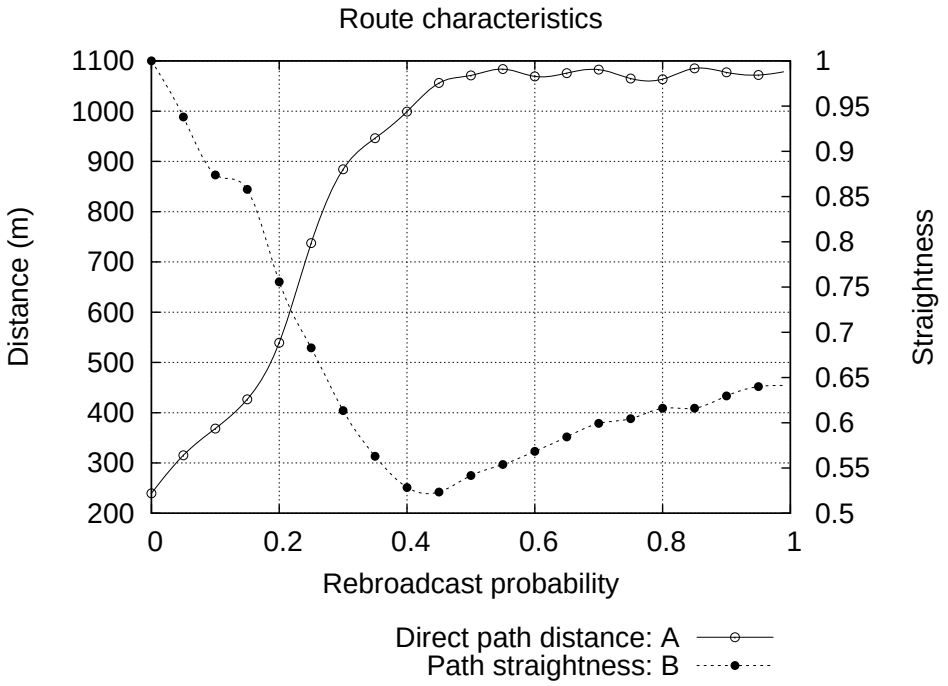


Figure 3.3: Delivery route characteristics: straightness

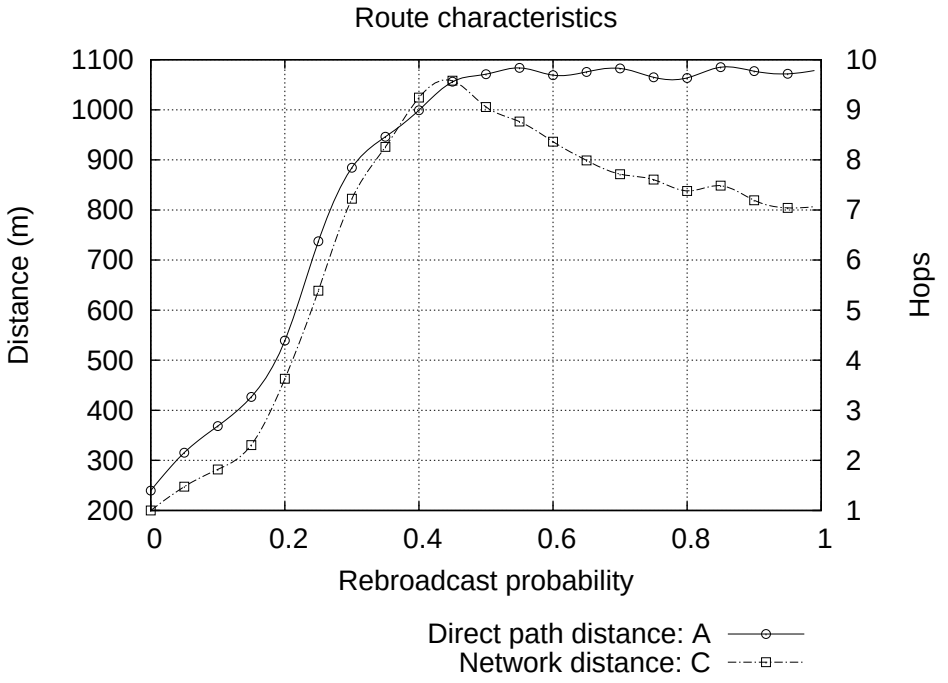


Figure 3.4: Delivery route characteristics: hopcounts

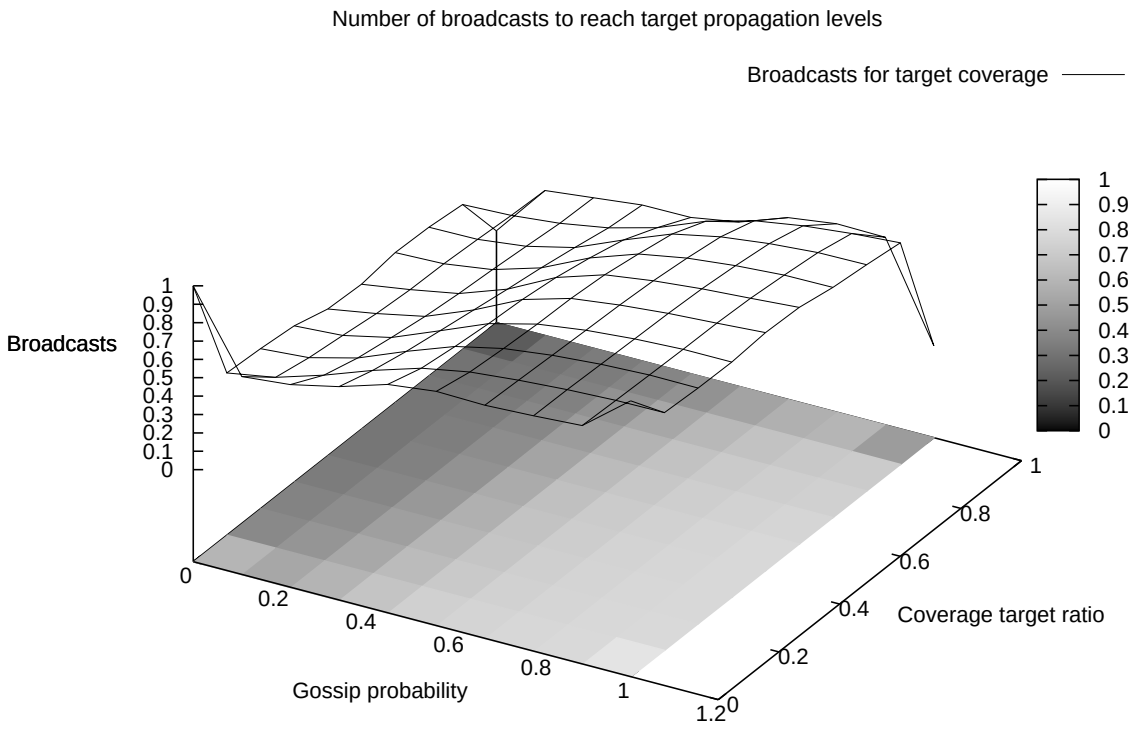


Figure 3.5: Coverage versus broadcasts

Figure 3.5 shows the three-way relationship between p on the x-axis, the proportion q of network exposed to a packet on the y-axis, and the total network-wide count r of broadcasts of this packet on the z-axis, normalised for packets successfully delivered only. q is equivalent to the probability that any given packet is delivered, and r is a heuristic measure of energy expended in the delivery attempt. Planar slices through the surface parallel to the yz-plane give the relationship between delivery probability and energy consumption for given p.

The surface can be approximated as two plateau parallel to the xy-plane, divided by a curve of shape similar to trace C in figure 3.2. This configuration illustrates that successful delivery attempts tend to induce a similar high number of broadcasts, and unsuccessful delivery attempts tend to induce a similar low number of broadcasts. If the surface had been closer to an inclined plane this would have indicated that sensornet designers could more finely tune the tradeoff between delivery success probability and energy consumption to suit application requirements.

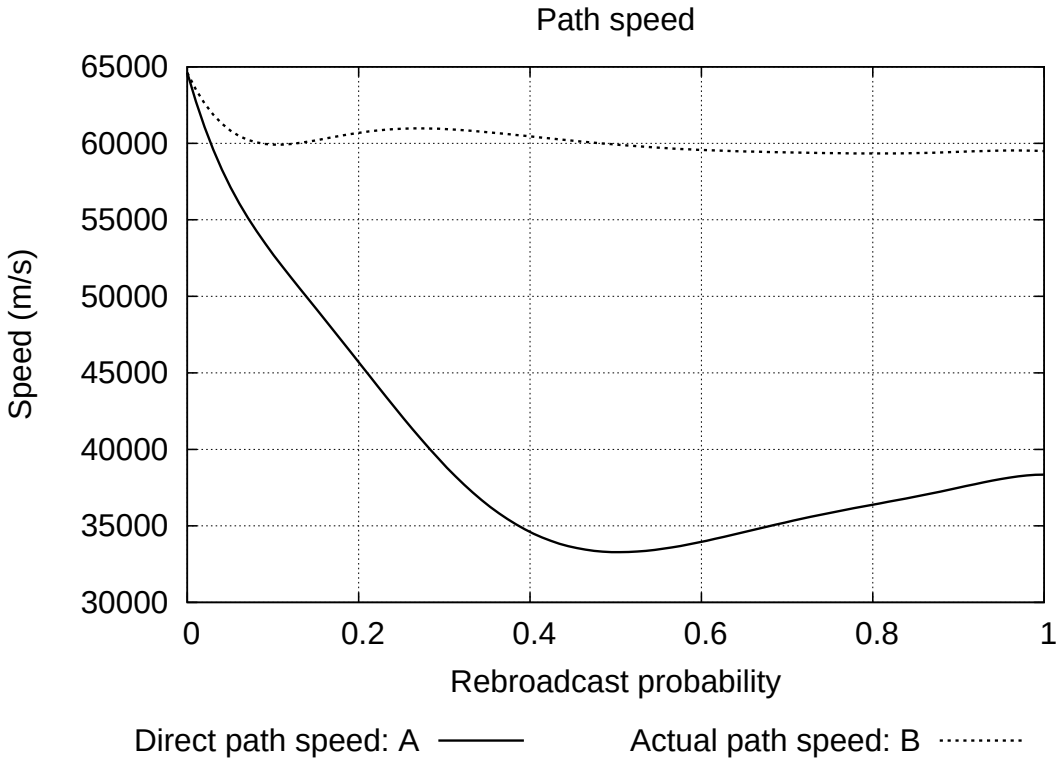


Figure 3.6: Path speed

3.4.3 Timing and latency effects

Figure 3.6 traces A and B indicated the speed at which packets traverse the network. Trace A shows the virtual packet speed v, as would be demonstrated by a packet travelling on a perfectly straight path between source and destination. This measure is useful in predicting time-bounded reachability for real-time applications. Packets with source-destination distance d and deadline t are likely (though not guaranteed) to arrive on time if $\frac{d}{t} \geq v$.

Figure 3.6 trace A shows that, as p increases, v declines rapidly due to network congestion until $p \approx p_c$ at which point the general increase in path straightness (see figure 3.3 trace B) allows s to increase. As p increases from 0 to p_c successful delivery becomes more likely but slower; for $p > p_c$ both speed and success probability increase in tandem. This is despite the actual physical speed remaining near-constant in increasing p, as shown by trace B in figure 3.6.

However, for $p > pc$ the delay at each node does not increase appreciably, as illustrated in figure 3.7 trace A; nodes have limited buffer space and packets have limited lifespans, so packet queues cannot grow without bound. As figure 3.7 trace B illustrates, if the

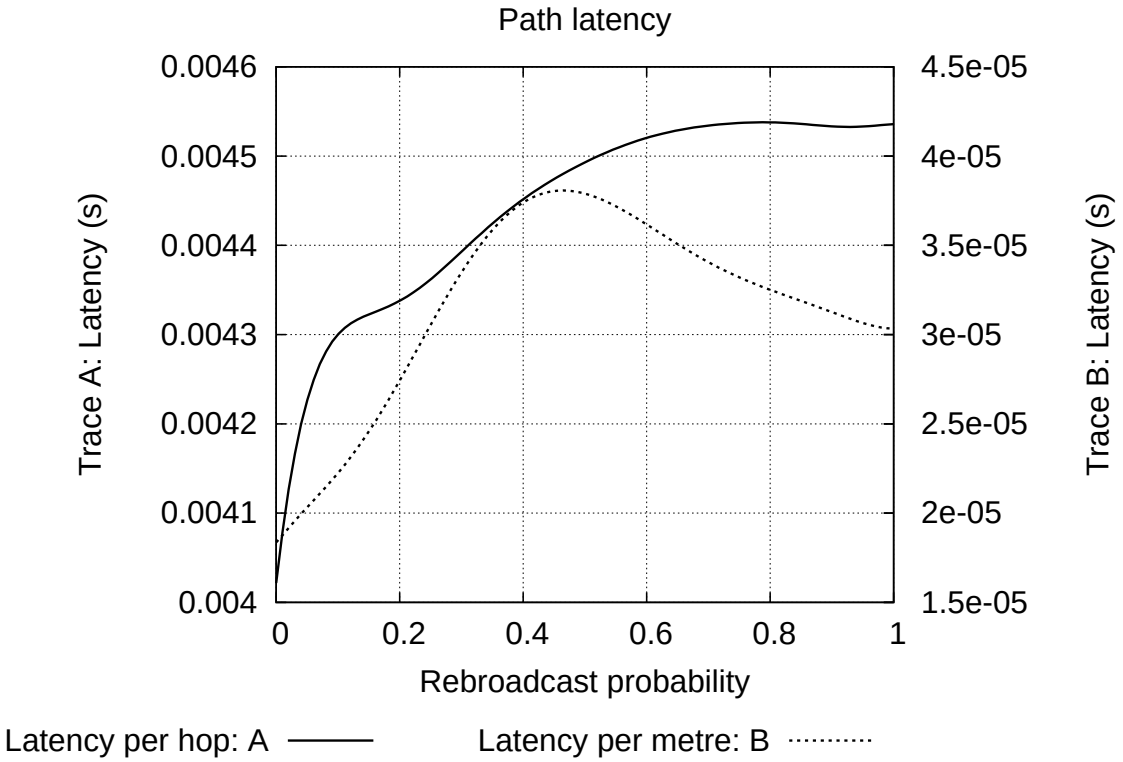


Figure 3.7: Path latency

virtual packet speed v increases but routes become straighter as p increases where $p > p_c$, then the time taken to travel some constant proportion of the route becomes smaller. Consequently, network designers may wish to decrease buffer sizes and packet lifetimes where newer packets are of greater value than older packets.

Figure 3.8 shows the three-way relationship between p on the x-axis, the proportion q of network exposed to a packet on the y-axis, and the average latency for delivered packets on the z-axis. Planar slices through the surface parallel to the yz-plane give the relationship between delivery probability and likely delivery latency; expected latency grows almost linearly in delivery probability, with some slope s_p. In other words, this relationship gives the probability that a packet will be delivered within a given time, assuming that it will be delivered at all.

This reinforces the interpretation in sections 3.4.1 and 3.4.2. The more network activity in response to a delivery attempt the greater the chance of success, but the lesser the delivery speed. Note that s_p is not constant; s_p increases with p from 0 reaching a maximum at p_c, decreasing once again as p approaches 1. Network designers requiring minimal variation in delivery latency might select p distant from p_c toward the extremes.

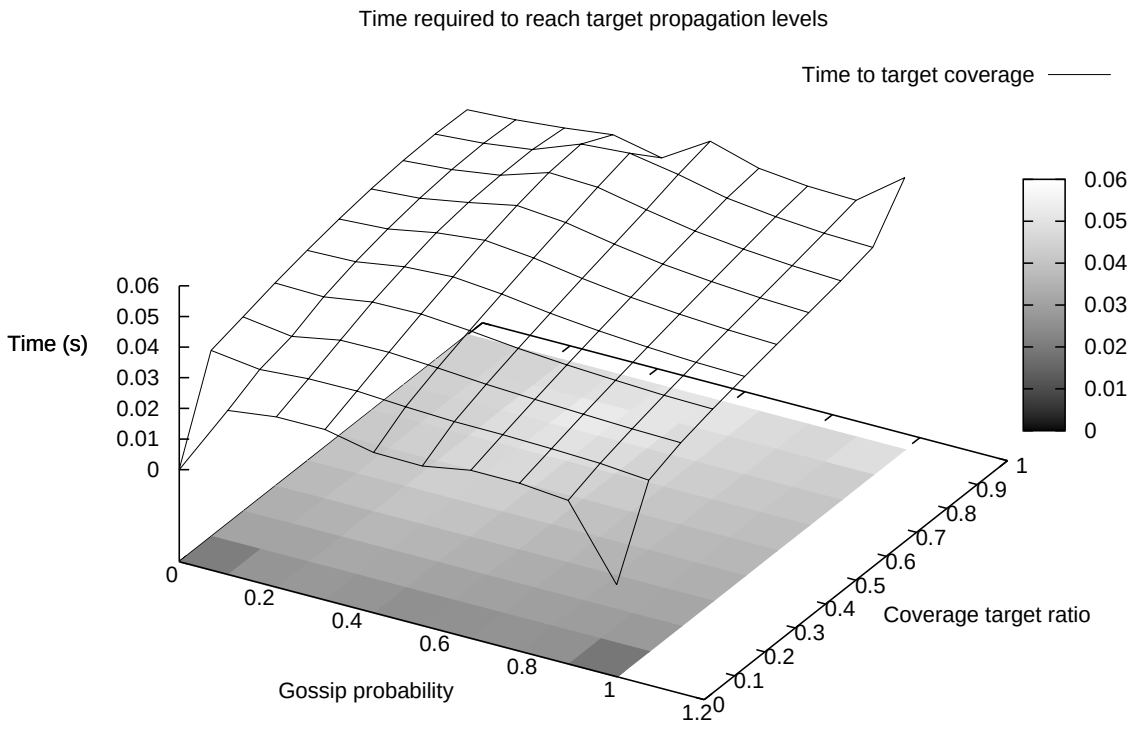


Figure 3.8: Coverage versus time

3.4.4 Energy efficiency

Figure 3.9 illustrates trends in two different measures of efficiency as a function of varying rebroadcast probability, p. Both traces are derived from figures for successfully delivered packets only, although the measured values are influenced by all packet delivery attempts. Trace A illustrates the average number of duplicate copies received at the destination for a successfully delivered packet. Ideally, exactly one copy would be received. Trace B illustrates the average amount of energy consumed by the network to deliver at least one copy of a packet to its intended destination. Lower values are preferable. Traces A and B relate to different network effects, so it is not appropriate to compare these types of raw data directly. However, it is useful to compare the resulting trends, so as to identify values of p which yield favourable performance in both network performance issues.

Figure 3.9 trace A indicates that as p increases nodes tend to receive more copies of packets, the number of duplicates growing logarithmically in p. This indicates that multiple independent delivery paths are active, as each node never broadcasts a given packet more than once. This is good for reliability, but might be considered bad for efficiency; after a packet has been received, there is no benefit derived from receiving it

again.

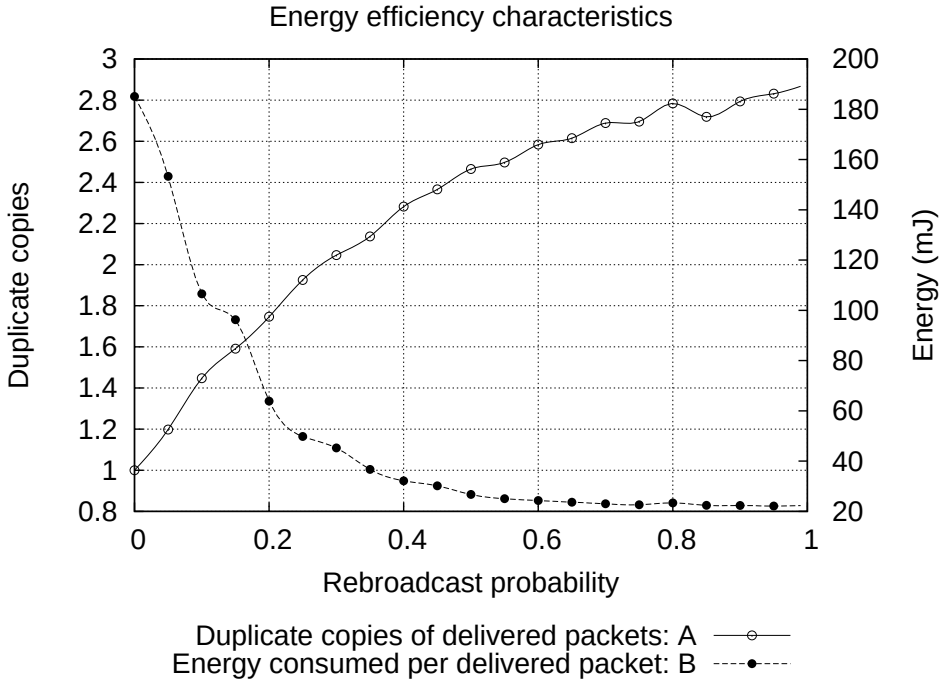


Figure 3.9: Energy efficiency characteristics

However, figure 3.9 trace B indicates that as p increases, and the level of network activity grows in response, the energy consumed per delivered packet falls sharply. This somewhat counter-intuitive finding is explained by increased reliability. A packet is either delivered, or it is not delivered; however, the energy invested in attempting to deliver a packet does not follow a similar binary relationship. As p increases the probability of delivery increases, and hence the probability that energy invested in packet delivery being thrown away decreases. This suggests that efficient protocols should fail as early as possible if delivery of a given packet will ultimately prove unsuccessful.

Now consider the further energy efficiency improvements that might be achieved by sensornets implementing an efficient node state management policy. Figure 3.10 trace A shows energy consumption for nodes that never switch off. Trace B shows energy consumption for nodes which are switched off whenever not performing useful work, but otherwise behave identically. It can be seen from these traces A and B that there exists the potential to save energy without compromising performance, although the overall pattern of the relationship is unchanged.

Figure 3.10 trace C shows the proportion of energy consumed under trace A which would be saved. Note that the plot is scaled to fill the y-axis; the actual wasted proportion

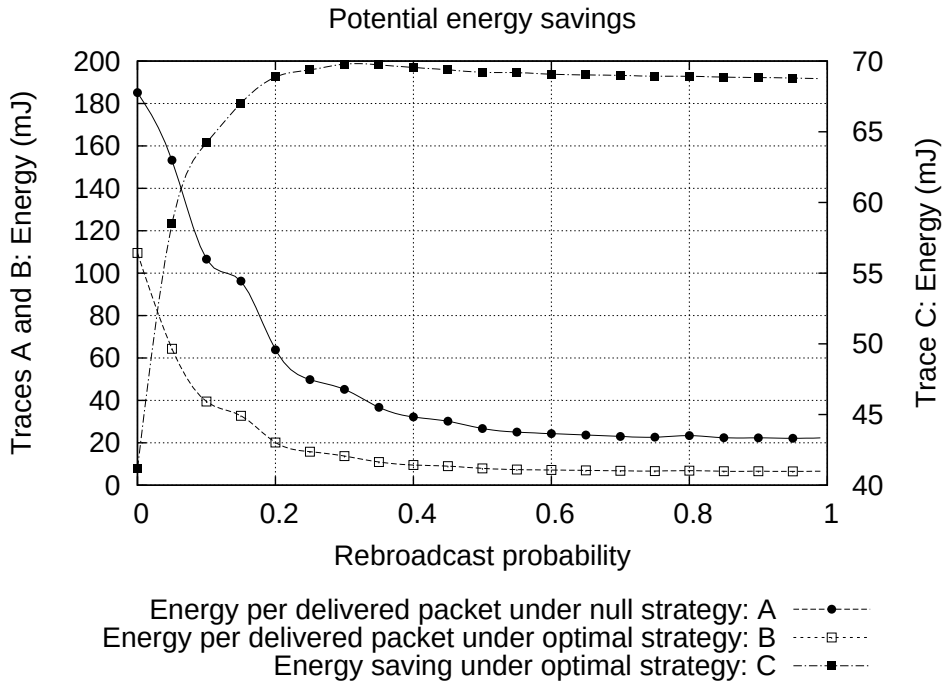


Figure 3.10: Potential energy savings

is around 45%, and real state management policies are unlikely to reach this level. Nevertheless, this illustrates that significant energy efficiency improvements might be possible if energy management and network activity management are integrated. Chapter 7 defines and evaluates suitable mechanisms.

3.5 Online dynamic protocol tuning

In this section we consider weaknesses in gossip protocol performance, proposing and evaluating remedies.

We define a set of metrics in section 3.5.1, a subset of those considered in section 3.4, to identify failure modes of the protocol. By examining the frequency with which values of these metrics are observed we quantitatively characterise the conditions under which packet delivery attempts fail. These metrics were chosen because all but one can be determined online within the network at minimal cost, and hence can be used to dynamically tune protocols to improve routing decisions with minimal overhead.

Each simulation was allowed to continue for 3600 simulated seconds to allow network behaviour to settle into stable patterns. A single gossip protocol configuration was taken as a baseline with static gossip rebroadcast probability of p throughout the network. We

set $p = 0.6$ as section 3.4.1 indicates this is sufficient to ensure delivery of most packets to most nodes at most times under the bimodal behaviour effect [116].

The frequency distribution for each failure characteristic metric was determined by taking the output from three simulations, corresponding to the three networks described in section 3.1.1, and counting the frequency of values falling into equal-width bins spaced evenly throughout the observed range. Experimentation showed that 20 bins provided a good balance between detail and clarity in the derived histograms, exposing trends without too much distracting noise.

Graphs were plotted showing bin midpoint values versus relative frequency. Each plot summarises the observed behaviour in a given response type over the total simulated period, and approximates the probability distribution function of the response value for any given individual packet similar to the greater packet population [270].

For each measured attribute described in section 3.5.1 a pair of graphs illustrates the approximated distribution function. The first graph shows the distributions for all packets, delivered packets, and undelivered packets. This graph illustrates undesirable behaviours evident for undelivered packets which differ from those of successfully delivered packets. The second graph shows the distribution of undelivered packets under variants of the gossip protocol, modified to address the suboptimal behaviour displayed in the first graph.

Each attribute is measured with respect to the *closest delivery attempt node*, C. We assume each packet π has a single source node, S, and a single destination node, D. For delivered packets, trivially $C = D$. For undelivered packets, C is the node geometrically closest to D which successfully received a copy of π.

Figure 3.11 illustrates the distribution of distances from *source* to *closest*, $||\overrightarrow{SC}||$. Figure 3.12 illustrates the distribution of distances from *closest* to *destination*, $||\overrightarrow{CD}||$. Figure 3.13 illustrates the distribution of hopcounts encountered by the first instance of packet π reaching C. Figure 3.14 illustrates the distribution of time elapsed during which the first instance of packet π reaches C.

Figure 3.15 illustrates the distribution of total numbers of node-to-node hops observed in delivery of a packet. This latter figure is taken as a heuristic measure of the total network resources expended in attempts to deliver a packet, assuming that each node-to-node hop consumes similar levels of resources of interest such as energy.

3.5.1 Distribution analysis: unmodified protocol

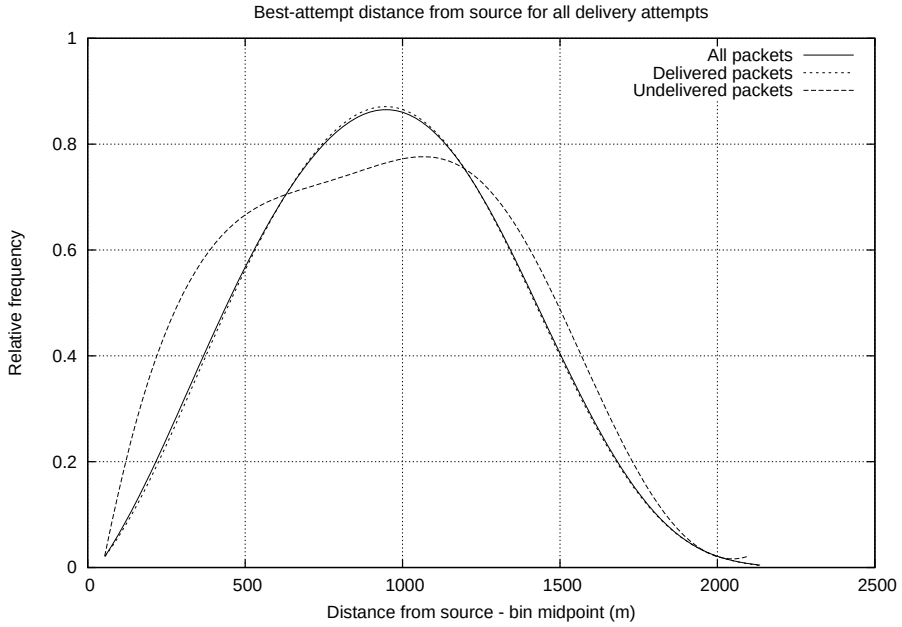
In this section we assess qualitatively the characteristic behaviour observed for packets generated by the simulated application. The behaviour demonstrated for all packets is compared against that demonstrated by those subsets which were delivered, and those not delivered. We assume that qualitative differences between the resulting distributions represent qualitative differences in behaviour of packets which do and do not reach their destination.

Accepting that some level of packet loss is inevitable in a wireless network not designed or optimised for reliability, ideally the loss probability would be equal for all packets as the simulated networks treat all packets as being of equal priority. Suppose some qualitative difference exists; this represents a weakness in the behaviour induced by a given protocol, which might be addressed by amending the protocol to deal with the specific cases in which this weakness becomes influential.

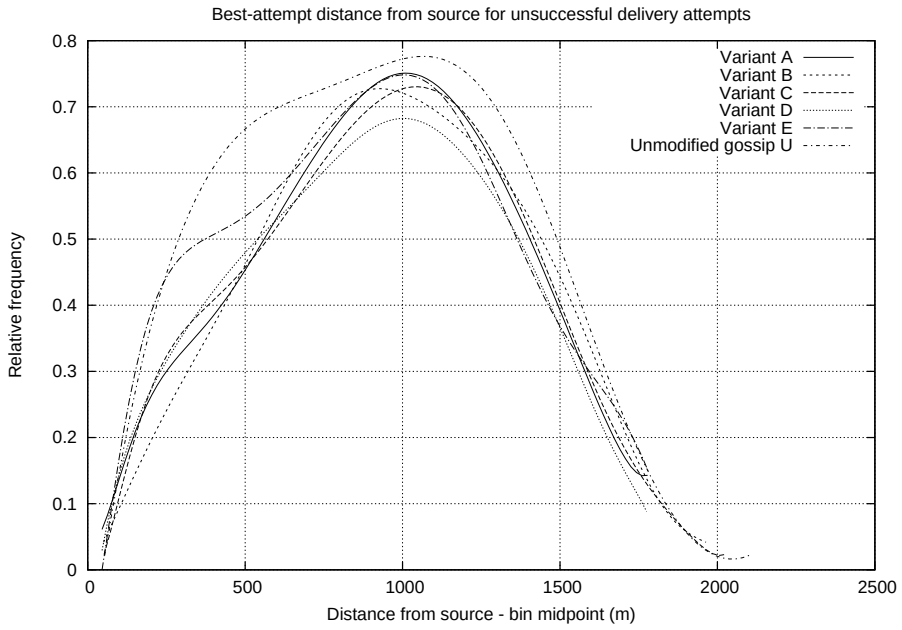
The key problematic behaviours associated with failed packet delivery attempts are *early fail* and *late fail*, corresponding to the bimodal behaviour inherent to gossiping protocols [116]. In *early fail* gossiping dies out early, wasting few resources, and stifling the delivery attempt before it becomes established. In *late fail* the packet covers most of the network and yet fails to reach the destination, thereby wasting all resources invested in the attempt.

In figures 3.11(a)-3.15(a) it is evident that for all metrics the probability distribution function for all packets is almost identical to that for delivered packets. This is unsurprising as around 92% of packets were successfully delivered in each case (see section 3.4.1). Furthermore, the distribution function for undelivered packets is generally similar to that of all or delivered packets, albeit with some difference in overall shape. It is these differences in shape we shall examine here. The relative height of features in the plots is not significant, as all have been scaled in the y-axis such that discrimination of individual plots is possible.

Figure 3.11(a) shows the distribution of relative frequency of distances to observe a good approximation of a Gaussian distribution. This is an expected geometric consequence of randomly selecting source and destination nodes for traffic packets which are distributed randomly and evenly throughout a cube. The distribution pertaining to undelivered packets more roughly approximates a Gaussian distribution with similar mean, but with the distinct asymmetry that the falloff is much less pronounced below this mean



(a) All delivery attempts for unmodified gossip

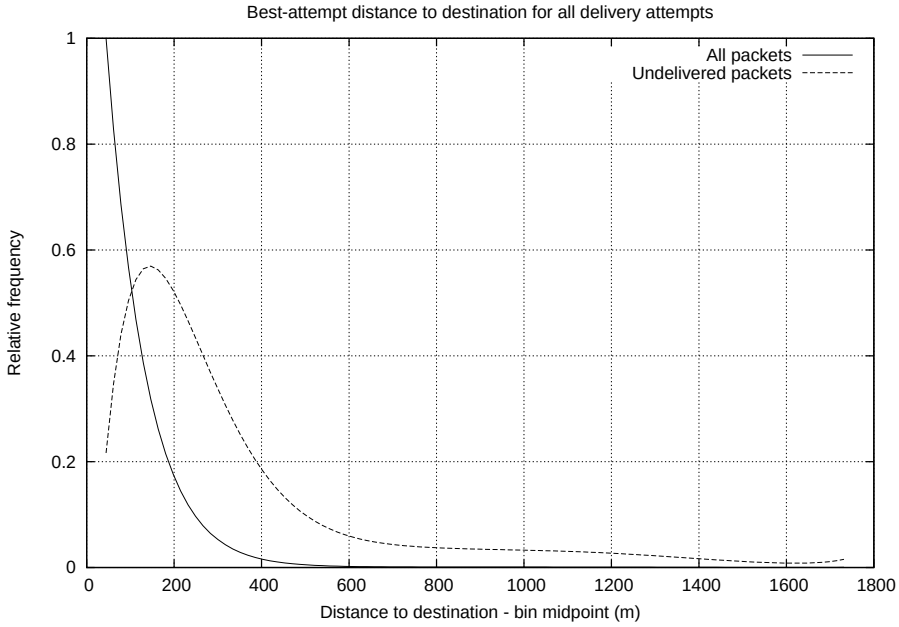


(b) Unsuccessful delivery attempts for gossip variants

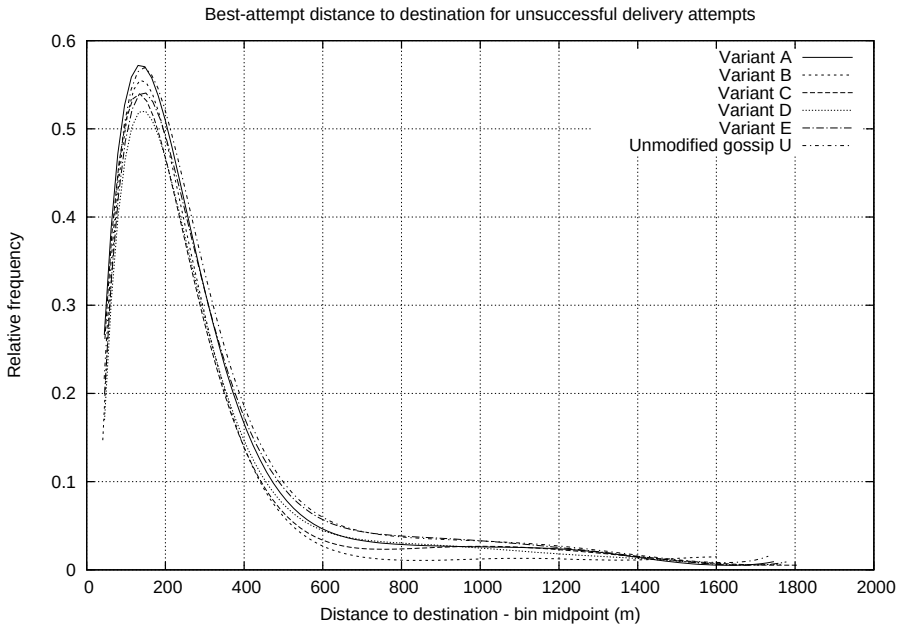
Figure 3.11: Source distance vs frequency

than above. This shows that undelivered packets are substantially less likely to get close to their intended destination than delivered packets, providing evidence of the *early fail* of gossip message propagation [116].

Figure 3.12(a) does not show the frequency distribution for delivered packets, as this is trivially zero at all places except at the origin where a Dirac delta would represent



(a) All delivery attempts for unmodified gossip



(b) Unsuccessful delivery attempts for gossip variants

Figure 3.12: Destination distance vs frequency

all delivered packets. In the distribution plot for all packets we find further evidence of bimodal behaviour in the hyperbolic curve; any given packet tends to cover almost all of the network as $p >= p_c$ in the experiment design and hence reaches nodes close to the intended destination.

The plot for undelivered packets tells much the same story, although there is a sharp

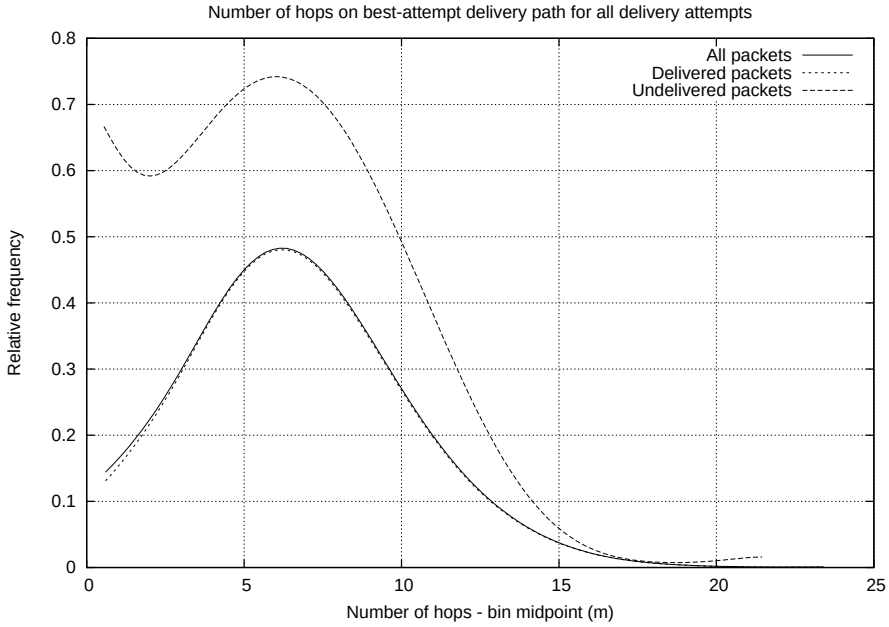
falloff in frequency at the y-axis due to the discrete categorisation of packets as *delivered* or *undelivered*. Even undelivered packets tend to reach nodes close to the destination, representing wasteful employment of network resources; if a delivery attempt is to fail, it is better that failure occurs sooner rather than later, such that substantial energy and time need not be wasted when the packet falls just short of the target. This is the *late fail* effect discussed earlier.

Figure 3.13(a) indicates that the path from the source node to whichever node of those receiving the packet is closest to the destination is far more likely to consist of a very small number of node-to-node hops for undelivered packets than for delivered packets. Of course, some source-destination node pairs happen to be located in close physical proximity; it would be expected here that the best internode route would contain few hops. More generally, however, we see further evidence of *early fail*.

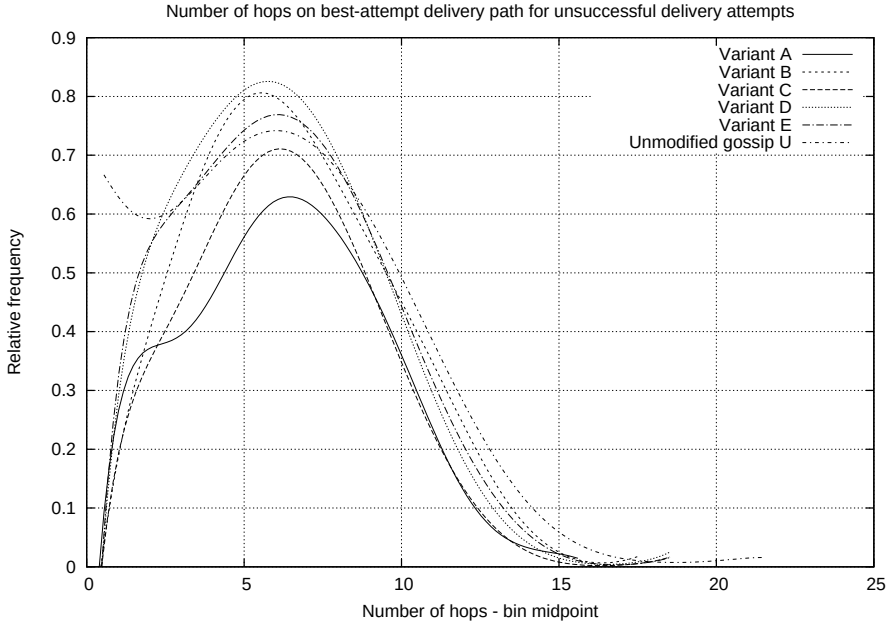
In figure 3.14(a) we see a similar effect to that illustrated in figure 3.13(a) when we consider time since delivery attempts began, although the qualitative difference between the distributions for delivered and undelivered packets is less pronounced. Nevertheless, these plots indicate that a time-based approach might offer a similar remedy to *early fail* to that offered by a hopcount-based approach.

Figure 3.15(a) is slightly different to the others considered in this section. The plots represent a heuristic measure of energy consumed per packet for successful and unsuccessful delivery attempts, working from the assumption that each node-to-node hope consumes a roughly comparable quantity of energy available within the network. Again we see evidence of *early fail* behaviour. Under the unmodified gossip protocol we expect a packet to cover almost all of the network or very little of the network, irrespective of the physical or logical proximity of the source and destination nodes.

If we assume that each node will rebroadcast a given packet only once then total network coverage in a network of n nodes would involve $n - 1$ hops. A substantial area under the undelivered packets curve toward the left of figure 3.15(a) indicates a substantial number of packets for which very few hops are counted, indicating little of the network was exposed to the packet; unless by chance the destination happens to be physically close to the source this renders unlikely the successful delivery of the packet.



(a) All delivery attempts for unmodified gossip

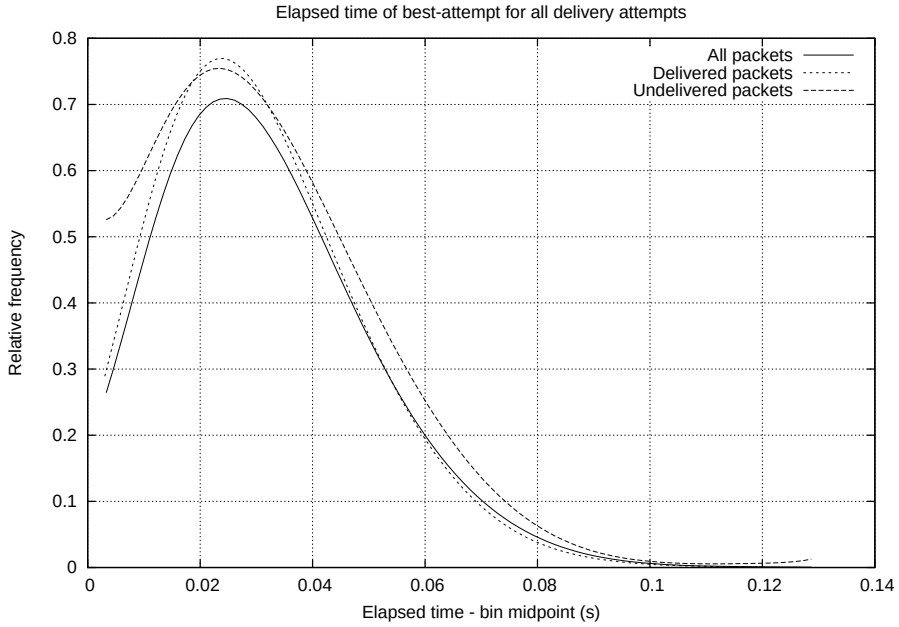


(b) Unsuccessful delivery attempts for gossip variants

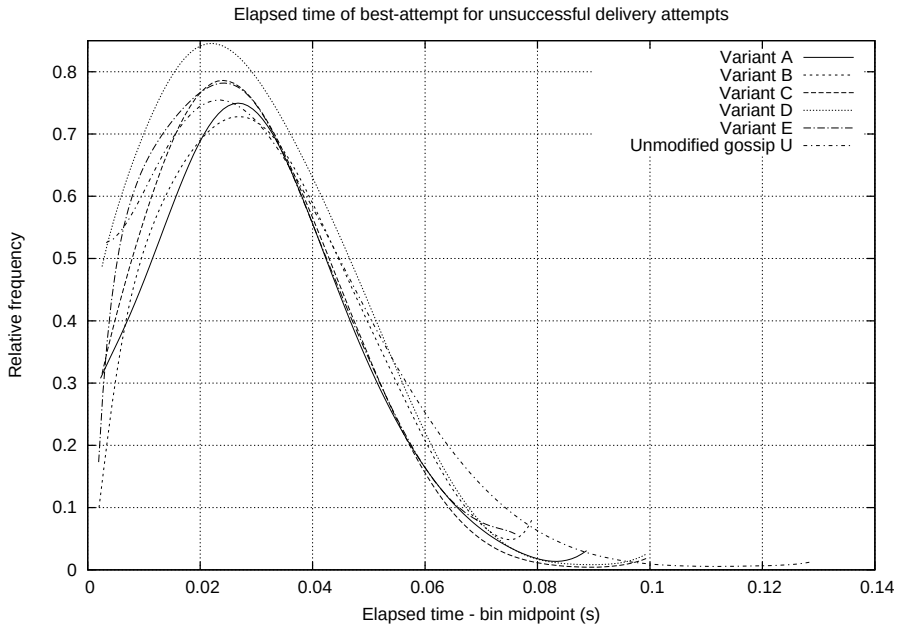
Figure 3.13: Best-path hops vs frequency

3.5.2 Distribution analysis: modified protocol variants

Percolation theory [116] predicts that flooding protocols will exhibit a bimodal behaviour transition as p increases such that either most nodes receive a packet, or very few do, along multi-hop delivery paths. Increasing network size should yield a sharper transition, which is predicted to occur at around $p \approx 0.6$. As a corollary, flooding protocol variants which



(a) All delivery attempts for unmodified gossip

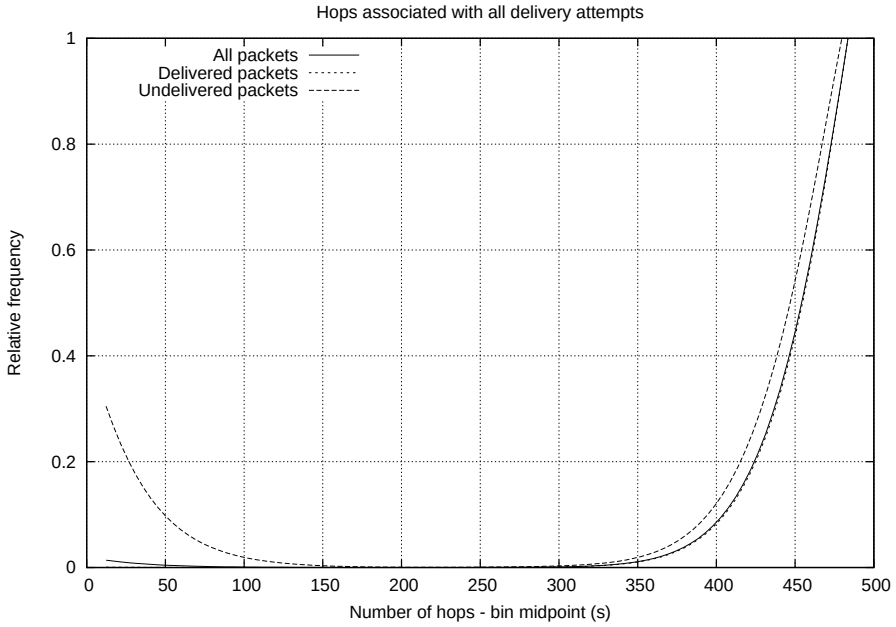


(b) Unsuccessful delivery attempts for gossip variants

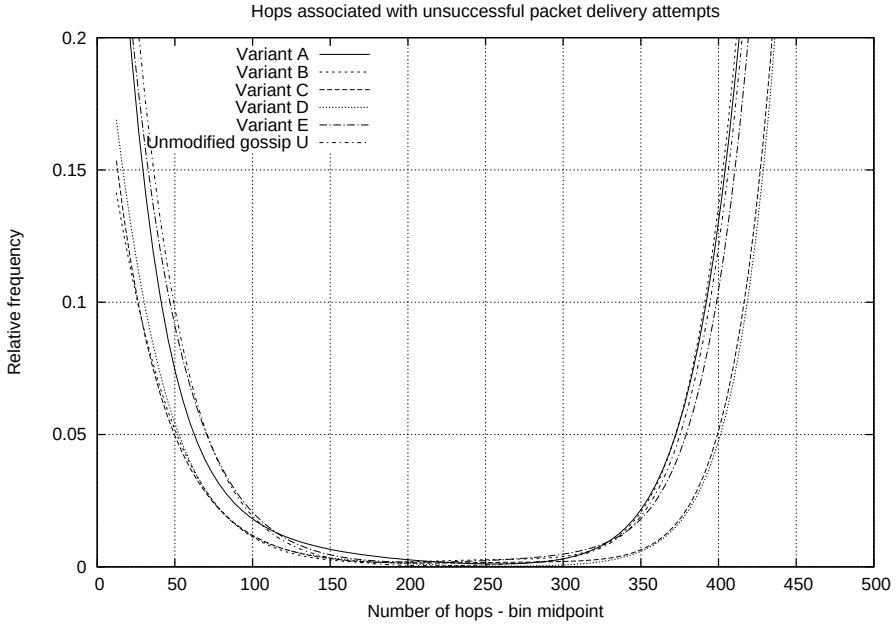
Figure 3.14: Best-path time vs frequency

implement dynamic protocol tuning by modulating rebroadcast probability dependent on network or traffic context are unlikely to offer any significant advantage over the standard version. We confirmed this experimentally by deploying a set of flooding derivative protocols into the same network configurations explored in section 3.5.1.

Having established and measured the undesirable characteristics of the unmodified



(a) All delivery attempts for unmodified gossip



(b) Unsuccessful delivery attempts for gossip variants

Figure 3.15: Total hops vs frequency

gossip protocol associated with failed packet delivery attempts, we now attempt to attack these weaknesses. We modify the gossip protocol to yield the following variants in which online heuristics are applied, at each node and for each packet individually, to modify the effective rebroadcast probability p_e from the base probability p under certain circumstances.

The underlying concept is that we might improve network performance by encouraging rebroadcast in nodes or regions which are likely to form part of good delivery paths, and discourage rebroadcast where it is unlikely to make a net positive contribution. Nodes do not have complete knowledge of the network state or future traffic production, so an optimal algorithm is not possible; however, heuristic approaches might offer useful improvement approaching theoretical bounds.

Our trials employed previously published flooding derivatives, as well as defining a further two variants based on similar principles. Each derivative protocol attempts to combat the *early fail* or *late fail* failure modes [116].

The gossip variants considered are:

- U: original unmodified gossip
- A: distance from source + elliptical bounding
- B: hops traversed + elliptical bounding
- C: time since start + elliptical bounding
- D: distance to destination + elliptical bounding
- E: elliptical bounding regions only

Variant U is the unmodified gossip protocol against which the variants are compared. Variant E implements the elliptical bounding of Li *et al.* [183], by setting $p_e = 0$ for nodes beyond a bounding sphere whose diameter is defined by the packet's source and destination nodes, to prevent packets "spilling out" into positions unlikely to contribute to successful delivery.

Variants $A-D$ also employ the elliptical bounding of E but augment this by increasing p_e under certain circumstances. For packets within the elliptical bounding region, each variant N of $A - D$ calculates p_e in the interval $[p, p_{max}]$ as a function of some packet attribute, where that attribute value falls between zero and some threshold value t_N. The value of t_N for each variant $A - D$ was obtained from the graphs of section 3.5.1 by observing the section(s) of the metric on the x-axis. These show dissimilarity in the relative frequency of observations for delivered and undelivered packets. Variant E always has a unitless threshold value of 1, the relative length of source-to-intermediate and source-to-destination vectors, and hence need not be derived through measurement.

Any number of functions might be selected, such as the *step function* or various types of exponential or trigonometric functions, but we select a simple *linear ramp function.* The latter requires few resources for calculation, and is a reasonable approximation of

numerous functions of greater computational cost. We define $p_{max} = 1$ to maximise the effect of the rebroadcast probability increase, although any $p_{max} \in [p, 1]$ can be used.

Variants $A - C$ boost p_e for packets early in attempted delivery to combat *early fail*. It would be possible to employ a compound heuristic using more than one of $A - C$, but this would be unlikely to yield much additional improvement as each addresses a similar problem.

Variant A is similar to *GeRaF* [347], and boosts p_e on the *geometric distance from the source location to the current packet location* with $t_A = 750m$. Variant B is similar to *GOSSIP1* [116] and *PPSNRP* [29], boosting p_e on the *number of hops traversed from the source node to the current node* with $t_B = 3$ *hops*. Variant C boosts p_e on the *time elapsed since delivery attempts began*, a novel approach not previously considered in the literature to the best of our knowledge, with $t_C = 0.03s$.

Variant D boosts p_e for packets late in attempted delivery to combat *late fail* where significant resources have already been invested in lost packets, an approach similar to *PPSNRP* [29] but replacing network distance with geometric distance similar to *GeRaF* [347]. Variant D boosts p_e on the *geometric distance from the current packet location to the destination location* with $d_D = 500m$.

In all cases the shapes of curves for variants $A-E$ were similar to those of U, suggesting that the pattern of network behaviour was qualitatively unchanged. However, variants $A - E$ generally showed improved behaviour in some metrics. In figure 3.12(b) there is little difference in any of the plots, suggesting that the gossip variants are more able to reduce *early fail* effects than *late fail*.

In figures 3.11(b), 3.13(b), and to a lesser extend 3.14(b), we see variant E performs somewhat better than the unmodified protocol U, and that variants $A - D$ perform somewhat better than E but broadly comparable with other variants $A - D$. This suggests some success in combating the *early fail* problem. However, figure 3.15(b) shows little variation between any plots, and little success in reducing the relative frequency for low-hop delivery attempts for undelivered packets to the low frequencies for successful delivery attempts illustrated in figure 3.15(a).

We therefore conclude that variants $A - E$ have some small influence in reducing the *early fail* problem, but insufficient influence to solve the problem completely. Further improvement may require protocols of greater complexity or which are at least partially stateful, for example maintaining local knowledge of successful and unsuccessful delivery

Metric	U	A	B	C	D	E
M1	*1.000*	0.975	0.962	0.957	0.962	0.987
M2	*1.000*	0.991	1.000	1.002	1.009	0.993
M3	*1.000*	1.012	1.000	1.014	0.999	1.010

Table 3.1: Normalised metrics for all variants

attempts to exploit in relay selection decisions.

3.5.3 Quantitative performance analysis

We now assess the relative merit of variants A-E against unmodified gossip U quantitatively. Metrics $M_1 - M_3$, as defined in section 3.3, are shown normalised against the value obtained for the unmodified gossip protocol configuration U in table 3.1. These metrics pertain to all network behaviour throughout network lifetime, and as such are suitable for analysing relative performance but are not suitable as online heuristics.

The values displayed in table 3.1 are unitless as they indicate the relative magnitude in comparison to the unmodified gossip protocol, rather than indicating the measured value itself. All figures are given to three decimal places. For some metric M_n, a measured value for some protocol variant v in A-E is given by M_{nv}. Where $M_{nv} < M_n$ the variant v reduced the metric, where $M_{nv} > M_n$ the variant v increased the metric, and where $M_{nv} = M_n$ the variant v had no effect on the metric. For metrics $M_1 - M_3$ higher values M_{nv} are desirable, whereas for $M_4 - M_5$ lower values are desirable.

Looking at table 3.1, it is immediately obvious that there is very little variation across the protocol variants in any of the metrics. This supports the findings of section 3.5.2 in which very little difference in the respective approximated probability distribution functions was observed. The simple context-sensitive heuristic-driven variants of the gossip protocol considered in section 3.5 do not perform appreciably better than the simpler, plain gossip protocol.

The variants also do not appear to harm network performance; if a given variant were found to confer some material advantage for some specialised traffic flow type in a given deployment context, it would seem entirely feasible to use this variant with reasonable confidence that more general traffic flows would not be affected adversely. However, this seems insufficient evidence of the merit of such a variant in the general case.

In the absence of demonstrable, meaningful performance improvement we conclude

that the additional complexity of the variant protocols cannot be justified, either in terms of the additional computational complexity at sparingly-resourced nodes in executing these protocols or in terms of analytical effort and design understandability.

An alternative approach might simply be to send multiple copies. If interpacket delay is sufficient to guarantee independence of the progress of n send attempts, and the probability of a given attempt succeeding is q, the probability of both attempts failing is $(1-q)^n$ which diminishes quickly as the nth power. If approximately 92% of delivery attempts succeed (see section 3.4.1) then just two send attempts should reduce the probability of both failing to $(1-0.92)^2 = 6.04 \times 10^{-3}$. However, this relies on the probability of successful delivery attempts being entirely independent. This assumption may not hold in sensornets; for example, network congestion associated with an initial attempt may render subsequent attempts less likely to succeed.

3.6 Summary

It has been shown that optimising even a single tunable parameter is a delicate balancing act with the results being highly dependent on the specific network and application. As such this form of investigation is a valid and valuable use of sensornet designers' time. Further gains may be achievable by tuning a given protocol for a specific application and network configuration.

Several undesirable behaviours relating to undelivered traffic were observed and measured. A number of areas for improvement were identified. Existing protocols were improved by defining variants which allow online adaptation at individual packet granularity to address these behaviours. The requirement for online adaptation is particularly acute at the beginning and end of delivery paths.

However, the observed performance improvement was not significant, suggesting that network designers should instead focus on tuning protocols to better fit the network problem, or tune the network design to better fit the selected protocol.

Chapter 4

Engineering methods for sensornet protocol tuning

The published literature suffers no shortage of network protocols of varying complexity, specialisation and scope. Deciding which protocol to select is difficult as there is generally no baseline against which to implement a fair and meaningful comparison. Good software engineering practice suggests the reuse of existing, well-tested protocols is generally preferable to the creation of new custom protocols for each application. However, the literature in which new protocols are presented frequently lacks any discussion of how the network designer might select appropriate values to assign to tunable parameters to obtain good performance. This chapter presents solutions to the protocol tuning problem based on principled search techniques.

4.1 The protocol tuning problem

A common strategy is to design new protocols for a specific intended deployment context [234]. We believe this strategy is flawed. Real-world deployment environments are usually not known precisely prior to deployment, and often change after deployment. Custom protocols often lack generality and become unusable if the expected and actual deployment environments differ, and may have unexpected behaviours or failure modes which would have been previously observed in more commonly used protocols.

We propose that network designers first attempt to fit existing protocols to their needs, only moving on to custom protocols if this should prove impossible. We describe a reusable engineering method to address this multi-dimensional optimisation problem, based on

sound principles widely recognised and applied beyond Computer Science. We provide a mechanism with which to de-risk deployment of sensornets tuned within training environments, evaluating the robustness of these tunings to changing environments. The mechanism is also useful for comparative evaluation of protocols within a fixed deployment context.

Most non-trivial protocols have sets of parameters which can be fine-tuned. One option is to introduce new energy aware networking protocols which seek to reduce energy consumption by managing network traffic more efficiently. However without suitable parameters the benefits can not be fully realised. In this chapter we explore how existing well-understood protocols can be optimised. The tuning method we describe is equally applicable to both energy-ignorant and energy-aware protocols. We demonstrate its efficacy and versatility by applying it to two fundamentally different protocols: *TTL-Bounded GOSSIP* (TBG) [217] and *Implicit Geographic Forwarding* (IGF) [120].

Sensornet designers must identify the most significant factors to avoid being swamped by unnecessary detail. Unfortunately, even identifying the relative importance of factors and their interactions is rarely trivial [93]. Discovering the best values to assign to these factors and understanding their impact on network behaviour tradeoffs is harder still, as discussed in chapter 3. Tunable parameters are often defined without clear default values and may be defined over an infinite range. Complex system design *can present a range of design options and subsystem choices that a designer unfamiliar in the art would find difficult to navigate* [82].

Where multiple controllable factors exist, each of which can take many values, combinatorial explosion renders exhaustive exploration impossible. Designers may resort to inefficient trial-and-improvement techniques or accept sub-optimal tunings. This is undesirable for numerous reasons including high cost, risk of low solution quality, and poor repeatability. Pragmatic engineering approaches can, in contrast, approximate exhaustive exploration in bounded time yielding repeatable results of known quality. Although generally desirable, this is particularly significant in organisations which implement a well-defined engineering process such as CMMI [56] certified against standards such as ISO 9001 [141].

We propose a principled search method based on *full factorial design* experiments [22] which addresses the process weaknesses identified above. If an *iterative design* methodology is employed, this method could form a useful and repeatable component of end-of-

iteration evaluation [164]. Our method is not specific to any given protocol type, and is sufficiently generic to remain applicable for any similar sensornet protocol tuning problem.

Considering one factor at a time is unlikely to give a thorough treatment of the tradeoffs and hence any solution found may differ significantly from the optimal. For instance, suppose we wish to maximise the probability that a packet reaches a given destination within a given deadline. If we increase the *gossip probability* are we helping timely delivery by ensuring that at least one short path from source to destination is traversed, or are we hindering timely delivery by congesting the network? How big should each node's *waiting packet queue size* be to avoid retaining packets that will inevitably expire prior to delivery but without throwing away packets which should still be viable? Does increasing the *gossip probability* upset this delicate balance by delivering too many irrelevant packets, undermining careful tuning of the *waiting packet queue size*?

Tackling this issue presents interesting challenges. Sensornet protocol tuning is not a simple, idealised problem. It is a complex real-world problem with multiple inputs, multiple outputs, and multiple objectives. The non-trivial interrelationships between these factors are not known at the outset, and thus cannot be targeted specifically during experiment design. Two important engineering challenges addressed by our method are the production of solutions exhibiting robustness to deployment context, and the uncertainty and noise inherent in any experimental data.

4.1.1 Description of the problem

Layered network protocol stacks, such as the OSI [343] and TCP/IP [79] models, partition the network functionality into a set of layers between the *physical layer* and the *application layer*. Within each layer the behaviour of network elements is defined and managed by one or more protocols. Whereas the behaviour of any layer may indirectly influence any other layer, each layer interacts only with the layer immediately above, and the layer immediately below, in the stack. This interaction occurs through well-defined interfaces exposed for this purpose, and allows the network designer to consider any given layer in isolation.

In this chapter we consider protocols functioning at the *Network layer* of the OSI layered model [343]. A large number of network routing protocols exist, some of which may be more suitable than others to a proposed sensornet deployment. Most of these protocols can be tuned by a network designer in an attempt to attain at least the minimum required

QoS, or to minimise or maximise one or more measurable performance attributes. Some tunable parameters are specific to a given protocol, whereas others are shared by several.

We define a combination of protocol and protocol tuning as an *acceptable solution* if all QoS requirements are achieved for the intended deployment configuration. It is possible that a given selection of protocol and protocol tuning will function more effectively than others in a given deployment environment. It is also possible that a given selection of protocol and protocol tuning will perform sufficiently well to be considered an *acceptable solution* across a broader range of possible deployment scenarios than others. This is of particular interest to network designers targeting real-world deployments as the real world is notoriously difficult to model in a manner which effectively captures the relevant attributes precisely or accurately [258]. It is also subject to unpredictable changes both before and after initial network deployment.

It is possible that zero *acceptable solutions* exist, or exactly one, or more than one. If zero *acceptable solutions* exist then no protocol tuning will achieve the required QoS. If exactly one *acceptable solution* exists then this must be selected. If more than one *acceptable solution* exists then any of these would provide the basic minimum required network effectiveness, but nevertheless differences in observed network capability may exist. Furthermore, it is possible that non-point regions of parameter space surrounding good solutions yield network behaviour measured within experimental error of the actual solutions.

It is likely that, if at least one *acceptable solution* exists, then many such *acceptable solutions* exist; this is particularly evident where one or more controllable factors is defined for continuous rather than discrete ranges. Given a number of *acceptable solutions*, how does the network designer decide which to use? We propose that network designers should select from the subset of *acceptable solutions* which offers an appropriate level of robustness to variation in deployment environment, as well as maximising any other desirable network characteristics appropriate to the application. This engineering approach yields a network design with an appropriate level of *tolerance* to unknown and unpredictable factors, in addition to maximising the potential utility of the network [295].

4.2 Tuning protocols by a Design Of Experiments approach

In this section we consider the experimental method employed to tune protocols to a given training environment. In the experiments described in section 4.2.1 we measure

network response metrics separately for each of three network sizes defined in section 4.3.1. Statistical models are fitted to the results using the methods described in section 4.2.7, from which we derive best-compromise protocol tunings for *TBG* and *IGF*. We then measure these network response metrics again for a longer, more detailed simulation of the same networks to yield estimated best performance measures attainable by each protocol in the tuning environment. Finally, in section 6.1 we measure these network response metrics for these best-compromise protocol tunings to assess robustness to a changing deployment configuration.

4.2.1 Three-phase Design of Experiments approach

Full factorial design [22,329] is used to systematically explore the factor-response relationship. This approach gives broad but shallow coverage of all possible combinations of all acceptable ranges of controlled factors. Unlike *Monte Carlo simulation* methods [199] the set of inputs is not randomly selected, but is instead defined so as to obtain a uniform sampling of the parameter landscape. We address the combinatorial explosion identified in section 4.2.8 by applying a three-phase method designed to avoid wasting resources and analytical effort on matters which will not significantly influence the outcome, allowing more detailed statistical models to be derived for a given cost.

The three phases are:

1. *Phase 1:* Determine the point at which measurement experiments are sufficiently stable to be sampled as representative of long-term stable behaviour.

2. *Phase 2:* Sample the problem space at low resolution. Statistical models are fitted to the resulting data points to identify insignificant factors, to be dropped in Phase 3 to make high resolution modelling tractable.

3. *Phase 3:* Sample the problem space at high resolution for significant factors only. Fit statistical models to the resulting data points to confirm the significance of selected factors, and summarise the complex interrelationships in a format suitable for further analysis. Extract protocol tunings associated with desired network behaviour.

We apply statistical methods to determine the significance and influence of each controlled factor to each response value. If a given factor is not correlated with the response we conclude this factor-response pair is independent. If all factor-response pairs are uncorrelated for a given factor, we conclude that this factor has no statistically significant

influence. If a factor has no significant influence on any response then it can safely be ignored. A full cost analysis of this experimental method can be found in section 4.2.8.

Every model is necessarily an imperfect abstraction of reality. Factor significance within the model implies, but does not guarantee, significance in reality. The quality of the model determines the extent to which results demonstrated within the model are valid. If a poor quality model is utilised it is possible that the supposed significance of a factor is an artefact of the model. We therefore employ a *pessimistic* approach, in which we retain factors which may later transpire to be statistically insignificant.

4.2.2 Phase 1: Analyse variance

The most computationally expensive component of the three-phase method is the evaluation of candidate solutions through simulation. Reducing simulation length or detail reduces this overhead. However, if taken too far this will risk experimental errors of unacceptable magnitude, leading to meaningless and unusable results. Phase 1 analyses variance of measured metrics with respect to simulated time and finds the minimum acceptable simulation time required to obtain results with experimental error no greater than a defined threshold. We set this threshold at $\pm 5\%$.

Owing to the stochastic nature of network communication and unpredictable radio environments, metrics are not guaranteed to be identical between repeats of the same experimental configuration. However, where network protocols converge on steady state behaviour, experiments can run until performance metrics converge within some acceptable range of their limiting value; experimental noise explains further variance within this range.

We measured the minimum time required for each metric under each protocol by experiment in networks of the design defined in section 4.3.1. Metrics are sampled periodically but are influenced by total simulated period from the start to the sampling point. Assuming that the network eventually reaches a steady state, measured metrics converge on the actual value with sample accuracy increasing with simulated time, until sampled values fall within experimental error margin at which point no further meaningful improvement is possible. At this point, metrics are sampled as representative of the stable value [18].

Assume the value of some convergent metric M_α at simulated time τ is given by $M_\alpha(\tau)$. $M_\alpha(\tau)$ approaches its limiting value $M_\alpha(\infty)$ as $\tau \to \infty$. At some simulated time τ_α the value $M_\alpha(\tau_\alpha)$ becomes sufficiently close to $M_\alpha(\infty)$ such that for all $\tau > \tau_\alpha$ the value $M_\alpha(\tau)$ is within $\pm\eta\%$ of $M_\alpha(\infty)$. We define metric M_α as *converged* at this simulated time τ_α.

Any further variation, including that deriving from noise and unblocked nuisance factors, is within $\pm\eta\%$ experimental error margin. We set $\eta = 5$ such that measured metrics used in later analysis have $\pm 5\%$ measurement error.

4.2.3 Phase 2: Identify significant factors

In section 3.2 we define tunable parameters for two candidate protocols, TBG and IGF, and the acceptable ranges for these parameters. A given protocol or set of protocols may have an arbitrary number of tunable parameters, some of which may be more significant than others. In Phase 2 we determine which parameters and parameter pairs are the most important predictors of network performance metrics, using the well-known ANOVA method [38]. We identify the minimal set which are significant at a given threshold confidence level; all other parameters can be discarded to reduce the problem size.

Recall from section 3.3.4 the notion that a mapping $S \mapsto T$ exists, though is not known *a priori*, which defines for any given protocol tuning the expected values of metrics that measure induced network behaviour. As this mapping contains an infinite number of elements where one or more controlled factors is continually variable we cannot evaluate the entire mapping; instead, we evaluate a carefully designed partial mapping of finite size.

We define a test case suite C by multi-level Full Factorial Design [18]. We sample each of the p controlled factors defined in section 3.2 at q evenly-spaced points, giving for each controlled factor a set of candidate values. If the experimenter happens to know that good quality solutions are likely to be found around specific regions of the parameter space, they can of course relax the requirement for evenly-spaced points and instead define a greater density of sampling points around known-good regions. We do not assume prior knowledge of the problem, and hence apply an even sampling strategy.

We then compose the test case suite by defining every valid combination of every candidate value of every controlled factor. A valid combination contains exactly one value for each controlled factor, where that value is taken from the set of candidate values for that controlled factor. Each combination represents a single candidate solution $C_\alpha \in C$ such that C samples the controlled factor design space at q^p points.

For each candidate solution $C_\alpha \in C$ we run a simulation experiment to measure the network behaviour under that protocol tuning. The resulting pairs of sample points and simulation-derived metrics represent exact solutions to specific known points in the gener-

alised model of the relationship between controlled factors and output metrics. However, these are not directly usable if we wish to know the relationship between input and output, or vice-versa, for other points in the input-output phase space.

For a finite set of sample points there exists the risk that an ideal value of some controlled factor falls between sample points, and hence is not evaluated directly. To consider points in the parameter space that have not been measured directly we need to interpolate by fitting a statistical model to the known sampled points to derive a set of equations describing a hypersurface in the phase space [38]. We then work with the fitted surface rather than specific individual experimental results. Increasing the per-parameter sample number q yields a fitted model which is a better approximation of the real relationship by providing more data for model fitting.

For each metric M_i a separate statistical model is derived in which n axes represents controlled factors X_1-X_n and a further axis in which the height of the hypersurface varies with the values of the output metric M_i. Axes corresponding to controlled factors X_1-X_n are common to all metrics M_1-M_m so a more complex surface can represent the interrelationships between all factors and all metrics. The model fitting procedure is described fully in section 4.2.7.

Although these fitted models define the relationship between controlled factor and induced network behaviour, we are not interested in this matter in Phase 2. Instead, we are interested only in the p-values [38] which define the confidence level for the contribution of each factor (and combination of interacting factors) to the predicted value of network metrics.

Where a given factor is found to contribute to the predicted value of one or more metrics with $p < 0.05$ this factor is significant with 95% confidence. All factors considered significant for at least one metric, in isolation or as part of an interaction pair, are retained for Phase 3; all others are dropped as their contribution is not statistically significant. Depending on the application requirements the experimenter may select higher or lower confidence levels, but 95% is generally a good choice unless there are specific reasons to favour an alternative confidence level [38].

The policy for dropping factors is a *pessimistic* policy. We err on the side of caution, and hence may retain factors which later transpire to be statistically insignificant. However, this pessimism is necessary if we are to define a mechanism that can be implemented mechanistically; we prioritise reliability over efficiency. When implementing this mecha-

nism by hand, the experimenter may choose to cull borderline factors where appropriate. We do not implement any such intuition-driven strategy here.

4.2.4 Phase 3: Extract protocol tunings

In Phase 2 we are interested only in identifying the significant factors, discarding the specific values of fitted model coefficients and the corresponding protocol tunings. We must do this to make high-resolution sampling of the problem space feasible, without wasting significant experimental effort on assessing irrelevant factors. Otherwise, we must either accept a lower than optimal sampling resolution, with the risk that no sampling point lies near optimal solutions, or expend considerably greater experimental effort than is strictly necessary.

In Phase 3 we are interested in the fitted model coefficients and specific protocol tunings. Having identified which factors have a statistically significant influence on measured responses, we can justifiably focus our experimental effort solely on tuning those significant factors. A set of experiments is defined and implemented in the same format as described in section 4.2.3 for Phase 2. However, in Phase 3 we increase q to sample the search range of each significant factor at a greater number of points; again, sampling points are distributed evenly throughout the search range. All insignificant factors are set to a reasonable default value; we select the midpoint of the search ranges.

We thus reduce the number of controlled factors, p, considered in the search, but increase the sampling resolution specified by q. As the cost analysis given in section 4.2.8 shows, the experiment cost grows polynomially in q but exponentially in p. It follows that for a fixed maximum boundary on experimental cost, it is possible to increase sampling resolution significantly by dropping a relatively small number of controlled factors.

Having run the experiments corresponding to this higher resolution sampling of the problem space, we now have a higher resolution partial mapping to approximate the mapping $S \mapsto T$. We may wish to use this partial mapping to extract high-quality protocol tunings associated with desirable network behaviour, or we may wish to use it to examine the generalised relationship between protocol tuning and network behaviour.

To extract high-quality protocol tunings, we calculate the solution quality E for each member of the mapping, using the method given in section 3.3.4. We select the candidate solution S_α associated with the best solution quality measurement. Experimental noise may, of course, award an undeservedly high quality rating to a candidate solution in a

single experiment instance. We mitigate this risk by aggregating the measurements for multiple experiment instances of each candidate solution, where we consider r similar but non-identical networks, and s repetitions of each combination of candidate solution and evaluation network [18]. Each of the s repetitions is perfectly repeatable, as evaluation occurs within simulation, but uses a different pseudo-random number generator seed to drive stochastic behaviour.

To confirm the significance of controlled factors selected in Phase 2, we repeat the model fitting and factor significance analysis described in section 4.2.3 using the high-resolution sampling of the problem space. The fitted models approximate the relationship between controlled factors and measured responses, and thus can predict likely network performance for any arbitrary set of input values. This is useful when exploring *what-if* scenarios. To interpolate in the parameter space between directly sampled values, the experimenter can find the set of controlled factor values which minimises the E metric defined in section 3.3.4, substituting the fitted models for each metric M_α into the formula for E.

An alternative use is to identify suitable regions, rather than single values, of the parameter space. Experimenters define sections of the multi-response hypersurface conformant to the required behaviours. Solving the simultaneous equations of the fitted model for these regions yields a set of inequalities which define usable regions of the controlled factor space. This is useful where protocol tuning is not the only design constraint; any protocol tuning which satisfies the inequalities will perform acceptably, if not optimally.

Experimenters interested in understanding the generalised relationship between significant factors and measurable responses can fit statistical models to the sampled points. This yields a set of coefficients for the selected statistical model which summarise the relationship between factors and responses for the selected protocol and network configuration as a set of simultaneous equations. However, we are interested in extracting a single near-optimal protocol tuning. As we have a reasonably dense grid sampling of the parameter space we do not require the interpolation effect offered by model fitting, and can evaluate our experimental data directly.

4.2.5 Automation

The experimental method as described in sections 4.2.2-4.2.4 does not require or imply any subjective decisions or human intuition. It defines a sequence of repeatable steps to

be followed mechanistically, which can either be implemented by human experimenters or composed into an automated process. An automated process might be implemented as part of a defined Software Process for quality assurance purposes [141], or for the collection of metrics tracking progress against development project goals [56]. The method described here can be applied automatically at the end of each project phase alongside conventional unit, integration, regression and performance tests.

4.2.6 Simulation environment

It is generally impractical, if not impossible, to perform the experiments described in this chapter using real networks owing to high overheads of logistics, cost and time. It is generally impossible to guarantee a consistent and unchanging environment for the total runtime of the tens of thousands of experiments. This would severely undermine the validity of comparison between results obtained from multiple experiments, which is critical to the analytical methods we propose.

To address these concerns all experiments were conducted by simulation, using the multithreaded simulator called *YASS* which is optimised for sensornet experiments and for this duty pattern. Multiple independent simulation instances can be executed in parallel to take advantage of low-cost commodity hardware. Unnecessary computational overhead is avoided by combining lazy evaluation methods with post-hoc trace analysis [132]. The design of *YASS* is considered in [287], in which the simulator is validated against the type of problem considered in this chapter and found to be accurate.

The perfect repeatability of simulation experiments is a desirable property. Where experiments include some stochastic element, however, it is useful to consider different representative examples of random behaviour. This is achieved in our simulation experiments by using pseudo-random number generators to drive stochastic elements. In section 4.2.4 we describe the s repeats of each experimental configuration of candidate protocol tuning and evaluation network. For each of the s repeats we seed the pseudo-random number generator with a different value. This set of seeds is reused for the s repeats of each experimental configuration. This allows our simulation experiments to repeatably model network configurations running at different times, with the modelled world behaving slightly differently in each of these s instances.

The resulting efficiency was such that simulated time passed faster than wall time; results were obtained more quickly than real-world experiments could provide results,

even if unlimited resources were available. However, the approach presented here could be implemented with equivalent results in any sensornet in which protocol factors can be controlled and solution quality metrics measured.

4.2.7 Fitting statistical models to experimental data

Assume we define p controlled factors, and sample each at q evenly-spaced points. This sampling defines q^p design points, distributed evenly throughout the protocol configuration space. The factorial design experiments described in section 4.2 map each sampling point to a set of metrics. These pairs of sample points and simulation-derived metrics represent exact solutions to specific known points in the generalised model of the relationship between controlled factors and output metrics. However, these are not directly usable if we wish to know the relationship between input and output, or vice-versa, for other points in the input-output phase space.

To consider points in the parameter space not measured directly, we use interpolation. A statistical model is fitted to the known sampled points, to derive a set of equations describing a hypersurface in the phase space [270]. We then work with the fitted surface rather than specific individual experimental results. An appropriate statistical model must be selected, which yields a surface with shape similar to that which would be observed if an infinite number of sample points were used. Previous work [295] has shown that linear first-order interaction models are a suitable approximation for the protocols considered in this chapter; we confirmed this by examining the correlation coefficient between the measured and predicted values.

For each output metric under consideration, a separate statistical model of the form given in Equation 4.1 can be fitted to the result set. β_0 is a constant, X_i is the ith controlled factor value, β_i is the coefficient for controlled factor X_i, β_{ij} is the coefficient for the interaction between controlled factors X_i and X_j, and ε is the normally-distributed noise term. The response M_i is influenced linearly by each factor and each pairing of potentially interacting factors. Our analysis shows this model to be a good fit for the experimental results considered in this chapter.

$$M_\alpha = \beta_0 + \sum_{i=1}^{n} \beta_i X_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \beta_{ij} X_i X_j + \varepsilon \tag{4.1}$$

For each output metric M_1-M_m a separate linear interaction model is produced by Equation 4.1 in which a set of n axes represents controlled factors X_1-X_n and a further

axis in which the height of the hypersurface varies with the values of the output metric M_α. As the axes corresponding to controlled factors X_1-X_n are common to all metrics M_1-M_m it is possible to combine them all to yield a more complex surface representing the interrelationships between all controlled factors and all metrics.

Finding sets of values for controlled factors corresponding to solutions with appropriate characteristics is equivalent to identifying regions of the axes representing controlled factors X_1-X_n with appropriate fitted surface height in the axes corresponding to output metrics M_1-M_m. Similarly, finding optimal or worst-case sets of controlled factors is equivalent to finding minima and maxima of the fitted surface. This is implemented by solving sets of simultaneous inequalities when identifying regions with suitable characteristics, or by solving sets of simultaneous equations when addressing optimal or worst-case characteristics.

Interactions involving any number of controlled factors could be considered. However, in this thesis we use the fitted models to identify significant factors, rather than to predict network behaviours directly. It follows that identifying the statistical significance of interacting pairs is sufficient to identify factors which have significant influence on network behaviour, but would otherwise not appear to be significant when considered in isolation.

Experiment designers can also consider other models, such as higher order linear models, selecting the model offering the best fit to the dataset. For example, the quadratic model shown in Equation 4.2 includes all terms of Equation 4.1 with additional terms for squares of controlled factors. Additional terms can be added to consider ever higher degrees of controlled factors and their interactions. Experiment designers might also consider generalised linear models in which the M_α term of Equations 4.1 and 4.2 are replaced by $f(M_\alpha)$, where a better fit might be achieved by applying a transformation to the measured response. For example, we could consider taking the natural logarithm of the response by defining $f(M_\alpha) = \ln M_\alpha$.

$$M_\alpha = \beta_0 + \sum_{i=1}^{n} \beta_i X_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \beta_{ij} X_i X_j + \sum_{k=1}^{n} \beta_k X_k^2 + \varepsilon \tag{4.2}$$

Sampling the parameter space at more points yields a fitted model which is a better approximation of the real relationship by providing more data for the model fitting algorithm. The minimum acceptable number is generally defined by the order of the fitted model; for example, linear models require at least two sampling points in each controlled factor, whereas quadratic models require three, and so on. However, experimenters may

decide to use more sampling points to increase the accuracy of fitted models. The actual number selected may, for example, be the highest number that allows the entire test suite to complete within acceptable time, as considered in section 4.2.8.

For a finite set of sample points there exists the risk that an interesting feature of the solution landscape falls between sample points, and hence is not present in the fitted model. Interpolation allows every candidate parameter set to be considered implicitly and simultaneously, including those not measured directly, but there is a risk that the optimal solution lies between directly measured points and is not revealed in the fitted model.

However, the approximating behaviour of model fitting has the useful consequence of minimising the impact of noise in experimental data. In any set of stochastic simulations or experiments the existence of non-uniform experimental noise implies that some results will be less accurate than others. Previous work [295] shows the noise term ε is normally distributed for experiments of the type considered in this chapter. This implies that for a given simulation of a given sample point it is possible, but unlikely, that this single result has large error. As every data point has equal influence in the model fitting algorithm the influence of these outliers is minimal, the larger number of more accurate results exerts significantly greater influence.

In selecting the statistical model to fit to the experimental data points it is important to address the tradeoff between sufficient detail to capture the significant features of the parameter landscape and sufficient approximation to smooth out the illusory features resulting from experimental noise. This usually requires the fitted surface to capture the significant low-frequency detail and to discard the insignificant high-frequency detail. However, if it is known that a given region of the parameter landscape has meaningful high-frequency detail then a large number of sampling points can be considered in this region; there is no requirement that sampling points be distributed uniformly.

4.2.8 Cost analysis

Exhaustive exploration of the protocol configuration space defined in section 4.1.1 is impossible due to combinatorial explosion. This is a consequence of both the number of controlled factors and the number of values which each factor can take, the latter being infinite for continuously variable factors. Our method, based on full factorial design [22], samples the protocol configuration space at a finite set of points to render tractable the evaluation effort. Increasing the number of experimental configurations increases the qual-

ity of fitted statistical models, and hence solution quality, but also increases experiment cost. A balance must be found which obtains solutions of acceptable quality within acceptable wall time and experimental overhead.

Consider the algorithmic complexity of this approach. Assume we define p controlled factors and sample each at q evenly-spaced points, giving q^p design points distributed evenly throughout the parameter space. If we evaluate each design point for each of r networks then we define rq^p test configurations. We simulate each test configuration s times to prevent results being unduly influenced by any single simulation instance, yielding the requirement to run rsq^p test cases in total. Determining the isomorphism of two or more such factorial designs is generally NP-hard [192], but our method is designed so as not to require this step.

The test suite size grows in p, q, r and s, but in a qualitatively different manner. Linear growth in r and s is observed as the set of design points is repeated for each of r networks, and the set of test configurations is repeated s times without modification. Polynomial and exponential growth in q and p respectively are observed because the design matrix defining the design points set can be represented as unit cells within a hypercube. Increasing q increases the length of the hypercube sides, whereas increasing p increases the dimensionality of the hypercube. Test suite cost grows as $\mathrm{O}(n)$ in r and s, $\mathrm{O}(n^c)$ in q, and $\mathrm{O}(c^n)$ in p.

Although the cost is exponential in p, our experimental method addresses this potential problem. Firstly, for a given network protocol there are a finite number of controllable factors, only a subset of which are likely to be of interest or permit alteration by the network designer. This places a small, finite upper bound on p for a given protocol. Secondly, Phase 2 of our experiments implements a *screening* approach which further reduces p by identifying insignificant controllable factors which can safely be disregarded. It is therefore possible in Phase 3 to increase q after reducing p and still have the full experiment set complete in acceptable wall time.

The polynomial growth in q is also managed in the experiment design. Recall from section 4.2.7 that we fit linear interaction models to measured values. A linear relationship in one factor can be uniquely defined by just two factor-response pairs. Extending this to a linear relationship in p factors requires two values of each controlled factor to be represented in the set of design points [22]. We therefore require only that $q \geq 2$, with low and high values of each factor representing the range for which the model is required to

predict metric values. Higher values of q obtain better fitted models but with decreasing gains for each additional sampling point, so small values of q work well [22] and minimise simulation cost. Higher-order linear models of order d would require $q \geq d$.

As all simulations imply similar computational overheads we assume each simulation completes in approximately constant wall time, t. All simulations are mutually independent and can therefore be executed in parallel, reducing total runtime to that of a single simulation if sufficient processing hosts are available. Assume a multiprocessing environment in which $x \in \mathbb{N}$ independent simulations can execute in parallel. In factorial design test suites there are no dependencies between simulations so any number can execute in parallel, all at cost t. The total wall time cost is $C = \frac{rsq^p}{x}t$. Note that $C \propto \frac{1}{x}$, reaching a minimum of $C = t$ where $x = rsq^p$.

4.3 Comparing tunings in training and deployment networks

In this section we consider the network performance observed when a given protocol tuning is deployed into environments that are similar or dissimilar to the training environment. We label the *TBG* protocol as A and the *IGF* protocol as B to prevent the following text becoming unnecessarily cluttered.

4.3.1 Network design

The techniques outlined in this chapter are independent of the specific protocols and network designs explored in the following experiments. However, these experiments explore only a finite portion of the unbounded design space of all networks and all protocols. It is likely that the trends we identify in network performance responses as a function of protocol tuning parameters are likely to apply in similar networking contexts. Nevertheless, we limit the scope of our claims to the portion of design space defined in this section, within which we have confidence in our findings as they are demonstrated to have statistical significance.

Three sets of typical sensornets, Ξ, Φ and Γ, were defined as the training environment for the protocol tuning experiments described in section 4.2. We state the resulting near-optimal tunings for protocols A and B in section 4.3.2. Each node has identical capability, and was modelled on the popular Crossbow MICA2 mote [65].

Motes were distributed randomly within a square of fixed side length l yielding an

irregular geographic distribution of uniform planar density. This side length was selected such that average degree of connectivity was approximately 20 which is typical of sensor networks [120], and is within the *ad hoc horizon* limit of 10-20 nodes collaborating independently without hierarchical or external control [115].

Each of Ξ, Φ and Γ contained three networks differing only in geographic distribution. Networks $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$ contained 100 nodes, networks $\Xi = \{\xi_1, \xi_2, \xi_3\}$ contained 500 nodes, and networks $\Phi = \{\phi_1, \phi_2, \phi_3\}$ contained 1000 nodes. Using manufacturer-supplied data for typical communication range, transmission power and receiver sensitivity [65], to maintain an average degree of 20 we require $l_\Gamma \approx 9.4$Km, $l_\Xi \approx 21.0$Km and $l_\Phi \approx 29.7$Km.

To ensure comparison between networks of different scale is meaningful, the set of nodes for each 500-node network in Ξ is a strict superset of the corresponding 100-node network in Γ. Likewise, the set of nodes for each 1000-node network in Φ is a strict superset of the corresponding 500-node network in Ξ, and transitively is a strict superset of the corresponding 100-node network in Γ. Each smaller network is embedded, unchanged, within the corresponding larger network. The increase in network size is achieved by placing additional modelled MICA2 motes in the surrounding area, maintaining the original configuration, spatial density and average degree of connectivity.

Sensornet applications can generally be divided into two categories [48]: low-power, low-rate applications running for a long time, and high-rate, high-fidelity applications running for only a short time. For sensornets deployed into hazardous environments, such as those deployed to monitor a burning building or some other disaster area, if the disaster will draw to a conclusion within hours there is no benefit in reducing performance to extend network lifetime in the order of weeks. The duty management policy of modelled nodes in this chapter is the *null policy*; each modelled node remains awake and active at all times. This prevents the network size changing during experiments, which might otherwise give misleading results where we measure network behaviour versus network size. It is a reasonable model for sensornets in which performance is more important than longevity, or the application requires the highest possible levels of coverage redundancy.

Where experimenters wish to consider networks implementing other duty management policies, such as those described in chapter 7, these can be implemented on modelled nodes without requiring the experimental method to be modified. Consider a sensornet containing a large number of nodes, within which a scheduled sleep management policy arranges for some small subset to be active at any given time and for this duty to be

rotated fairly [288, 335]. Provided that the active subnetwork is similar at all times, the logical network visible to energy-ignorant protocols operating at the *Network layer* [343] is equivalent. It follows that the *Network layer* protocols can safely remain ignorant of this lower-layer detail without changed behaviour.

All internodal communication was defined to occur through anisotropic radio broadcast in an obstacle-free vacuum. Signal propagation and attenuation was modelled using the Friis free space model with exponent of 2.0 [97]. These are reasonable assumptions for modelled MICA2 motes which are fitted with conventional omnidirectional whip antennas [90] in a planar spatial configuration. As the MAC layer of the protocol stack is not within the scope of this chapter, we select a simple *Data Link Layer* based on IEEE 802.11 [137] which is both typical and popular for sensornets [341], and is compatible with the CC1000 RF transceiver [54] employed by the MICA2 platform [104]. Retransmission is disabled to prevent *Data Link Layer* detail obscuring the *Network Layer* effects.

For a uniform radio environment without shadowing effects caused by obstacles [227], such as that described above, it is reasonable to assume [204] that reception of any given packet broadcast at any given receiver may fail stochastically and independently with uniform probability ρ. We set $\rho = 0.05$ to model corruption derived from noise in the wireless medium, although network designers considering a specific deployment environment may wish to obtain a suitable value of ρ experimentally [328].

This stochastic packet loss can occur independently at any point along a multi-hop delivery path, hence the probability of a packet being successfully delivered along a path containing n nodes is given by $(1-\rho)^{n-1}$. Note that this is not the probability that a packet will be delivered successfully, as there may be more than one possible delivery path. Packet loss is also observed where two or more simultaneous but unconnected broadcasts overlap and interfere at a given recipient as a consequence of the *hidden terminal problem* [98].

The protocols considered in this chapter can achieve nearly 100% packet delivery under ideal conditions. However, little insight is to be gained by experiments addressing unrealistically favourable or disfavourable workloads. We therefore selected the network load and radio environment such that any protocol would be unlikely to achieve 100% delivery due to contention and broadcast corruption, but not so heavily as to load the network substantially beyond its capacity. This is to model usage patterns for a typical sensornet where the application designer wishes to process as much data within the network as possible, but without being so ambitious as to compromise network function.

Sensornet motes ran a simulated distributed sensing application in which every node periodically produces a small data packet. The destination of each packet is randomly selected from all motes in the network to prevent bias from any implicit structure in the geographic mote distribution. Other applications can be modelled by changing the distribution and frequency of packet production to match that of the desired application.

Each node in the network acted at different times as a packet source, a packet destination, or a packet relay. When a source node creates a packet it is queued for broadcast to the wireless medium. If the packet is eventually broadcast it may be received by one or more other nodes within communication range able to successfully extract the packet data from background noise. Packet headers specify one or more destinations, defining the only nodes at which a given packet can be consumed. In our experiments we specify exactly one destination per packet. Packet headers also specify TTL in terms of node-to-node hops and temporal lifespan to prevent stale packets circulating indefinitely.

Each packet recipient node independently determines how to handle an incoming packet. Three main classes of action are possible; the packet may be consumed, queued for rebroadcast, or dropped. The details of the criteria upon which the node makes this decision, and the state information upon which this decision is based, is dependent on the traffic distribution protocol selected by the sensornet designer. Some protocols may allow combinations; for example, a multicast packet may be both consumed and queued for rebroadcast to other recipients. Nevertheless, for all protocols the range of available actions is generally limited to these three possibilities.

Packet fusion [182], flow fusion [313], and data fusion [194] functionality is not considered in this chapter, as the effect of holding back simple or compound data packets until suitable fusion partners become available may obscure and confound the effects of tuning the routing protocols in the *Network layer*.

In section 6.1 we consider the robustness of tuning solutions to changes in network design by modifying networks $\xi_\alpha \in \Xi$. We provide full details of each alteration in section 6.1, but in all cases the modified networks are derived from Ξ by changing exactly one factor at a time.

4.3.2 Obtaining protocol tunings for training networks

Here we apply the three-phase experimental method defined in sections 4.2.2, 4.2.3 and 4.2.4 to obtain protocol tunings for training network sets Γ, Ξ and Φ.

	M_1	M_2	M_3
TBG	43	58	46
IGF	78	63	61

Table 4.1: Phase 1: τ_α values for metrics $M_1 - M_3$

4.3.2.1 Phase 1: Variance analysis

Consider $M = \{M_1, M_2, M_3\}$, the set of metrics defined in section 3.3. Table 4.1 presents τ_α measured experimentally for metrics $M_\alpha \in M$. Each value is rounded as $\lceil \tau_\alpha \rceil$, with the unit of measurement being the *second*. In each case we consider protocols running in the largest networks considered in this chapter $\phi_\alpha \in \Phi$; larger networks potentially require longer stabilisation periods as they are larger systems, and it may take longer for a given packet to traverse the network diameter. For the TBG protocol, $\forall M_\alpha \in M \bullet \tau_\alpha < 60s$. For protocol the IGF protocol, $\forall M_\alpha \in M \bullet \tau_{Ci} < 120s$. We therefore select simulation length $\tau_{sim} = 120s$ for both protocols, ensuring fair comparison and allowing a large safety margin for any anomalous solution instability.

4.3.2.2 Phase 2: Identify significant factors

In Phase 2 we identify which of the protocol controlled factors are the best predictors of the network performance metrics. This requires a small number of points in the parameter space to be sampled in the axis corresponding to each controlled factor, and a set of simulation experiments to be run to measure network performance under each combination. The ANOVA method is applied to assess which controlled factors are significant to the experimental outcomes [38]. Any factors which are deemed statistically insignificant at the 95% confidence level are dropped at this stage, and are not considered in Phase 3.

In this section we present result tables for 500-node networks $\xi_\alpha \in \Xi$. The same factors are found significant at the 95% confidence level for the 100-node networks $\gamma_\alpha \in \Gamma$ and 1000-node networks $\phi_\alpha \in \Phi$. All values are given to 4 decimal places; very small rounded values which are rounded to zero are actually small positive numbers.

Controlled factors $\{X_1 - X_6\}$ were considered at this stage for protocol A. The test suite size was calculated using the formula given in section 4.2.8 with $p = 6$, $q = 3$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 3^6 = 6561$, hence 6561 points in the factor-response phase space are available for model fitting. Table 4.2 presents the p-values for each controlled factor, and first-order pairwise interaction between factors.

Factors $\{X_4, X_5, X_6\}$ are significant in isolation with 95% confidence ($p < 0.05$) for at

	M_1	M_2	M_3
X_1	0.0565	0.2753	0.5196
X_2	0.7422	0.4093	0.9509
X_3	0.3925	0.6711	0.6048
X_4	0.0000	0.0000	0.3521
X_5	0.6881	0.0000	0.0000
X_6	0.0056	0.0000	0.1947
$X_1 \times X_2$	0.8716	0.1924	0.4157
$X_1 \times X_3$	0.8779	0.9825	0.4967
$X_1 \times X_4$	0.0189	0.7474	0.7598
$X_1 \times X_5$	0.9491	0.4740	0.9856
$X_1 \times X_6$	0.9787	0.7412	0.1371
$X_2 \times X_3$	0.2412	0.9027	0.9031
$X_2 \times X_4$	0.4802	0.5331	0.1796
$X_2 \times X_5$	0.3899	0.3729	0.9500
$X_2 \times X_6$	0.7407	0.0166	0.4551
$X_3 \times X_4$	0.4156	0.0737	0.4733
$X_3 \times X_5$	0.7441	0.7504	0.8016
$X_3 \times X_6$	0.8538	0.8753	0.5632
$X_4 \times X_6$	0.5187	0.0000	0.7627
$X_4 \times X_5$	0.1707	0.0000	0.7280
$X_5 \times X_6$	0.0014	0.0000	0.3845

Table 4.2: Phase 2: p-values for controlled factors X_1-X_6 and interactions for metrics M_1-M_3 for protocol A in Ξ

least two of the metrics M_1-M_3, and at least one of $\{X_4, X_5, X_6\}$ is evident in almost all interaction pairs deemed significant with 95% confidence. Factors $\{X_1, X_2, X_3\}$ are not significant in isolation for any metric, or as a member of an interaction pair which does not include any of $\{X_4, X_5, X_6\}$. Notably, the protocol-specific factor X_6 is statistically significant indicating that attempts to tune this protocol are appropriate.

Controlled factors $\{X_1 - X_5, X_7 - X_8\}$ were considered at this stage for protocol B. The test suite size was calculated using the formula given in section 4.2.8 with $p = 6$, $q = 3$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 3^6 = 6561$, hence 6561 points in the factor-response phase space are available for model fitting. Table 4.3 presents the p values for each controlled factor, and first-order pairwise interaction between factors.

Factors $\{X_4, X_7, X_8\}$ are significant in isolation with 99% confidence ($p < 0.01$) for all metrics $M_1 - M_3$. The controlled factor X_2 is significant with 99% confidence ($p < 0.01$) for metric M_1 and significant with 90% confidence ($p < 0.1$) for metric M_2. At least one of $\{X_2, X_4, X_7, X_8\}$ is evident in all interaction pairs deemed significant with at least 95% confidence ($p < 0.05$).

	M_1	M_2	M_3
X_1	0.4719	0.0700	0.8498
X_2	0.0000	0.0097	0.4277
X_3	0.7991	0.9275	0.9520
X_4	0.0000	0.0000	0.0000
X_5	0.2715	0.7656	0.8296
X_7	0.0000	0.0000	0.0000
X_8	0.0000	0.0000	0.0000
$X_1 \times X_2$	0.4286	0.0168	0.2781
$X_1 \times X_3$	0.7062	0.4756	0.7134
$X_1 \times X_4$	0.5566	0.4543	0.9572
$X_1 \times X_5$	0.3430	0.8381	0.1979
$X_1 \times X_7$	0.3985	0.5033	0.8244
$X_1 \times X_8$	0.2500	0.0281	0.0976
$X_2 \times X_3$	0.9996	0.7497	0.9969
$X_2 \times X_4$	0.0254	0.4034	0.6460
$X_2 \times X_5$	0.8217	0.8072	0.0623
$X_2 \times X_7$	0.7903	0.0997	0.9839
$X_2 \times X_8$	0.0392	0.0907	0.7985
$X_3 \times X_4$	0.9386	0.9718	0.9952
$X_3 \times X_5$	0.8137	0.8711	0.4870
$X_3 \times X_7$	0.5805	0.4790	0.9800
$X_3 \times X_8$	0.9058	0.8309	0.9626
$X_4 \times X_5$	0.7270	0.9223	0.9520
$X_4 \times X_7$	0.0000	0.0000	0.0000
$X_4 \times X_8$	0.0000	0.0000	0.0000
$X_5 \times X_8$	0.8240	0.7740	0.9712
$X_5 \times X_7$	0.3849	0.9782	0.8474
$X_7 \times X_8$	0.0000	0.0000	0.0000

Table 4.3: Phase 2: p-values for controlled factors X_1-X_5 and $X_7 - X_8$ and interactions for metrics M_1-M_3 for protocol B in Ξ

4.3.2.3 Phase 3: Extract protocol tunings

We consider only those tunable factors deemed statistically significant by Phase 2 of the three-phase experiment defined in section 4.2.1. In Phase 3 we sample the parameter space along the corresponding axis in a greater number of points for each statistically significant controlled factor. We select the midpoint value of the boundaries defined in section 3.2 for each controllable factor shown to be insignificant in Phase 2. Again, a set of simulation experiments was performed to measure network performance under each configuration. The ANOVA method was reapplied to confirm that significant factors were selected.

Factors $\{X_4, X_5, X_6\}$ were considered at this stage for protocol A. The test suite size was calculated using the formula given in section 4.2.8 with $p = 3$, $q = 10$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 10^3 = 9000$, hence 9000 points in the factor-response phase space are available for model fitting. Factors $\{X_2, X_4, X_7, X_8\}$ were considered at this stage for protocol B. The test suite size was calculated using the formula given in section 4.2.8 with $p = 4$, $q = 7$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 7^4 = 21609$, hence 21609 points in the factor-response phase space are available for model fitting. Analysis showed that all selected factors remained significant with at least 95% confidence.

The best-known input protocol tuning values for protocols A and B in training network sets Γ, Ξ and Φ are extracted by the method described in section 4.2.4. These protocol tuning parameter value sets are labelled $I_{A\Gamma}$, $I_{A\Xi}$, $I_{A\Phi}$, $I_{B\Gamma}$, $I_{B\Xi}$ and $I_{B\Phi}$, and are given to 2 decimal places in table 4.4. The values are taken directly from the candidate solutions S_α evaluated directly; better values may exist near these approximations to optimal values.

Function $f(I_{p\alpha}, I_{p\beta})$ in table 4.4 gives value I from network set β as a proportion of value I from network set α, both under protocol p. The magnitude of scaling factor f allows comparison of parameters tuned in different training networks. The magnitude of this scaling factor gives insight into the extent to which the value of a given factor is likely to change when tuned for different sizes of network. It is meaningless to compare tunable parameters across two or more different protocols (though it is not meaningless to compare the resulting network response metrics, which we do in section 4.3.3).

4.3.3 Measuring protocol tuning performance for deployment networks

We now evaluate the protocol tunings given in section 4.3.2 above to find the corresponding network behaviour characteristics in terms of the metrics M_1-M_3 defined in section 3.3. For

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
$I_{A\Gamma}$	*5.50*	*5.50*	*0.55*	2.30	5.00	0.60	-	-
$I_{A\Xi}$	*5.50*	*5.50*	*0.55*	2.82	10.00	0.73	-	-
$I_{A\Phi}$	*5.50*	*5.50*	*0.55*	8.35	10.00	1.00	-	-
$f(I_{A\Xi}, I_{A\Gamma})$	*1.00*	*1.00*	*1.00*	1.23	2.00	1.22	-	-
$f(I_{A\Phi}, I_{A\Xi})$	*1.00*	*1.00*	*1.00*	2.96	1.00	1.37	-	-
$f(I_{A\Phi}, I_{A\Gamma})$	*1.00*	*1.00*	*1.00*	3.63	2.00	1.67	-	-
$I_{B\Gamma}$	*5.50*	3.00	*0.55*	3.40	*5.50*	-	18.33	0.05
$I_{B\Xi}$	*5.50*	4.00	*0.55*	10.00	*5.50*	-	31.67	0.10
$I_{B\Phi}$	*5.50*	8.00	*0.55*	10.00	*5.50*	-	37.00	0.28
$f(I_{B\Xi}, I_{B\Gamma})$	*1.00*	1.33	*1.00*	2.94	*1.00*	-	1.73	2.00
$f(I_{B\Phi}, I_{B\Xi})$	*1.00*	2.00	*1.00*	1.00	*1.00*	-	1.17	2.80
$f(I_{B\Phi}, I_{B\Gamma})$	*1.00*	2.67	*1.00*	2.94	*1.00*	-	2.02	5.60

Table 4.4: Best-known protocol tunings: controlled factors

each protocol A and B we have distinct protocol tunings obtained in a 100-node network context, a 500-node network context, and a 1000-node network context. In section 4.3.2 we observe that the protocol tunings for different sizes of network differ in some controlled factors. We now determine whether this leads to a measurable difference in network behaviour.

Here we seek to address two issues.

1. *What is the best possible performance for each size of network? Is it the same for each?*

   It is harder for distributed applications and protocols to function effectively in large networks than in small networks. It would be unfair to compare dissimilar scenarios without reference to performance under a custom-tuned protocol.

2. *If we substitute a different size of deployment network to the training network, how severely (if at all) is the performance diminished?*

   This establishes whether the derived tunings are robust to redeployment in a different size of network.

From section 4.3.2 we have tunings $I_{A\Gamma}$, $I_{A\Xi}$ and $I_{A\Phi}$ for networks of size 100, 500 and 1000 respectively running protocol A, and tunings $I_{B\Gamma}$, $I_{B\Xi}$ and $I_{B\Phi}$ for networks of size 100, 500 and 1000 respectively running protocol B. We reuse our sets of test networks, testing each protocol and protocol tuning configuration enumerated above against each set of test networks Γ, Ξ and Φ. We repeat each combination of test network set, protocol and tuning 100 times in simulation experiments, presenting the arithmetic mean of resultant

solutions to 4 decimal places in table 4.5; all measurements are ±5%. Relative solution quality is measured by the E value defined in the interval $[0, \sqrt{3}]$ as described in section 3.3.4. We normalise E to the range $[0, 1]$ in the rightmost column for convenience.

Function $g(M_{q\alpha}, M_{q\beta})$ in table 4.4 gives value M from network set β as a proportion of value M from network set α, both under protocol tuning q. The selection of protocol tuning q implicitly defines the selected protocol as these are not interchangeable. The magnitude of scaling factor g gives insight into the extent to which the value of a given metric is likely to change when a given protocol tuning, derived from a given training network, is deployed into other networks of different size.

At this stage it is worth highlighting that there is no default or initial tuning against which to compare any other given tuning. In this chapter we do not attempt to calculate the attainable *best-case* and *worst-case* values algorithmically; instead, we use experimental observations. A wide range of values were observed for each of the metrics M_1-M_3 during the experiments from which the values given in table 4.5 were derived, covering the entire spectrum of behaviour from highly effective to highly deficient. It is against these observed extremes that we compare all other observed values as they provide the only meaningful baseline for comparison. The theoretically ideal solution in which all metrics M_1-M_3 are zero is impossible to achieve in reality, as is the theoretically worst solution in which all metrics M_1-M_3 take the maximum value defined in section 3.3.

4.3.4 Comparing tuned parameter sets

In this section we consider whether changing the training environment results in different parameter tunings.

Firstly, consider the TBG protocol tuned for 100-, 500- and 1000-node networks as given by $I_{A\Gamma}$, $I_{A\Xi}$ and $I_{A\Phi}$ respectively in table 4.4. We consider only parameters $X_4 - X_6$ as parameters $X_1 - X_3$ are not significant (see section 4.2.1) and parameters $X_7 - X_8$ are not defined for TBG.

We observe that, as the training network size increases, the best value for each parameter X_4, X_5 and X_6 also increases. For packet lifetime X_4 and TTL X_5 this is explained trivially by the average successful delivery route becoming longer with increasing network size; a longer route implies more hops and greater end-to-end latency.

For TBG rebroadcast probability X_6 we again consider increasing route length. As any node-to-node hop succeeds with probability $p \leq 1$, the cumulative probability of the

Protocol	Networks	Tuning	M_1	M_2	M_3	E	$E \div \sqrt{3}$
A: TTL-Bounded Gossip (TBG)	Γ	$I_{A\Gamma}$	7.6934×10^{-6}	1.8403×10^{-2}	6.6838×10^{-6}	0.5366	0.3098
		$I_{A\Phi}$	7.6179×10^{-6}	1.5903×10^{-2}	6.6760×10^{-6}	0.5331	0.3078
		$I_{A\Xi}$	7.5011×10^{-6}	4.8611×10^{-2}	7.2139×10^{-6}	0.5501	0.3176
		$g(E_{A\Xi}, E_{A\Gamma})$				0.9935	
		$g(E_{A\Phi}, E_{A\Xi})$				1.0319	
		$g(E_{A\Phi}, E_{A\Gamma})$				1.0252	
	Ξ	$I_{A\Gamma}$	7.7137×10^{-6}	1.9289×10^{-1}	3.7299×10^{-6}	0.3510	0.2027
		$I_{A\Xi}$	7.5597×10^{-6}	1.8758×10^{-1}	3.7206×10^{-6}	0.3431	0.1981
		$I_{A\Phi}$	6.9839×10^{-6}	5.1603×10^{-1}	1.5100×10^{-5}	0.6105	0.3525
		$g(E_{A\Xi}, E_{A\Gamma})$				0.9775	
		$g(E_{A\Phi}, E_{A\Xi})$				1.7794	
		$g(E_{A\Phi}, E_{A\Gamma})$				1.7393	
	Φ	$I_{A\Gamma}$	7.0807×10^{-6}	7.3619×10^{-1}	6.5871×10^{-6}	0.7616	0.4397
		$I_{A\Xi}$	8.0713×10^{-6}	5.7489×10^{-1}	1.5803×10^{-6}	0.6070	0.3522
		$I_{A\Phi}$	8.2265×10^{-6}	5.7574×10^{-1}	1.5948×10^{-6}	0.6119	0.3533
		$g(E_{A\Xi}, E_{A\Gamma})$				0.7970	
		$g(E_{A\Phi}, E_{A\Xi})$				1.0081	
		$g(E_{A\Phi}, E_{A\Gamma})$				0.8034	
B: Implicit Geographic Forwarding (IGF)	Γ	$I_{B\Gamma}$	3.3408×10^{-5}	1.0186×10^{-1}	3.2307×10^{-5}	0.1020	0.0589
		$I_{B\Xi}$	6.0906×10^{-5}	1.3398×10^{-1}	3.5870×10^{-5}	0.1343	0.0775
		$I_{B\Phi}$	1.6890×10^{-4}	2.0541×10^{-1}	4.3742×10^{-5}	0.2071	0.1196
		$g(E_{B\Xi}, E_{B\Gamma})$				1.3167	
		$g(E_{B\Phi}, E_{B\Xi})$				1.5421	
		$g(E_{B\Phi}, E_{B\Gamma})$				2.0304	
	Ξ	$I_{B\Gamma}$	3.1234×10^{-5}	3.0535×10^{-1}	3.0208×10^{-5}	0.3059	0.1766
		$I_{B\Xi}$	5.5244×10^{-5}	3.0329×10^{-1}	2.9227×10^{-5}	0.3040	0.1755
		$I_{B\Phi}$	1.3931×10^{-4}	4.0803×10^{-1}	2.0890×10^{-5}	0.4092	0.2362
		$g(E_{B\Xi}, E_{B\Gamma})$				0.9938	
		$g(E_{B\Phi}, E_{B\Xi})$				1.3461	
		$g(E_{B\Phi}, E_{B\Gamma})$				1.3377	
	Φ	$I_{B\Gamma}$	3.1869×10^{-5}	6.7005×10^{-1}	9.8354×10^{-6}	0.6733	0.3887
		$I_{B\Xi}$	5.5884×10^{-5}	6.4415×10^{-1}	8.6119×10^{-6}	0.6474	0.3738
		$I_{B\Phi}$	1.4589×10^{-4}	5.4937×10^{-1}	4.9874×10^{-6}	0.5527	0.3191
		$g(E_{B\Xi}, E_{B\Gamma})$				0.9615	
		$g(E_{B\Phi}, E_{B\Xi})$				0.8537	
		$g(E_{A\Phi}, E_{A\Gamma})$				0.8209	

Table 4.5: Best-known protocol tunings: measured responses

sequence of x node-to-node hops along any possible delivery path is given by $p^x \leq p$. As the network size increases, the number of potential delivery paths also increases. However, the probability of successful delivery along any such path decreases simultaneously.

Secondly, we consider the IGF protocol tuned for 500- and 1000-node networks as given by $I_{B\Xi}$ and $I_{B\Phi}$ respectively in table 4.4. We consider only parameters $\{X_2, X_4, X_7, X_8\}$ as parameters $\{X_1, X_3, X_5\}$ are not significant (see section 4.2.1) and parameter X_6 is not defined for IGF. As training network size increases, the best value for each parameter X_2, X_4, X_7, and X_8 increases.

Average successful delivery route length grows with increasing network size. The packet lifetime X_4 must therefore increase as packets require more time to traverse the longer delivery routes. Under the IGF protocol, a single node-to-node hop implies a three-phase process. As described in section 3.2.4 a sending node transmits a short CTS packet, but then must listen for RTS responses from neighbouring nodes before proceeding. As this wait time is non-zero, longer routes imply more of these waiting periods.

Unlike TBG, the TTL X_5 is not statistically significant. This is explained by IGF always selecting next-hop relay nodes such that delivery routes closely approximate straight lines between source and destination. The straighter the path, the fewer node-to-node hops are implied. As IGF tends to deliver along relatively straight paths, and average delivery path length grows sublinearly in node count, the impact on average TTL is not significant in networks of the size considered in this chapter. It might be expected that this factor becomes significant for substantially larger networks.

As described in section 3.2.4, in selecting a next-hop relay the IGF protocol seeks to minimise the angle between two lines; the first between the current packet location and the destination, and the second between the next-hop relay node and the destination. Larger angles imply a greater deviation from the ideal. In short delivery routes, the impact of a single significant deviation from the ideal is greater than a similar deviation in a longer delivery route. As larger networks tend to imply longer average delivery routes, it is more acceptable for individual node-to-node hops to divert substantially from the ideal without significant adverse impact.

As the number of network nodes increases, the number of packet sources and the number of concurrent packet delivery attempts increases also. The number of packets in transit at any given time will thus increase, implying a need for larger buffers for waiting packets as specified by X_2. A given node might be the best next-hop candidate for a given

packet, but may not be immediately available if already handling a different packet. The time in which a node waits for RTS packets from neighbours, specified by state timeout base X_8, must therefore increase such that a sufficient number of nodes are available as candidate next-hop relays such that at least one is a reasonably high-quality choice.

4.3.5 Comparing networks under best and alternative tunings

In this section we consider whether there is a substantial difference in observed network performance when deploying a protocol tuned to that specific environment as opposed to a protocol tuned to a different environment. Table 4.5 presents unnormalised values of metrics M_1-M_3 and overall solution quality E for all possible combinations of protocol (TBG and IGF) and test network set (Γ, Ξ and Φ) considered in this chapter. For each metric M_1-M_3 a wide range of measured values were observed; although the values of a given metric for a given protocol vary somewhat in table 4.5, this variation is small when compared to the variation between the smallest and largest values observed across the experiment set. As described in section 3.3.4 all values of E are found in $[0, \sqrt{3}]$; for convenience we also provide values of E normalised to the range $[0, 1]$.

We are interested in the relative performance of different protocol tunings for a given network rather than the absolute performance of any given tuning. More specifically, we assess whether any observed difference in measured performance is within the $\pm 5\%$ experimental error defined in section 4.2.1. If not, then the observed difference cannot be dismissed as insignificant. Function g gives the ratio of two E values, allowing comparison of two such values independently of scale. Provided that $g \in [0.95, 1.05]$ we conclude that the observed difference is statistically insignificant. Although not the focus of this section, we observe in passing that the E values are substantially smaller for IGF than TBG for each network size considered, suggesting informally that the IGF protocol outperforms TBG for these networks.

Firstly, consider networks implementing the TBG protocol. We observe little difference in behaviour for 100-node networks in Γ across tunings $I_{A\Gamma}$, $I_{A\Xi}$ and $I_{A\Phi}$. For 500-node networks in Ξ, we observe little difference between $I_{A\Xi}$ and $I_{A\Gamma}$, but a significant difference between $I_{A\Xi}$ and $I_{A\Phi}$. Likewise, for 1000-node networks in Φ, we observe little difference between $I_{A\Phi}$ and $I_{A\Xi}$, but a significant difference between $I_{A\Phi}$ and $I_{A\Gamma}$. We conclude that the selection of an appropriate TBG tuning is more significant for larger networks.

Secondly, consider networks implementing the IGF protocol. For 100-node networks

in Γ a significant difference is observed across tunings $I_{B\Gamma}$, $I_{B\Xi}$ and $I_{B\Phi}$. For 500-node networks in Ξ, we observe little difference between $I_{B\Xi}$ and $I_{B\Gamma}$, but a significant difference between $I_{B\Xi}$ and $I_{B\Phi}$. Likewise, for 1000-node networks in Φ, we observe little difference between $I_{B\Phi}$ and $I_{B\Gamma}$, but a significant difference between $I_{B\Phi}$ and $I_{B\Xi}$. We conclude that the selection of an appropriate IGF tuning is significant for all network sizes considered in this chapter.

We note that for each protocol, and for each network size, the protocol tuning obtained using a training network similar to the deployment network is associated with good network performance. A given protocol tuning may work well in networks smaller or larger than the training network, but this is not guaranteed. For each protocol, network performance tended to decline with increasing network size; we address this matter in greater depth in section 6.5.

4.3.6 Observations and limitations

The method described here works well in complex situations where there are multiple controlled factors and multiple measured responses. However, it is not perfect. It is necessary to define the ranges of controlled factors to be explored. Sometimes this is straightforward. For example, X_6 is defined over a small finite range with good values known to be found near the centre. In other cases it is necessary to perform preliminary experiments to find useful ranges of controlled factors, and it is not guaranteed that these will be the best ranges when combined with other factors.

In Phase 1 it is possible that some network scenarios never reach stable behaviour and hence there does not exist a simulation period after which the simulation can be considered sufficiently stable that the behaviour can be measured and sampled in any meaningful manner.

In Phase 2 a large number of controlled factors are considered at low resolution, and in Phase 3 a small number of controlled factors are considered at high resolution. However, this assumes that there exists a small set of controlled factors which are more significant than the others. If this assumption does not hold then it is difficult to specify which can be discarded, leaving the subsequent Phase 3 high resolution modelling intractable or at least infeasible in reasonable time.

One solution is to tighten the definition of factor significance. The required confidence level (reducing the maximum acceptable value of p as discussed in section 4.2.3) could be

increased, for example from 95% to 99%. Alternatively, we could rank the factors in order of decreasing measured significance and retain only the top f most significant factors, where f is some acceptably small integer.

In Phase 2 we are interested only in identifying the significant factors and are uninterested in any specific value of these factors. However, in Phase 3 we would like to have confidence that the solution obtained is a reasonable approximation of the optimal solution. Difficulties are encountered if the factor-response landscape is particularly complex. If there are insufficient sampling points it is possible that the best values fall between these points and are therefore missed.

Sensornets are often deployed into hostile environments, the details of which cannot be fully known at design time prior to deployment [191]. It is generally not possible to measure all possible solutions or deployment scenarios if there are a large or infinite number. A *robust* solution is able to cope with a range of deployment environments which differ from the training environment; if network *Quality of Service* (QoS) is affected, it will gracefully degrade. In contrast, a *fragile* solution may offer excellent performance in a specific training environment but degrade quickly if the deployment environment deviates even slightly from the training environment [62].

A sensornet implementing a given protocol tuning I is expected to perform well where the deployment context is similar to that in which the tuning I is obtained. However, consider the deployment of a sensornet in the real world where the protocol tuning work has been implemented by simulation. It is unrealistic to assume that the simulation model is a perfect and complete model of reality. It is also unrealistic to assume that the modelled environment, or the composition of the sensornet itself, will necessarily remain constant. For example, if a sensornet is deployed to monitor seismic activity, should an earthquake strike it may cause nodes to move their geographic position or to fail. Additional nodes might be added to areas of emerging importance during such activity.

It is therefore important to consider the *robustness* of a given protocol tuning to changes in network composition and deployment context. In the preceding work we define and implement a three-phase method to tune protocols for a specified sensornet environment. We now assess the extent to which the measured behaviour of a sensornet, in which the resulting tuned protocols are deployed, is *robust* to variation in node numbers, spatial distribution and spatial density. A *robust* protocol tuning would perform acceptably in the modified deployment context, whereas a *brittle* protocol tuning would perform poorly.

We measure a set of network response metrics for a given fixed protocol tuning as we vary a single deployment context controlled factor in isolation, and examine the relationship between the measured responses and the controlled factor.

4.4 Summarising factor-response relationships

In section 4.2.4 we discuss a method in which the parameter space is sampled evenly, with each sampling point evaluated for merit as a candidate solution. As discussed above, this runs the risk that good candidate solutions are missed as they fall between sampling points, which is particularly problematic with noisy data or a complicated parameter landscape. One approach to addressing this problem is to produce the evenly distributed factor-response relationship sample set as per section 4.2.4, but then apply statistical methods to interpolate between sampling points.

Statistical techniques can be employed to interpolate between sampling points by fitting statistical models to the set of sampling points and associated network measurements, similar to the method outlined in section 4.2.3. This model fitting method yields a set of simultaneous equations, at least one per metric, defined in terms of the controlled factors. Instead of taking the protocol tuning solution directly from observed measurements of the $S \mapsto T$ mapping, it is extracted by a simple minimisation approach. The simultaneous equations for metrics are substituted for the absolute values in the solution quality functions discussed in section 3.3.4, and the resulting equation is minimised for E [291].

Fitting a surface to approximate the parameter landscape may be difficult if a suitable statistical model cannot be found, or if multiple transformations must be applied to the raw data to ensure an adequate fit. There also exists the issue of determining a sufficient number of data points to obtain a fitted model offering acceptably accurate interpolation between sampled points. Well-known statistical techniques exist to assess the *quality of fit* [38]. More complex factor-response relationships, such as those associated with complex protocols, may yield more complex surfaces which would be inadequately approximated by overly simple statistical models. In such situations the solutions extracted from fitted statistical models may be of lower quality than the principled sampling approach discussed in this chapter.

4.5 Summary

Section 4.1 describes and defines the sensornet protocol tuning problem. It is a complex real-world problem with multiple inputs, multiple outputs, and multiple objectives. The non-trivial interrelationships between these factors are not known at the outset, and thus cannot be targeted specifically during experiment design.

Section 4.2 defines a repeatable and automatable three-phase engineering method based on *Design Of Experiments* principles with which to tune a sensornet protocol for a given network environment. This method analyses the statistical relationship between controlled factors and measured responses, as described in sections 3.2 and 3.3 respectively, using simulation to sample the continuous parameter space in bounded time. Mathematical models are fitted to experimental results to approximate and summarise the factor-response relationship. These models can then be analysed to estimate response values for arbitrary protocol tunings, or to obtain sets of protocol tunings for which the estimated response conforms to specified network non-functional requirements.

Section 4.3 presents protocol tuning solutions for the TBG and IGF protocols trained in 100-, 500- and 1000-node networks obtained using the method described in section 4.2. Estimates of solution quality are given in comparison to the ideal solution, as described in section 3.3. Section 4.3 demonstrates that a good protocol tuning solution derived in a given training environment may be similar, but not identical, to good protocol tuning solutions derived in different training environments.

Chapter 5

Evolutionary methods for sensornet protocol tuning

The principled search method, as defined and applied in chapter 4, uses Factorial Design (FD) to sample the entire problem space in a systematic and even manner [22]. This is a broad-but-shallow deterministic search method, and has been shown to be effective for the protocol tuning problems considered thus far. However, the experimental cost grows rapidly in the number of controlled factors. This is problematic if Phase 2 of the experimental method, as defined in section 4.2.3, does not reduce the significant factor set to an acceptable size. This is an attribute of the tuning problem rather than the tuning method, and hence cannot be ignored.

One possible solution is to replace the sampling approach of Phases 2-3 with an alternative approach better suited to large numbers of controlled factors. For example, a *Multi-Objective Evolutionary Algorithm* (MOEA) such as NSGA2 [73] or Two-Archive [236] could be applied to the optimisation problem. These algorithms sample the problem space in a guided and uneven manner. This is a narrow-but-deep stochastic search method. In this section we compare a principled sampling search approach against an evolutionary approach, measuring differences in the optimised solutions and the corresponding network behaviour these solutions induce.[1]

[1]Evolutionary search experiments in chapter 5 were conducted in collaboration with B. Woolford-Lim and X. Yao, as discussed in the thesis declaration on page xvii, who were responsible for the selection of evolutionary search algorithm and associated search parameters, but not the experiment design, analytical method, interpretation of results, or extraction of conclusions.

5.1 Characteristics of evolutionary search

The evolutionary approach is better suited to problems involving large numbers of potentially significant factors, as all factors can be mutated at each generation. The evolutionary approach is also better suited to the exploration of complex, undulating parameter landscapes, for which an even sampling could miss interesting regions between sampling points. However, there are also downsides. We sacrifice the guarantee of discovering global maxima and minima. Furthermore, we are no longer guaranteed to cover all regions of the parameter landscape, and may therefore fail to identify global minima and maxima. We also do not produce the source data required for the building of predictive statistical fitted models as discussed in section 4.4.

We can optionally use the sampling approach outlined in section 4.2.4 to survey the parameter landscape, identifying regions likely to contain good values, and use these to seed the initial population. To minimise experimental cost and maximise solution quality for the evolutionary algorithms we could obtain a survey of the problem space, executing a low-resolution version of the FD approach to locate regions of the problem space in which some high quality solutions reside. We could then apply this insight to focus the evolutionary algorithms, seeding the working sets with values in these regions but allowing the evolutionary algorithms to explore the entire problem space. However, the results presented in section 5.3 demonstrate that this is not necessary for the MOEAs considered in this chapter. A random initial population, equivalent to a single-stage random search, is acceptable in obtaining a sufficient diversity of candidate solutions.

5.2 Experimental method

The experimental method employed to tune sensornet protocols using Multi-Objective Evolutionary Algorithms is similar to that employed in the evaluation of principled search methods described in section 4.2. With each evolutionary generation the *fitness* of a set of candidate protocol tuning solutions is measured by simulation experiment, and the resulting *fitness scores* are used by the evolutionary algorithm to determine which of the candidate solutions survive to the next generation, or contribute to new candidates by mutation and crossover, or dropped completely.

A set of three typical sensornets, $\Omega = \{\omega_1, \omega_2, \omega_3\}$ was defined and reused for all experiments in chapter 5. Each sensornet $\omega_\alpha \in \Omega$ consisted of 250 static motes of identical

capability modelled on the Crossbow MICA2 mote, and apart from the node population size is similar to the networks in Γ, Ξ and Φ discussed in sections 4.3 and 6.1. The fixed side length for the square planar region within which motes are spatially distributed is set as $l_\Omega \approx 14.8$Km for mote spatial density and degree of connectivity consistent with Γ, Ξ and Φ. Simulated motes run a distributed sensing application in which every node periodically produces a small data packet. The destination of each packet is randomly selected from all motes in the network to prevent bias from any implicit structure in the mote distribution.

5.2.1 Controlled factors and measured responses

Multi-Objective Evolutionary Algorithms are, as the name suggests, designed for problems in which there are multiple objectives [113]. In chapter 5 we increase the number of optimisation objectives to highlight the advantage conferred by this approach. The metrics M_1-M_3 are defined in section 3.3. We extend this set by defining metrics M_{1a} and M_{3a}, which are similar to metrics M_1 and M_3 respectively but measure distance in terms of logical network units rather than the original physical units. It follows that the E composite quality metric, defined in section 3.3.4, covers the range $[0, \sqrt{5}]$ as $|M| = 5$. We therefore scale E by $1 \div \sqrt{5}$ to map $[0, \sqrt{5}]$ to $[0, 1]$.

M_{1a}**:** *Latency per hop:* Mean time for a packet to travel 1 node-node hop. Measured in $hop^{-1}s$. Defined in the range $(0, \infty)$.

M_{3a}**:** *Energy per packet per hop:* Mean energy for 1 packet to travel 1 node-node hop. Measured in $Jpacket^{-1}hop^{-1}$. Defined in the range $(0, \infty)$.

The controlled factors for the TBG and IGF protocols are unchanged from those defined in section 3.2.

5.2.2 Design Of Experiments approach: Factorial Design

The Factorial Design approach described in section 4.2 is repeated for the 250-node networks $\omega_\alpha \in \Omega$. This section discusses the specific experiments performed rather than describing the method.

	M_1	M_{1a}	M_2	M_3	M_{3a}
TBG	43	19	58	46	49
IGF	78	27	63	61	51

Table 5.1: Phase 1: τ_α values for metrics $M_1 - M_{3a}$

5.2.2.1 Phase 1: Variance analysis

The variance analysis experiment results given in section 4.3.2.1 were repeated for the new metrics M_{1a} and M_{3a}. Table 5.1 presents τ_α measured experimentally, for all metrics $M = \{M_1, M_{1a}, M_2, M_3, M_{3a}\}$. Each value is rounded as $\lceil \tau_\alpha \rceil$, with the unit of measurement being the *second*. Again we find that, for the TBG protocol, $\forall M_\alpha \in M \bullet \tau_\alpha < 60s$. For protocol the IGF protocol, $\forall M_\alpha \in M \bullet \tau_{Ci} < 120s$. We therefore select simulation length $\tau_{sim} = 120s$ for both protocols, ensuring fair comparison and allowing a large safety margin for any anomalous solution instability.

5.2.2.2 Phase 2: Factor significance screening

The factor significance screening experiments given in section 4.3.2.2 are repeated for the TBG and IGF protocols in networks $\omega_\alpha \in \Omega$.

Controlled factors $\{X_1 - X_6\}$ were considered at this stage for TBG. The test suite size was calculated using the formula given in section 4.2.8 with $p = 6$, $q = 3$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 3^6 = 6561$, hence 6561 points in the factor-response phase space are available for model fitting.

Factors $\{X_4, X_5, X_6\}$ are significant in isolation with 95% confidence ($p < 0.05$) for at least two of the metrics M_1-M_{3a}, and at least one of $\{X_4, X_5, X_6\}$ is evident in almost all interaction pairs deemed significant with 95% confidence. Factors $\{X_1, X_2, X_3\}$ are not significant in isolation for any metric, or as a member of an interaction pair which does not include any of $\{X_4, X_5, X_6\}$. Notably, the protocol-specific factor X_6 is statistically significant indicating that attempts to tune this protocol are appropriate.

Controlled factors $\{X_1 - X_5, X_7 - X_8\}$ were considered at this stage for IGF. The test suite size was calculated using the formula given in section 4.2.8 with $p = 7$, $q = 3$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 3^7 = 19683$, hence 19683 points in the factor-response phase space are available for model fitting.

Factors $\{X_4, X_7, X_8\}$ are significant in isolation with 99% confidence ($p < 0.01$) for all metrics $M_1 - M_{3a}$. The controlled factor X_2 is significant with 99% confidence ($p < 0.01$) for metric M_1 and significant with 90% confidence ($p < 0.1$) for metric M_3. At least one

of $\{X_2, X_4, X_7, X_8\}$ is evident in all interaction pairs deemed significant with at least 95% confidence ($p < 0.05$).

5.2.2.3 Phase 3: High resolution modelling

The factor significance screening experiments given in section 4.3.2.3 are repeated for the TBG and IGF protocols in networks $\omega_\alpha \in \Omega$.

Factors $\{X_4, X_5, X_6\}$ were considered at this stage for the TBG protocol. The test suite size was calculated using the formula given in section 4.2.8 with $p = 3$, $q = 10$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 10^3 = 9000$, hence 9000 points in the factor-response phase space are available for model fitting. Factors $\{X_2, X_4, X_7, X_8\}$ were considered at this stage for the IGF protocol. The test suite size was calculated using the formula given in section 4.2.8 with $p = 4$, $q = 7$, $r = 3$ and $s = 3$. This gives a test suite size of $3 \times 3 \times 7^4 = 21609$, hence 21609 points in the factor-response phase space are available for model fitting.

We calculate the solution quality metric E, defined in section 3.1, for each of the rsq^p sampling points. To mitigate the influence of outliers and experimental noise, we calculate the mean value of E from the rs experiments corresponding to each of the q^p unique candidate solutions. We select the candidate solution associated with the lowest mean E value, and hence highest solution quality. Section 5.3 presents the resulting solutions for the TBG and IGF protocols.

5.2.3 Evolutionary approach: SPEA2

In this section we define the experiments with which the parameter landscape is explored, at narrow scope but substantial depth, using an evolutionary approaches. The *Strength Pareto Evolutionary Algorithm 2* (SPEA2), widely covered in the literature [113], is utilised.

5.2.3.1 SPEA2 experimental configuration

SPEA2 [345] is a revised version of the Strength Pareto Evolutionary Algorithm designed by Zitzler *et al.* [346]. The population size s defines the size of the working population, and also defines the archive capacity. Greater numbers of optimisation objectives require greater s-values, which in turn imply greater computation cost. Although we consider 5 metrics, the uniform spatial density of the networks leads to a strong correlation between

M_1 and M_{1a}, and between M_3 and M_{3a}. It follows that we can justifiably use the 3-metric value of the population scheme defined by Khare *et al.* [158], giving $s = 50$. SPEA2 also takes a parameter, k, specifying the k-th nearest neighbour in density estimation calculations. As in the original SPEA2 experiments [345] we take $k = \sqrt{2s}$, rounded to $k = 7$.

Simulated Binary Crossover (SBX) [72], polynomial mutation [72], and random selection operators were implemented. SBX takes as parameters a crossover rate, c, and an η_c value controlling the probability of *near-parent solutions* being generated; higher values produce closer matches to parents. Appropriate values must be selected for these parameters; there is some debate as to the relative importance of crossover and mutation [322]. We set $c = 0.7$, so that crossover occurred often to generate a diverse range of child solutions. We set $\eta_c = 15$, to encourage generation of relatively "close" solutions. Polynomial mutation takes two parameters; the mutation rate, m, and mutation distance, η_m. We set $m = \frac{1}{6}$ such that, on average, one input variable would be mutated in each solution. We set $\eta_m = 20$ to promote small mutation steps and thus encourage convergence.

Experiments were conducted in which all values were represented internally as 64-bit precision floats. Where a given parameter is defined only for integral values, the float value was rounded down to the nearest integer at the point of use. Each candidate solution fitness evaluation considered the three networks defined in section 5.2, with each combination of candidate solution and network repeated three times to reject the influence of outliers.

Preliminary tests showed rapid convergence within the early generations, with few improvements thereafter. Based on these results, all tests were run for 50 generations to allow convergence to occur. Data on the best known candidate solutions were logged at every generation to provide insight into the running convergence of the system.

5.2.3.2 Cost analysis

The SPEA2 algorithm runtime is negligible compared to that of fitness function evaluation by simulation, so we need consider only those overheads relating to fitness function evaluation. Consider an evolutionary run with a population size of a for which b generations are required to attain the required solution quality. It is also necessary to evaluate the initial zeroeth population prior to evolution commencing.

Within each generation it is necessary to evaluate the fitness function once for each candidate solution, requiring $a(b + 1)$ evaluation instances for all population members

across all generations. As with the FD experiments, we test r networks and repeat each experimental configuration s times, requiring $a(b+1)rs$ simulations in total. Total cost grows linearly in each of a, b, r and s. This predictable and readily controllable cost growth is a desirable attribute.

It is possible, though unlikely, that a candidate solution already in the archive be selected as a parent but no crossover or mutation occurs. In this rare situation it would be possible to re-use an earlier fitness evaluation and hence reduce experimental cost. However, we do not cache fitness evaluation data between generations. By forcing evaluation of each candidate solution in each iteration we significantly decrease the possibility that a low-quality candidate, assigned an undeservedly high-quality evaluation owing to experimental noise, can survive from generation to generation.

5.2.4 Evolutionary approach: Two-Archive

In this section we define the experiments with which the parameter landscape is explored using a different evolutionary algorithm. The *Two-Archive* (TA) algorithm has separate convergence and diversity archives, rather than the single archive used by SPEA2, and therefore may be more or less suited to the protocol optimisation problem considered here.

5.2.4.1 Two-Archive experimental configuration

The Two-Archive algorithm is a multi-objective evolutionary algorithm developed by Praditwong and Yao [236]. The working population size s is also the total combined capacity of separate convergence and diversity archives. Two-Archive takes a parameter, $r \in [0, 1]$, defining the ratio of parent selection between the convergence and diversity archives. A higher ratio r selects more parents from the convergence archive, obtaining faster solution convergence at the expense of potentially reduced solution diversity.

As with SPEA2, Simulated Binary Crossover (SBX) [72], polynomial mutation [72] and random selection operators are used. Where SPEA2 and Two-Archive have a given parameter in common, we reuse the value considered in section 5.2.3.1 to enable meaningful comparison between the algorithms. We use $s = 50$, $c = 0.7$, $\eta_c = 15$, $m = \frac{1}{6}$ and $\eta_m = 20$. We set $r = 0.9$ to favour parents from the convergence archive, encouraging strong solution convergence in a reasonable number of generations.

Each candidate solution fitness evaluation implied three simulation iterations for each of the three networks defined in 5.2 to reject the influence of outliers. All tests were run

for 50 generations.

5.2.4.2 Cost analysis

The Two-Archive algorithm cost is similar to that of the SPEA2 algorithm cost, as discussed in section 5.2.3.2. As with SPEA2, the cost of executing the evolutionary algorithm itself is negligible in comparison to the cost of fitness function evaluation, so we need only consider those costs relating to the latter. As with the SPEA2 experiments we require $ars(b+1)$ simulation instances in total. Total cost grows linearly in each of a, b, r and s.

5.3 Results

We now summarise the output of the Factorial Design experiments described in section 5.2.2, the SPEA2 experiments described in sections 5.2.3, and the Two-Archive experiments described in section 5.2.4. We label the Factorial Design instance as A, the SPEA2 instance as B, and the Two-Archive instance as C.

5.3.1 Optimised protocol tunings for TBG

Tables 5.2 and 5.3 gives summarised optimised tuning results for the TBG protocol. The sets of tuned protocol values corresponding to A, B and C are labelled I_A, I_B and I_C respectively. All figures for non-integral factors are given to 4 decimal places.

For each experimental approach, the set of values assigned to controlled factors X_1-X_6 corresponding to the highest quality solution discovered is given in table 5.2. For approach A some controlled factors were not evaluated directly in Phase 2 of the experiment. For these controlled factors, italicised in table 5.2, we take the midpoint of search ranges defined in sections 3.2 and 5.2.1.

	X_1	X_2	X_3	X_4	X_5	X_6
I_A	*6*	*6*	*0.5500*	2.8200	10	0.7300
I_B	5	3	0.88857	1.1151	9	0.75832
I_C	5	3	0.3864	6.8320	9	0.8425

Table 5.2: Best known TBG tunings

We define the highest quality solution I_α for approach α as being that which offers the smallest Euclidean distance E_α between O_α and the theoretical perfect values of metrics, as defined in sections 5.2.1 and 3.3.4. Table 5.2 shows the Euclidean distances E_A, E_B

and E_C from which I_A, I_B and I_C were identified as the highest quality solutions derived by approaches A, B and C respectively. Note that the theoretical perfect metric values are not necessarily attainable under *any* real protocol tuning.

To ensure fair comparison of the quality of solutions obtained by the two experimental approaches, it is necessary to eliminate any factors which could unfairly influence the outcome. We achieved this goal by conducting further simulation experiments as per section 5.2 where the simulation scenario is identical in all respects, except for the protocol parameter set which is either I_A, I_B, or I_C as appropriate.

Three hundred simulations were executed for each of I_A, I_B, and I_C as defined in table 5.2; 100 repeats for each of the 3 networks considered in the experiments of sections 5.2.2, 5.2.3 and 5.2.4. Where a controlled factor X_1-X_6 is defined only for integral values, but the value identified by experiment and analysis is non-integral, we configure our experiments with values rounded to the nearest integer. For each combination of experimental approach and metric M_1-M_5, a set of 300 output values is produced. The arithmetic mean of each set is taken as the final value and presented in table 5.3. The sets of output metrics corresponding to A and B are labelled O_A, O_B, and O_C respectively.

All figures for M_1-M_5 are give to 5 significant figures and scaled by a factor of 10^6 for clarity.

	M_1	M_{1a}	M_2	M_3	M_{3a}	E
O_A	15475	9.0312	439010	758.28	0.39873	0.50528
O_B	15458	8.7817	453610	847.75	0.43416	0.50669
O_C	14297	8.1415	419650	781.58	0.39808	0.46875
Best	TA	TA	TA	TA	FD	

Table 5.3: Network performance for best known TBG tunings

It is notable that the protocol tunings found by the three approaches, as shown in Table 5.2, are different but broadly similar. The corresponding observed metrics given in Table 5.3 confirm that the network behaviour induced by each of the three approaches was different but broadly similar. This similarity suggests that each derived protocol tuning approximates an optimal solution nearby in the parameter landscape.

Observe that the Two-Archive approach yielded the best result for metrics M_1-M_3, the Factorial Design approach yielded the best result for metric M_{3a}, and the SPEA2 approach did not yield the best result for any metric. It is therefore unsurprising that Two-Archive yielded the solution with best overall quality as measured in E, followed

by Factorial Design and SPEA2 in order of declining quality. We conclude that the Two-Archive approach is superior for tuning the TBG protocol, but that any of those approaches considered here would yield a reasonable solution.

Figure 5.1 plots normalised solution quality, E, versus evolutionary generation to illustrate the convergence of SPEA2- and Two-Archive-derived solution quality toward the final solution quality which could be achieved with an infinite number of generations. For comparison, the solution quality obtained by the Factorial Design approach is also shown as a constant.

The quality of solutions obtained by the SPEA2 approach was not observed to supplant that attainable by Factorial Design results, but gradually improved until generation 18 after which no further improvement was observed. Although theoretically possible, the attainment of parity or advantage by further improvement under SPEA2 is unlikely to occur within acceptable time. Under the Two-Archive approach, however, the solution quality is near-constant for the first 9 generations before improving dramatically, at which point it becomes significantly better than that attained by Factorial Design, with no further improvement observed after generation 11.

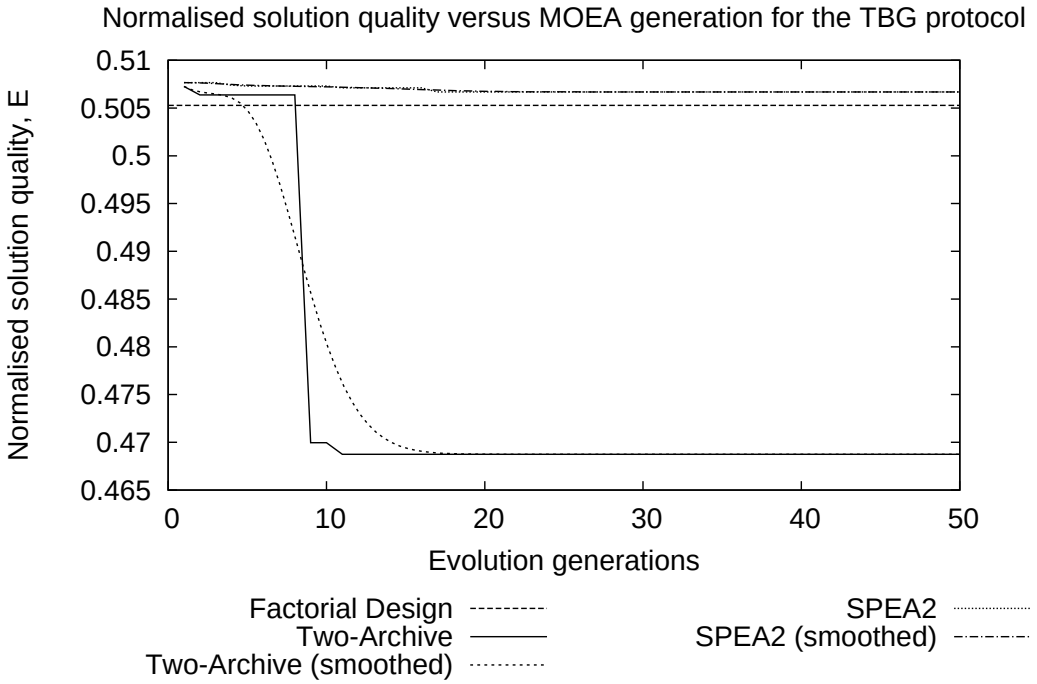


Figure 5.1: Comparing TBG solution quality using Factorial Design, SPEA2 and TA

This type of behaviour is characteristic of MOEAs [113]; solution quality improvement is possible, but not guaranteed, from any given generation to the next. Evolutionary al-

gorithms better suited to a given problem are more likely to make progress toward the optimal solution. Two-Archive significantly outperforms SPEA2, suggesting that MOEAs which maintain separate convergence and diversity archives may be better suited to sensornet protocol optimisation than those which maintain only a single archive.

We now consider the tradeoff between experimental cost and solution quality between the experimental approaches. Recall from section 4.2.8 that Factorial Design experiments require rsq^p simulation instances for each of Phase 1 and Phase 2. For these experiments this implies $6561 + 9000 = 15561$ simulation instances are required. We compare the experimental costs of the evolutionary approaches to this baseline figure.

We first consider SPEA2, which requires $a(b+1)rs$ simulation instances to be executed as per section 5.2.3.2. The best SPEA2-derived solution was obtained at generation 18, at which point 8550 instances had completed. This cost is 54.9% of the Factorial Design baseline. We conclude that significant cost advantage was observed for SPEA2 in tuning the TBG protocol if the tuning process is terminated at this point.

Now consider Two-Archive, also requiring $a(b+1)rs$ simulation instances to be executed as per section 5.2.4.2. Two-Archive produced a better solution than Factorial Design at generation 10 and produced its best solution at generation 11, corresponding to 4950 and 5400 simulation instances respectively. As these costs are 31.8% and 34.7% of the Factorial Design baseline, and better solutions were obtained, it follows that significant cost advantage is observed for Two-Archive in tuning TBG if the process is terminated at this point.

5.3.2 Optimised protocol tunings for IGF

Tables 5.4 and 5.5 gives summarised optimised tuning results for the IGF protocol. The sets of tuned protocol values corresponding to A, B and C are labelled I_A, I_B and I_C respectively. All figures for non-integral factors are given to 4 decimal places.

For each experimental approach, the set of values assigned to controlled factors X_1-X_6 corresponding to the highest quality solution discovered is given in table 5.4. For approach A some controlled factors were not evaluated directly in Phase 2 of the experiment. For these controlled factors, italicised in table 5.4, we take the midpoint of search ranges defined in sections 3.2 and 5.2.1.

	X_1	X_2	X_3	X_4	X_5	X_7	X_8
I_A	*6*	3	*0.5500*	10.0000	*6*	31.6667	0.1000
I_B	7	6	0.51566	6.5660	1	42.946	0.26423
I_C	7	6	0.16388	7.2508	1	46.939	0.66054

Table 5.4: Best known IGF tunings

Each IGF protocol tuning given in Table 5.4 was evaluated using the process described for the TBG protocol tuning solutions as described in section 5.3.1. Table 5.2 shows the Euclidean distances E_A, E_B and E_C from which I_A, I_B and I_C were identified as the highest quality solutions derived by approaches A, B and C respectively. Note that the theoretical perfect metric values are not necessarily attainable under *any* real protocol tuning. All figures for M_1-M_{3a} are give to 5 significant figures and scaled by a factor of 10^6 for clarity.

	M_1	M_{1a}	M_2	M_3	M_{3a}	E
O_A	122510	53.786	338380	72626	29.229	0.15701
O_B	292880	129.40	280390	58107	23.773	0.15694
O_C	82915	38.853	303150	62262	25.262	0.13918
Best	TA	TA	SPEA2	SPEA2	SPEA2	

Table 5.5: Network performance for best known IGF tunings

The protocol tunings shown in Table 5.4, have substantial differences in some controlled factors; notably, these are the factors identified as significant by the Factorial Design approach in section 5.2.2.2. The corresponding observed metrics given in Table 5.5 confirm that the network behaviour induced by each of the three approaches was significantly different, despite each approach being applied to the same protocol tuning problem.

Observe that the Two-Archive approach yielded the best result for metrics M_1-M_{1a}, the SPEA2 approach yielded the best result for metrics M_2-M_{3a}, and the Factorial Design approach did not yield the best result for any metric. It is therefore unsurprising that SPEA2 yielded the solution with best overall quality as measured in E, followed by Two-Archive and Factorial Design in order of declining quality.

Interestingly, however, the SPEA2 values for M_1-M_{1a} were an order of magnitude worse than those obtained under Two-Archive or Factorial Design, indicating that the SPEA2 values for M_2-M_{3a} were much better than those of Two-Archive in order to counterbalance this disadvantage. We conclude that the SPEA2 approach is superior for tuning the

IGF protocol, but that any of those approaches considered in this chapter would yield a reasonable solution.

Figure 5.2 plots normalised solution quality, E, versus evolutionary generation to illustrate the convergence of SPEA2- and Two-Archive-derived solution quality toward the final solution quality which could be achieved with an infinite number of generations. For comparison, the solution quality obtained by the Factorial Design approach is also shown as a constant.

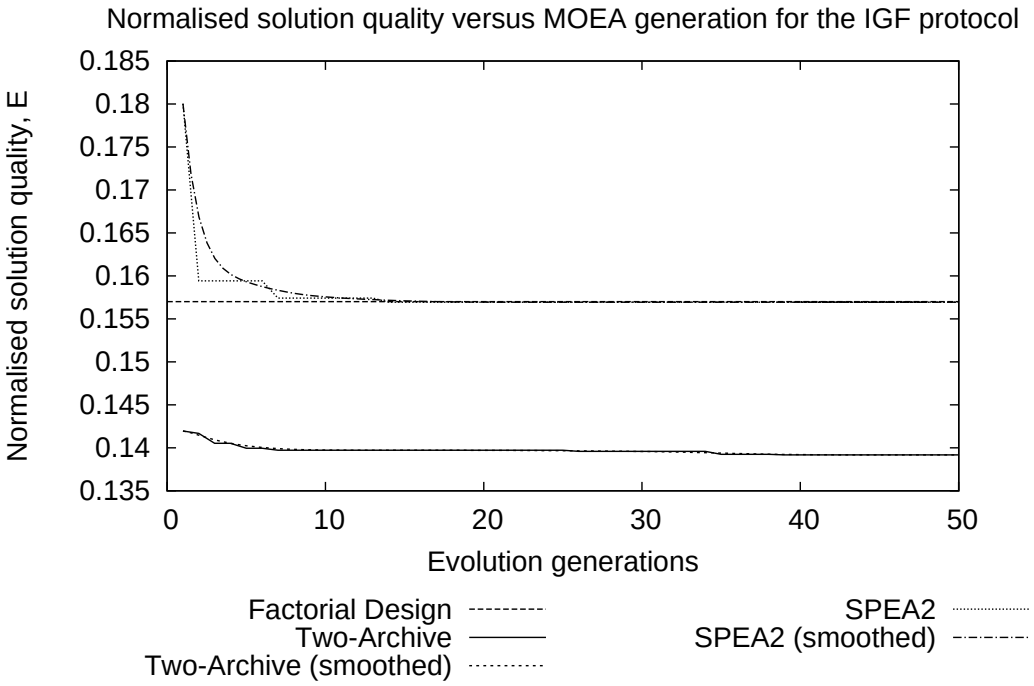


Figure 5.2: Comparing IGF solution quality using Factorial Design, SPEA2 and Two-Archive

The quality of solutions obtained by the SPEA2 approach improved quickly at first, but this improvement slowed quickly after the first few generations and converged on a very similar solution quality to that observed under Factorial Design; after generation 14 no further improvement was observed.

With Two-Archive the solution quality gradually improved until generation 7, after which no further improvement was observed. Interestingly, all Two-Archive generations showed a higher solution quality than Factorial Design or SPEA2. Although it might be considered somewhat fortuitous that the first evolved generation was of such high quality, this nevertheless illustrates that the evolutionary strategy of Two-Archive is effective for this problem type.

Now consider the tradeoff between experimental cost and solution quality between the experimental approaches. Recall from section 4.2.8 that Factorial Design experiments require rsq^p simulation instances for each of Phase 1 and Phase 2. For these experiments this implies $19683 + 21609 = 41292$ simulation instances are required. We compare the experimental costs of the evolutionary approaches to this baseline figure.

We first consider SPEA2, which requires $a(b+1)rs$ simulation instances to be executed as per section 5.2.3.2. The best SPEA2-derived solution was obtained at generation 14, at which point 6750 instances had completed. This cost is 16.3% of the Factorial Design baseline. We conclude that significant cost advantage was observed for SPEA2 in tuning the IGF protocol if the tuning process is terminated at this point.

Next, we consider Two-Archive, also requiring $a(b+1)rs$ simulation instances to be executed as per section 5.2.4.2. Two-Archive produced a better solution than Factorial Design at generation 1 and produced its best solution at generation 7, corresponding to 900 and 3600 simulation instances respectively. As these costs are 2.2% and 8.7% of the Factorial Design baseline, and better solutions were obtained, it follows that significant cost advantage is observed for Two-Archive in tuning IGF if the process is terminated at this point.

5.3.3 Comparative cost analysis

We now consider the relative costs of the tuning approaches. The costs of Factorial Design are described in section 4.2.8, the costs of SPEA2 are described in section 5.2.3.2, and the costs of Two-Archive are described in section 5.2.4.2. Note that in all cases the overhead of auxiliary calculations is orders of magnitude less than that of fitness function evaluation, so we discount the former in our analysis.

Assume each simulation instance completes in t seconds. The Factorial Design approach has wall time cost $C_\alpha = rsq^pt$. The SPEA2 and Two-Archive approaches have wall time cost $C_\beta = ars(b+1)t$. Given a single uniprocessor host, the SPEA2 and Two-Archive approaches will terminate before the Factorial Design approach if $C_\beta < C_\alpha$, a condition which is fulfilled where $a(b+1) < q^p$.

Now assume a multiprocessing environment in which x independent simulations can execute in parallel. For Factorial Design experiments there are no dependencies between simulations so any number can execute in parallel, all at cost t. The total wall time cost is $C_\gamma = \frac{rsq^p}{x}t$. Note that $C_\gamma \propto \frac{1}{x}$, reaching a minimum of $C_\gamma = t$ where $x \geq rsq^p$. For

SPEA2 and Two-Archive experiments it is possible to run all ars simulations of a given generation in parallel at cost $arst$, but all simulations of a given generation must complete before the next generation can begin. The total wall time cost is $C_\delta = \frac{ars}{x}(b+1)t$. Note that $C_\delta \propto \frac{1}{x}$, reaching a minimum of $C_\delta = (b+1)t$ where $x \geq ars$. If x is large then Factorial Design experiments will complete before SPEA2 and Two-Archive experiments.

We observe that the Factorial Design approach incurs a fixed wall time cost of $C_\alpha = rsq^pt$ regardless of solution quality, whereas the SPEA2 and Two-Archive evolutionary algorithms incur a wall time cost $C_\delta = \frac{ars}{x}(b+1)t$ such that experimenters can restrict cost by specifying the number of generations b. As the solution quality E is monotonically non-decreasing in b a tradeoff exists between cost and quality.

5.4 Quality of evolving solutions

Sections 5.3.1 and 5.3.2 show the potential savings in experimental cost which are possible by selecting an Evolutionary Algorithm approach over a Design Of Experiments approach to the protocol tuning problem. However, to achieve these potential savings, the experimenter must somehow determine the evolutionary generation at which the process is unlikely to yield further gains and hence should be terminated.

Figures 5.1 and 5.2 plot normalised solution quality, E, versus evolutionary generation to illustrate the convergence of SPEA2- and Two-Archive-derived solution quality toward the final solution quality which could be achieved with an infinite number of generations. For comparison, the solution quality obtained by the Factorial Design approach is also shown.

The E plots for the EAs observe a step function as quality E increases discretely between generations when a better solution is found and added to the archive. Smoothing the discrete step function curve into a continuous curve by considering individual data as *control points* of a Bézier curve, we observe that a hyperbolic curve is a reasonable approximation.

We apply statistical model fitting techniques using *MATLAB Curve Fitting Toolbox* [196] to quantify the relationship between evolutionary generation and solution quality as given by Equation 5.1. α, β, γ are constants for each combination of evolutionary algorithm and sensornet routing protocol considered in this chapter, and $g \leq b$ is the generation for which an solution quality estimate E_g is required. Results are given in table 5.6 to 4 significant figures. R^2 values indicate the quality of fit [38]; all fittings given in table 5.6

are of sufficient quality for the intended purpose.

$$E_g = \frac{\alpha}{g + \beta} + \gamma \tag{5.1}$$

Sensornet designers can estimate the evolutionary generations g required to find a protocol tuning solution of sufficient quality E_{req} such that $E_g \leq E_{req}$ by solving the inequality given by Equation 5.2 in g, though of course this does not actually yield the protocol tuning solution itself. If no positive real solutions for g exist then the evolutionary algorithm is not expected to find any solution of suitable quality in finite time.

$$g \geq \left\lceil \frac{\alpha}{E_{req} - \gamma} - \beta \right\rceil \tag{5.2}$$

This technique allows sensornet designers to estimate in advance the computational overhead implied in finding protocol tuning solutions of a given quality. A related technique can be applied to estimate the rate at which solution quality improves with respect to MOEA generation. Equation 5.3 gives the derivative of the fitted curve given by Equation 5.1 in terms of generation g.

$$\frac{dE}{dg} = -\frac{\alpha}{(g + \beta)^2} \tag{5.3}$$

As the MOEA progresses from generation to generation, the rate of change of solution quality can be estimated. The magnitude of this rate of change is relatively large at the start of the MOEA execution, but $\frac{dE}{dg} \to 0$ as $g \to \infty$ because E is monotonically non-increasing. At some point the rate of solution quality improvement will be sufficiently small that any further improvement is not significant, and hence the protocol tuning effort can be terminated.

Provided that the computational overhead of fitting a curve of form Equation 5.1 is small compared to the overhead $arst$ of assessing a MOEA generation as per section 5.2.4.2 and 5.2.3.2, it is feasible to implement this technique online during MOEA execution. Under this approach the MOEA assesses the solution quality derivative given by Equation 5.3 at the end of each generation; if the derivative magnitude is sufficiently large another MOEA generation executes, otherwise the process terminates at a point where no further significant solution quality improvement is likely within reasonable time.

An interesting hybrid approach would be to extend the MOEAs such that each candidate solution retained at each generation is taken as the centre of a small region of the

		α	β	γ	R^2
TBG	Two-Archive	0.3456	4.887	0.4588	0.7403
	SPEA2	0.01115	6.575	0.5064	0.8891
IGF	Two-Archive	0.006443	1.164	0.1392	0.8817
	SPEA2	0.004746	-0.7953	0.1569	0.9838

Table 5.6: α, β and γ for combinations of MOEA and sensornet protocol

parameter space which is sampled by Factorial Design methods as described in sections 4.2.3 and 4.2.4. The interpolation implicit in this approach would allow the MOEA to consider alternative candidate solutions that are close to, but potentially better than, those created explicitly by the mutation and crossover processes. This hybrid approach would of course increase experimental overhead with increased numbers of fitness function evaluations and auxiliary calculations implied by the MOEA itself and the model fitting.

5.4.1 Comparing evolutionary and non-evolutionary approaches

We see that both SPEA2 and Two-Archive yield high quality near-optimal solutions within a small number of evolutionary generations, with diminishing returns on further investments of experimental effort in the monotonically non-decreasing solution quality. Consider the experimental cost of the Factorial Design approach given in section 4.2.8, and that of the SPEA2 and Two-Archive approaches given in section 5.2.3.2 and 5.2.4.2 respectively. As the cost associated with the evolutionary algorithms is linear in generation count, limiting the number of generations also limits the cost.

Significantly, the SPEA2 and Two-Archive costs are independent of the number of controlled factors, unlike the Factorial Design cost. It follows that the merit of MOEAs becomes more apparent as the number of controlled factors grows and the cost of Factorial Design experiments becomes prohibitively large. It is the responsibility of the experimenter to decide whether it is appropriate to invest in the high monolithic cost of the Factorial Design approach, or the potentially lower but growing costs of the MOEA approaches. Provided that convergence on an acceptable solution occurs within a certain number of generations the overhead of MOEA approaches is lesser; in section 5.3.3 we calculate the critical point at which the MOEA approaches cease to offer cost benefits.

The Factorial Design approach samples the factor-response space evenly whereas the MOEAs focus computation resources on promising candidate solutions. The MOEAs tend

to gravitate toward a single solution whereas the Factorial Design approach provides an overview of the complete problem space which may encompass multiple good solutions, but if the experimenter requires only a single good solution this increased diversity is less important than the experimental cost.

We conclude that both evolutionary and non-evolutionary approaches offer benefit to the sensornet designer, but serve different purposes. The sensornet designer might usefully apply the Factorial Design approach to summarise the factor-response relationship to narrow the search space to those portions in which good solutions are known to reside, then apply evolutionary approaches to navigate any non-linear regions within this narrowed search space to obtain better near-optimal solutions.

5.5 Summary

The Factorial Design, SPEA2 and Two-Archive approaches described in sections 5.2.2, 5.2.3 and 5.2.4 all achieve results close to the theoretical optimum for the protocol optimisation problem described in section 4.1. The IGF protocol considered in this chapter represents the state-of-the-art; it is lightweight, geography-aware, stateless, and contemporary. In contrast, the TBG protocol is simplistic and potentially highly inefficient. Despite these qualitative differences, poor configurations of the state-of-the-art protocol were substantially outperformed by good configurations of the less sophisticated protocol in our experiments.

Our results demonstrate that, even if the network designer selects a state-of-the-art protocol such as IGF, selecting an appropriate configuration remains an open question. This important issue is regrettably ignored in many papers which describe new protocols, despite the potential impact on network applications employing these protocols.

Results presented in section 5.3 show that MOEAs can significantly outperform a simple Factorial Design experimental approach when tuning sensornet protocols against multiple objectives, producing higher quality solutions with lower experimental overhead. This is the first study in which sensornet protocol optimisation has been explicitly formulated as a multi-objective problem, and state-of-the-art multi-objective evolutionary algorithms applied in its solution. The Two-Archive algorithm outperformed the SPEA2 algorithm, at each generation and in the final evolved solution, for each protocol considered in this chapter.

Results presented in section 5.4 illustrate that the experimental cost of Factorial De-

sign experiment suites is fixed and independent of solution quality, whereas Evolutionary Algorithm approaches allow experimenters to manage the tradeoff between experimental cost and solution quality. Co-evolution of protocol designs and protocol tunings offers further scope for improved performance in future work.

Chapter 6

Protocol tuning robustness

Consider the deployment of a sensornet in the real world, where the network protocols have been tuned to perform well against a model of the deployment context. It is unrealistic to assume that the model is a perfect and complete model of reality. It is also unrealistic to assume that the modelled environment, or the composition of the sensornet itself, will necessarily remain constant. For example, if a sensornet is deployed to monitor seismic activity [319], the harsh conditions may cause nodes to fail or to move to a new geographic position. Additional nodes might be added to monitor areas of emerging importance.

It is therefore important to consider the *robustness* of a given protocol tuning to changes in network composition and deployment context. This chapter evaluates the extent to which the specific tuning of a protocol influences the observed performance attributes. This implicitly identifies the characteristics of a network in which the tuned protocol can function effectively, defining the desired outcome of infrastructure management approaches such as that defined in chapters 7 to 9.

6.1 Robustness analysis

In chapter 4 we define and implement a three-phase method to tune protocols for a specified sensornet environment. We now assess the extent to which the measured behaviour of a sensornet, in which the resulting tuned protocols are deployed, is *robust* to variation in node numbers, spatial distribution and spatial density.

Parameter tuning and robustness analysis are different but related tasks. In both parameter tuning and robustness analysis we define a system, measure its performance as one or more attributes of the system are varied, and examine the results to determine

whether acceptable system performance was observed. The primary difference is in the nature of the varied attributes. In parameter tuning we identify a set of controllable factors which represent design goals, are controlled factors which can be specified by sensornet designers, and hence for which we wish to find good values. In robustness analysis we consider those attributes of the sensornet which do not represent design goals of the parameter tuning process, and are therefore not controlled factors to be specified by sensornet designers, but which may nevertheless influence system performance.

The protocol tunings $I_{A\Xi}$ and $I_{B\Xi}$, obtained for network set Ξ under protocols A and B respectively and given in section 4.3, are employed throughout this section. Note that these protocol tunings were not optimised for robustness during the three-phase protocol tuning method described in section 4.2, but were derived using multiple networks which implicitly favours solutions which work well in a range of scenarios. All figures in this section normalise network metrics into the range [0,1] to allow multiple plots to be overlaid on the same axes for ease of comparison.

Ideally, we would prefer that protocols need not be tuned specifically for robustness as this implies a much greater experimental overhead. Each sampling point in the parameter space would have to be assessed many times across a diverse array of representative deployment contexts, in addition to the standard repetition described in section 4.2, which may be sufficient to render the problem intractable for sensornets of realistic scale. We would also prefer that robustness to variation in deployment context be achievable without compromising performance. If the network designer must tradeoff robustness against other other desirable properties then the design decision process becomes rather more complicated.

6.2 Network response metrics

In addition to the metrics $M_1 - M_3$ defined in section 3.3 we consider a further four metrics of network behaviour, $M_4 - M_7$, described below. Unlike $M_1 - M_3$ we do not use $M_4 - M_7$ in model fitting or assessing candidate solution quality in section 4.2. We include these additional metrics to demonstrate that robustness can be observed in network response metrics that were not considered during the protocol tuning process. We do not claim that these metrics embody some notion of robustness; instead, we aim to show that the values of these metrics are robust to variation in network deployment context.

Higher values of $M_4 - M_7$ imply more desirable behaviour, in accordance with the

natural interpretation of these metrics. These metrics pertain to *reachability*, the property that any given pair of nodes can exchange messages in a reasonably efficient and reliable manner. These metrics are particularly useful when building sensornet protocols with soft real-time requirements [121].

M_4**:** *Euclidian speed:* Mean physical speed with which a packet would travel from source to destination if it travelled in a straight line. Measured in m. Defined in the range $(0, \infty)$.

M_5**:** *Euclidian distance:* Mean physical distance separating source and destination for delivered packets. Measured in ms^{-1}. Defined in the range $(0, \infty)$.

M_6**:** *Path straightness:* The extent to which the actual path traversed by delivered packets conforms to the ideal straight path . Unitless. Defined in the range $(0, 1]$.

M_7**:** *PSEP:* The *Potential Saved Energy Proportion* [295] attainable under optimal state management policies. Unitless. Defined in the range $(0, 1]$.

M_4 is used to estimate whether a packet is likely to be deliverable within a given deadline. If a packet must be delivered from source S to destination D within t seconds, and can travel distance $d = tM_4$ within this deadline, then provided that $||\overrightarrow{SD}|| \leq tM_4$ the packet is likely to be deliverable. M_5 is used to estimate whether a packet is likely to be deliverable between a source and destination separated by a given physical distance regardless of deadline. As a network grows, the average distance over which packets are delivered should increase if the network is reliable; if it does not increase, the network is not equally reliable for all distances $||\overrightarrow{SD}||$. M_6 indicates the quality of routes discovered by the routing algorithm; a delivery path in a dense network that wanders far from the ideal path indicates that packets are reaching more nodes than is necessary. M_7, the *Potential Saved Energy Proportion*, measures the maximum proportion of energy expended by the sensornet which could be saved by a more efficient node state management policy without altering observed network behaviour traces [132]. Achieving this maximal energy saving would require perfect knowledge of current and future network traffic; this is impossible, of course, but better state management protocols get closer to this upper bound.

6.3 Spatial distribution

Recall from section 4.3.1 that all networks $\xi_\alpha \in \Xi$ contain 500 MICA2 motes with uniform random distribution within a square planar deployment region. In this section we consider whether a protocol tuning derived from a network of uniform spatial density remains appropriate if the intended deployment network is non-uniform.

We define two further sets of networks, U and G. Each of U and G is similar to Ξ except for the spatial position of nodes within the deployment region. For U we retain the uniform random spatial distribution of nodes, albeit with a different position for each node than in Ξ. For G we retain the randomness of the spatial distribution but discard the uniformity in favour of a Gaussian random spatial distribution. It follows that spatial density of networks in G is non-uniform, being much greater near the centre, but average spatial density remains unchanged and equivalent to networks in U.

Assume all sensornet motes are equivalent. Networks in set U provide near-equal sensor coverage of the entire geographic area covered by the sensornet, whereas networks in set G provide significantly greater sensor coverage near the centre. It follows that networks of type U are deployed where the entire covered area is of equal interest to the network operator, whereas networks of type G are centred around a single point of greater interest than the rest of the covered area.

Figure 6.1 gives plots for metrics $M_1 - M_3$ and figure 6.2 gives plots for metrics $M_4 - M_7$. For each M_α there are four bars in the bar graph. The values for each set have been normalised such that the largest value is 1 to facilitate comparison between bars in the set. It is meaningless to compare values across bar sets. The bars labelled *TBG-U* and *TBG-G* illustrate performance of networks of uniform and Gaussian spatial distribution respectively under the TBG protocol. The bars labelled *IGF-U* and *IGF-G* illustrate performance of networks of uniform and Gaussian spatial distribution respectively under the TBG protocol. To consider the impact of spatial distribution we compare the relative values of metrics measured under networks of different spatial distribution. In figures 6.1 and 6.2 this implies comparison of the U- and G-values for each combination of metric and protocol.

Firstly, consider the network response metrics $M_1 - M_3$ against which protocols were tuned, as illustrated in figure 6.1, where lower values indicate more desirable network behaviour. We observe that under the TBG protocol the measured values of $M_1 - M_3$ are very similar for networks of uniform and Gaussian spatial distribution. However, the same

can not be said for networks under the IGF protocol. The measured values of M_1 are very close, showing that network *performance* is unaffected. However, the measured values of $M_2 - M_3$ are significantly worse for Gaussian spatial distributions than for uniform spatial distributions, showing that network *reliability* and *efficiency* is affected adversely.

Secondly, consider the network reachability metrics $M_4 - M_7$, as illustrated in figure 6.2, where higher values indicate more desirable network behaviour. Again, we observe that under the TBG protocol the measured values of $M_4 - M_7$ are very similar for networks of uniform and Gaussian spatial distribution. Under the IGF protocol, $M_4 - M_6$ are very similar for both spatial distributions but significantly different for M_7 where the measured value is significantly worse for Gaussian spatial distributions than for uniform spatial distributions.

Tunings of the TBG protocol obtained using a training network of uniform spatial distribution yield similar performance in deployment networks of uniform and Gaussian spatial distribution. However, tunings of the IGF protocol obtained using a training network of uniform spatial distribution yield dissimilar performance in deployment networks of Gaussian spatial distribution. We conclude that the tuned TBG protocol is *robust* to this change of spatial distribution, whereas the tuned IGF protocol is not.

As these experiments were conducted with the best-known protocol tunings established by the factorial design experiments described in section 4.2.1 it is not possible to conclude that no tuning of these protocols exhibits superior *robustness*, although there is no evidence in favour or against the existence of any such superior tuning. However, it is appropriate to consider protocol tunings in this manner because we are assessing the robustness of protocols tuned to other criteria, rather than assessing the robustness of protocols tuned specifically for robustness. The latter would be at best pointless and at worst meaningless.

Note that consistency of measured behaviour across different spatial distribution does not imply that the observed behaviour is *good*; it merely implies it to be *consistent*. However, this consistency is of significance to network designers who may favour a *robust* solution over a *fragile* solution, even if the latter may yield better performance under ideal conditions.

6.4 Spatial density

In this section we consider the impact on network performance metrics as the uniform spatial density is varied. We measure metrics sets $M_1 - M_3$ and $M_4 - M_7$ in networks

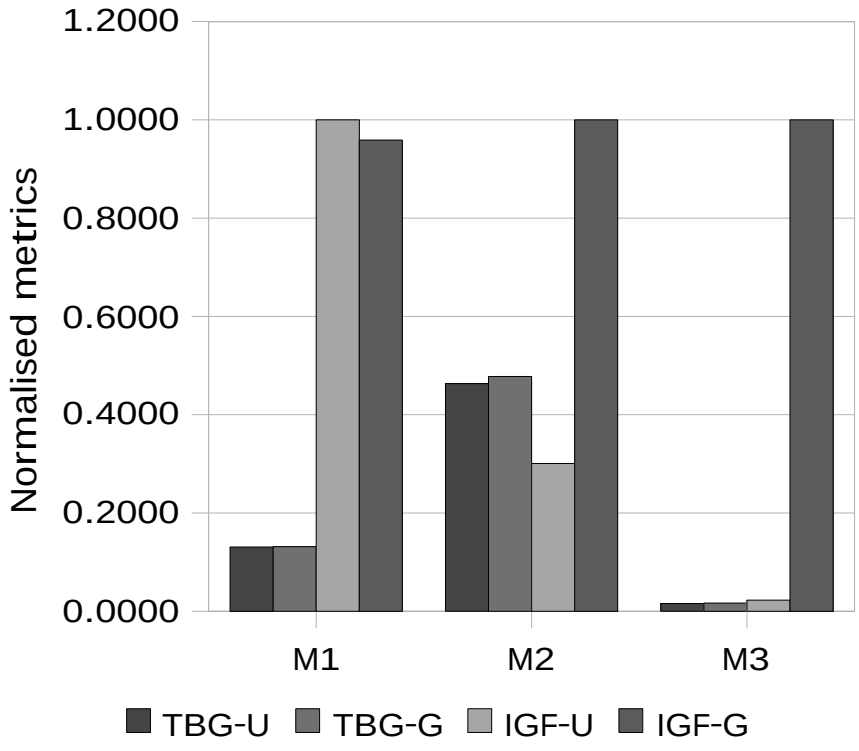


Figure 6.1: Network response metrics under controlled spatial distribution

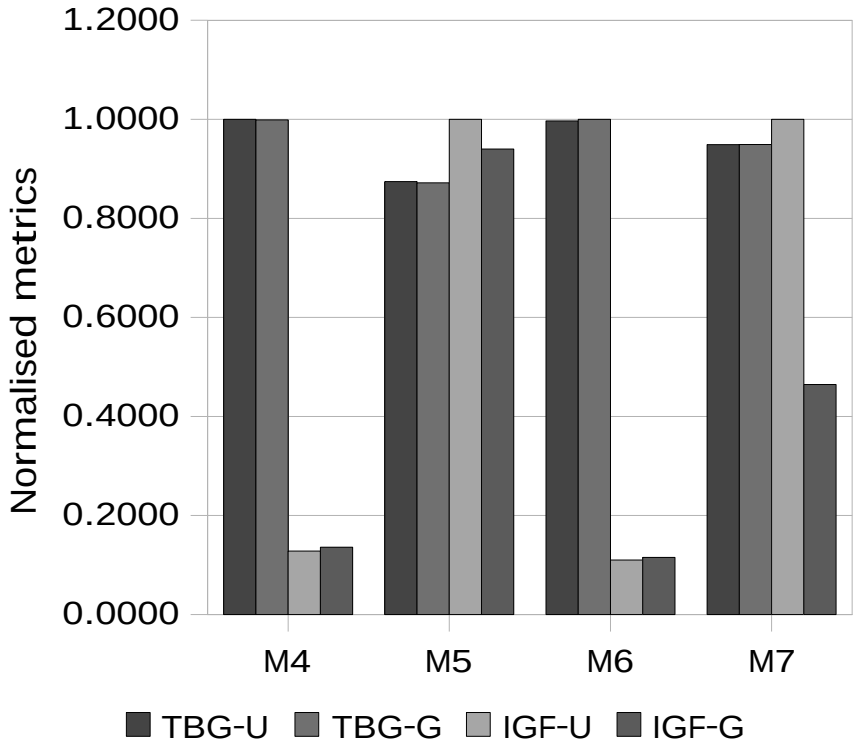


Figure 6.2: Reachability metrics under controlled spatial distribution

similar to $\xi_\alpha \in \Xi$ in which the size of the deployment region is altered such that a range of spatial densities are explored. In each case the relative spatial position of nodes within the deployment region remains unchanged as the size of the deployment region increases. By way of analogy, consider the manner in which dots drawn on the surface of a balloon move apart as the balloon is inflated but retain their relative positions. This prevents alteration of any underlying implicit structure in the network as the physical size of the network changes. The protocol tunings $I_{A\Xi}$ and $I_{B\Xi}$, obtained for network set Ξ under the TBG and IGF protocols respectively and given in section 4.3, are employed throughout this section.

Figures 6.3 and 6.4 summarise the relationship between controlled parameters and measured responses under the TBG protocol as spatial density d increases in the range $[5.66893 \times 10^{-8}, 5.66893 \times 10^{-6}]$ *node* m^{-2}. Likewise, figures 6.5 and 6.6 summarise the relationship between controlled parameters and measured responses under the IGF protocol. All values have been normalised to the range $[0, 1]$ to show the entire range of observed responses in a comparable manner.

Firstly, consider the network response metrics $M_1 - M_3$ for which lower values correspond to more desirable network behaviour. Figure 6.3 shows the relationship between metrics $M_1 - M_3$ and spatial density d for the TBG protocol. Figure 6.5 shows the relationship between metrics $M_1 - M_3$ and spatial density d for the IGF protocol. The density of the training networks is found at the centre of each x-axis.

Both protocols share similar relationships between d and M_1, and between d and M_3. As d increases we observe that M_1 increases, with the rate of growth also increasing with d. As density d increases the network *performance* falls, which is likely to be a consequence of high-density networks suffering substantial congestion and localised overloading from multiply redundant nodes attempting to perform the same task at the same time in the same spatial region.

These figures also show that, as d increases, the value of M_3 falls and converges on a stable minimum value. Network *efficiency* increases with spatial density d, suggesting that the negative effects of congestion and localised overloading are counterbalanced by positive effects; it may take longer for a packet to travel unit distance in the network, but it will take less energy to do so. This is likely to be a consequence of higher-density networks offering a greater number of forwarding candidates at each stage of packet delivery, and hence more likely to offer a high-quality candidate at each stage. This would seem to suggest

that selection of an appropriate next-hop forwarding candidate is highly significant at each decision point; it can not be assumed that the consequences of individual poor choices will be "smoothed out" along multi-hop delivery routes.

Dissimilar relationships exist between d and M_2 for TBG and IGF. For TBG the plot describes a monotonically non-increasing sigmoid shape, where network *reliability* improves with increasing density d. In contrast, the plot for IGF features a point of inflection at $d_c \approx 1.0 \times 10^{-6}$. For densities less than d_c, network *reliability* improves with increasing density. For densities greater than d_c, network *reliability* gets worse with increasing density. Assume that as d increases the number of neighbours increases uniformly for all nodes, with a corresponding increase in packet forwarding candidates. TBG will use any number of neighbours as packet relays at each stage, so increasing the number of neighbours will always increase the chance that at least one of these neighbours is a high quality forwarding candidate. In contrast, IGF selects only the single best packet relay from the set of all self-selected forwarding candidates. Although this approach reduces congestion from multiple transmissions of the packet, it creates the possibility of clashes between attempted *CTS* transmissions from forwarding candidates. A consequence is that as d increases, as does the possibility that the *CTS* from the best candidate will be lost in the short-term congestion.

We now consider the *robustness* of the protocol tunings by examining the relationship between d and metrics $M_1 - M_3$ either side of the training network density $d_t = 5.66893 \times 10^{-7}$ *node* m^{-2}. Firstly, we consider the TBG protocol. If d decreases then M_2 and M_3 take less favourable values, and M_1 takes more favourable values. Conversely, if d increases then M_2 and M_3 take more favourable values, and M_1 takes less favourable values. We therefore conclude that the TBG tuning $I_{A\Xi}$ is *robust* to decreasing d for *performance* properties, and is *robust* to increasing d for *reliability* and *efficiency* properties. Now we consider the IGF protocol. If d decreases then M_2 and M_3 take less favourable values, and M_1 takes more favourable values. Conversely, if d increases then M_3 take more favourable values, and M_1 takes less favourable values. M_2 is a little more complex in that as d increases the observed values will initially be more favourable, but if $d > d_c$ they will start to become less favourable. We therefore conclude that the IGF tuning $I_{B\Xi}$ is *robust* to decreasing d for *performance* properties, is *robust* to increasing d for *efficiency* properties, but is only *robust* for *reliability* properties within a finite band of possible values of d centred around d_c.

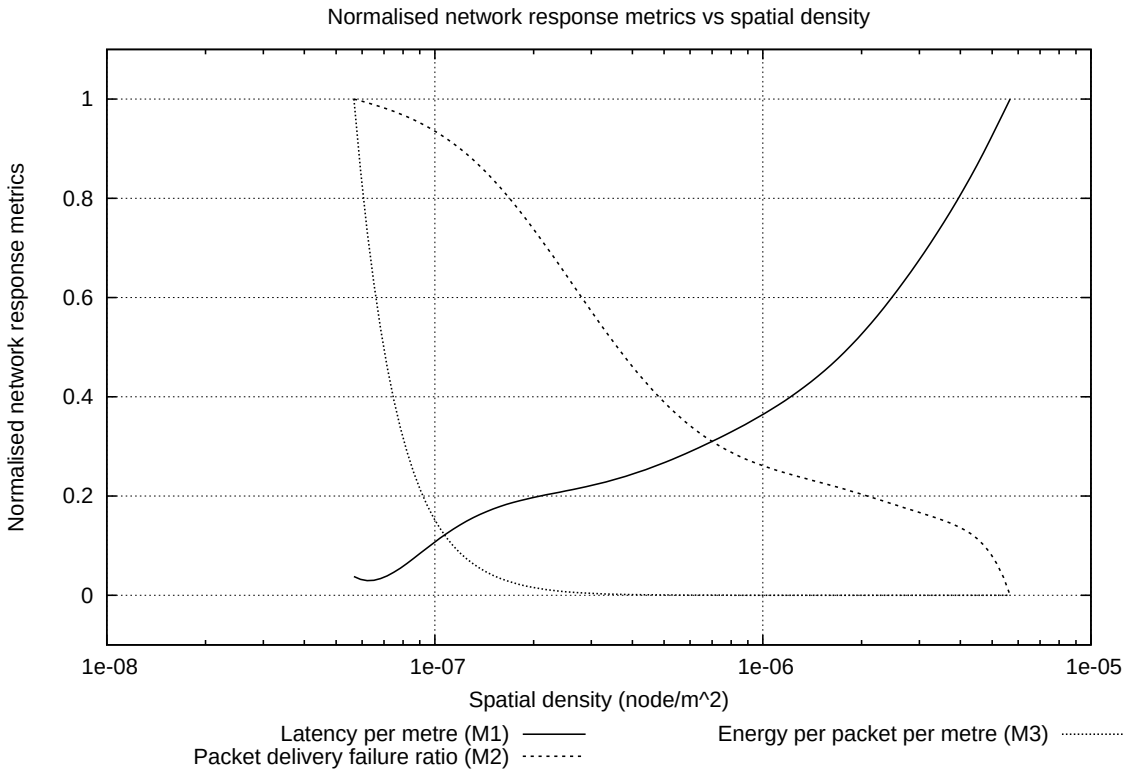


Figure 6.3: Network response metrics versus spatial density under TBG

For both TBG and IGF we observe that most variation in M_3 occurs toward the left of the plot with relatively little variation elsewhere. There is no value of d which minimises the value of all metrics and hence there is no ideal value of d. Therefore, if the spatial density of the deployment network can be controlled, the network designer must carefully tradeoff among these factors. However, if the IGF protocol is to be selected it is unlikely that a value of d greater than that at which the minimum of M_2 is observed would be selected, as for higher values of d both M_1 and M_3 increase rapidly and the decrease in M_2 is very slow.

Secondly, consider the network reachability metrics $M_4 - M_7$ for which higher values correspond to more desirable network behaviour. Figure 6.4 shows the relationship between metrics $M_4 - M_7$ and spatial density d for the TBG protocol, and figure 6.6 shows the relationship between metrics $M_4 - M_7$ and spatial density d for the IGF protocol.

The $d - M_4$ relationship is qualitatively similar for TBG and IGF, though quantitative differences are observed. The mean *Euclidian speed* is monotonically non-increasing, describing a sigmoid curve with a point of inflection around the density of the training network, d_t, where the rate of change of M_4 is relatively small compared to that distant

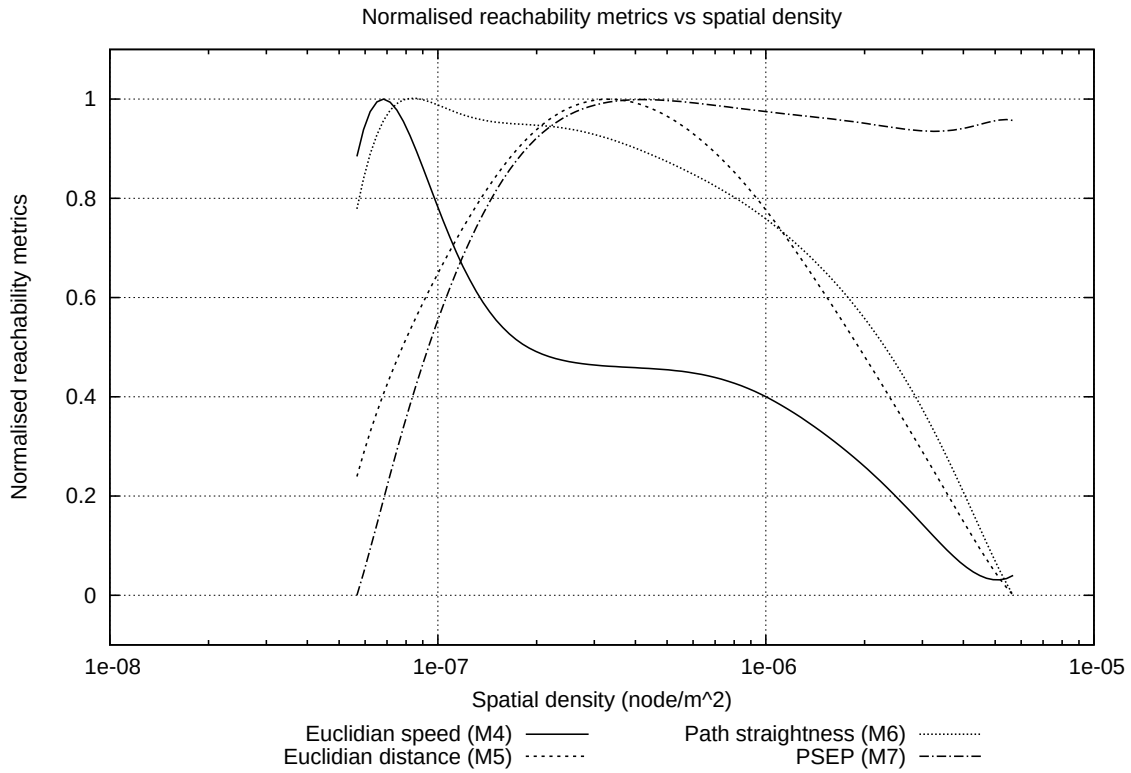


Figure 6.4: Reachability metrics versus spatial density under TBG

from d_t. A number of factors influence this relationship. As network density increases, the average physical distance between nodes tends to decrease. If, in unit time, a given packet traverses a given number of node-to-node hops, then the greater the spatial density the smaller the physical distance traversed. This leads directly to a lowered average *Euclidian speed* for delivered packets. The increase in localised congestion that results from increasing spatial density and the consequent growth in average degree of connectivity should also be taken into account.

Analysis of the $d - M_5$ relationship picks up where analysis of the $d - M_4$ reaches a conclusion. For networks of very low density the average degree of connectivity is also very low, such that packets are very unlikely to successfully traverse many node-to-node hops. For networks of very high density the average degree of connectivity is also very high, leading to high levels of congestion and a decrease in the average distance separating a randomly selected pair of source and destination nodes. It is consequently unsurprising that M_5 has a well-defined maximum with the observed value falling away quickly on either side. However, it is interesting to note that for both TBG and IGF this maximum is very near to d_t, suggesting that this metric is highly sensitive to the protocol tuning.

It is also interesting to note that the plots for $d - M_5$ have slightly different shapes for TBG and IGF, being near-Gaussian and near-parabolic curves respectively, although this difference may simply be an experimental artefact rather than anything more significant.

The $d - M_6$ relationship differs between TBG and IGF. Under TBG the relationship follows a sigmoid curve as per M_4 in which there is a point of inflection about d_t representing a range of densities around the density of the training network for which *Path straightness* is near-constant, but falls substantially quicker as the density diverges from d_t in either direction. Under IGF there is no well-defined point of inflection. Instead, M_6 is decreasing at all points of the curve, with the non-uniform rate of change being smallest for smaller densities and largest for larger densities. However, under both protocols the overall analysis is the same; as network density increases, the optimality of delivery routes decreases.

Finally, the $d - M_7$ relationship is very similar for both TBG and IGF. The *PSEP* value increases with d up to a point of inflection near d_t, and decreases slowly for $d > d_t$. For low spatial densities there is little redundant network capacity provision, and hence little scope to conserve energy through state management. However, once the network density reaches the minimum needed to support the best standard of network behaviour that can be attained (see metrics $M_1 - M_3$ in section 3.3) there is little to be gained by provision of additional redundant capacity; as capacity provision grows, so does the rate at which the operating network consumes this capacity.

We conclude the protocol tunings $I_{A\Xi}$ and $I_{B\Xi}$ are reasonably robust for metrics M_4 and M_6 for $d < d_t$, and for metric M_7 for $d > d_t$. This implies that these tunings are *fragile* for all other combinations of metric and network density d. It is interesting that points of inflection are often seen around the training network density d_t, suggesting that the choice of training network spatial density is significant in determining the *robustness* of the resulting protocol tunings to changes in spatial density.

6.5 Node count

In this section we consider the impact on network performance as the number of nodes participating in the sensornet is varied. We measure metrics sets $M_1 - M_3$ and $M_4 - M_7$ in networks similar to $\xi_\alpha \in \Xi$, altering aspects of the deployment network to keep all other controllable factors unchanged. All training networks $\xi_\alpha \in \Xi$ contain 500 nodes. For deployment networks containing fewer than 500 nodes we take a subset, implicitly

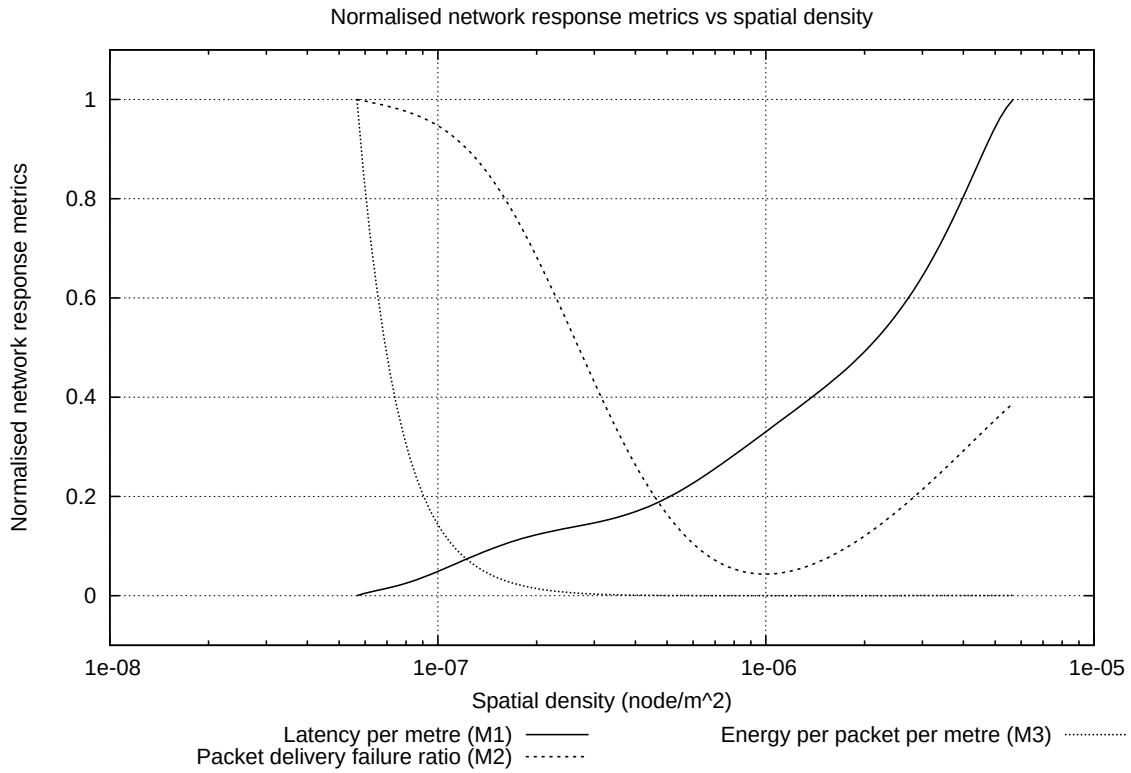


Figure 6.5: Network response metrics versus spatial density under IGF

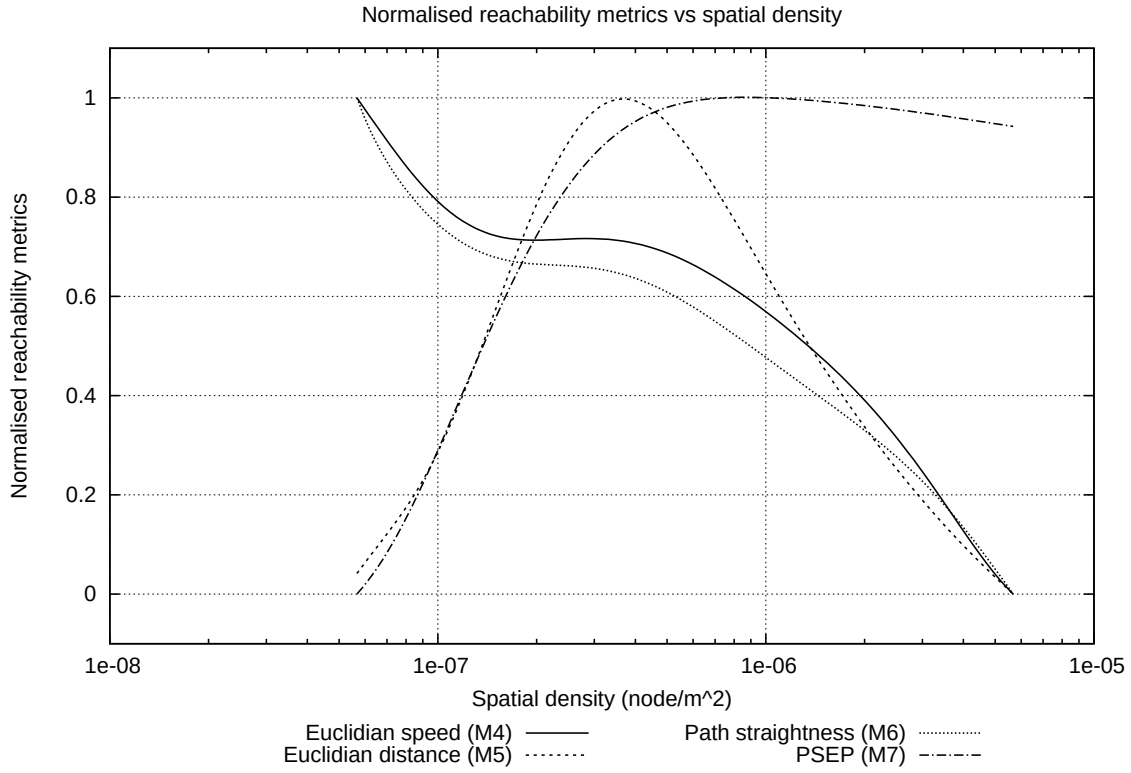


Figure 6.6: Reachability metrics versus spatial density under IGF

defining a subnetwork of similar spatial density and distribution but dissimilar node count and physical size. For deployment networks containing more than 500 nodes we extend the training network with additional nodes in the surrounding area or volume such that the latter is a subnetwork of the former. This is achieved by the same process described in section 4.3.3. The protocol tunings $I_{A\Xi}$ and $I_{B\Xi}$, obtained for network set Ξ under the TBG and IGF protocols respectively and given in section 4.3, are employed throughout this section.

Figures 6.7 and 6.8 summarise the relationship between controlled parameters and measured responses under the TBG protocol as node count n increases in the range $[50, 2000]$ nodes. Likewise, figures 6.9 and 6.10 summarise the relationship between controlled parameters and measured responses under the IGF protocol. All values have been normalised to the range $[0, 1]$ to show the entire range of observed responses in a comparable manner.

Firstly, consider the network response metrics $M_1 - M_3$ for which lower values correspond to more desirable network behaviour. Figure 6.7 shows the relationship between metrics $M_1 - M_3$ and node count n for the TBG protocol. Figure 6.9 shows the relationship between metrics $M_1 - M_3$ and node count n for the IGF protocol. The node count of the training networks is found at $x = 500$ on each x-axis.

Both protocols share similar relationships between n and metrics $M_1 - M_3$. As n increases we observe that metric M_1 increases rapidly from a low starting value where n is small, quickly reaching a maximum after which it varies only slightly. M_3 does the opposite, decreasing rapidly from a high starting point where n is small, quickly reaching a minimum after which it increases slowly. M_2 behaviour is different again, starting at a low value for small n and increasing gradually over the entire observed range.

Consider the $n - M_1$ relationship. As the number of nodes increases, as does the average distance between source and destination. If the spatial density remains constant then, as the average end-to-end physical distance grows, so does the average number of node-to-node hops required to deliver a packet from source to destination. If we assume that the delay encountered at each node does not decrease then the average end-to-end latency must necessarily increase too. This would remain true for any routing protocol operating in networks where individual nodes have fixed transmission ranges. However, an upper bound on end-to-end latency is defined explicitly as X_4, and implicitly as a consequence of X_2 and X_3.

Next, consider the $n - M_2$ relationship. Assume the average number of node-to-node

hops increases with n, that all node-to-node hops along a delivery route must succeed for delivery to succeed, and that every node-to-node hop implies some risk of failure. It follows that as n increases the probability of any given packet delivery attempt failing must also increase.

Now consider the $n - M_3$ relationship. Although we find that the possibility of any given candidate delivery route being unsuccessful increases with n, an effect running in parallel is that a greater number of nodes gives a greater number of candidate delivery routes. Because the number of candidate routes becomes larger, and we require only one of these candidate routes to successfully deliver a packet within defined QoS requirements, the effect of greater numbers of routes outweighs the effect of individual candidate routes becoming less dependable. It is more efficient to deliver a packet by a route of sub-optimal efficiency than to start delivering a packet by an optimal route but terminate prior to successful delivery.

We now consider the *robustness* of the protocol tunings by examining the relationship between n and metrics $M_1 - M_3$ either side of the training network node count $n = 500$ *nodes*. We address both TBG and IGF together. If n decreases then M_3 takes less favourable values, and M_1 and M_2 take more favourable values. We conclude that the TBG tuning $I_{A\Xi}$ and the IGF tuning $I_{B\Xi}$ are *robust* to decreasing n for *performance* and *reliability* properties. If n increases there is little change in M_1 and M_3, whereas M_2 takes less favourable values. We conclude that the TBG tuning $I_{A\Xi}$ and the IGF tuning $I_{B\Xi}$ are *robust* to increasing n for *performance* and *efficiency* properties. It is interesting to note that the set of properties to which the protocol tunings are *robust* is not the same for decreasing n as for increasing n.

For both TBG and IGF we observe that most variation in M_1 and M_3 occurs toward the left of the plot with relatively little variation elsewhere. There is no value of d which minimises the value of all metrics and hence there is no ideal value of n. If the number of nodes participating in the sensornet can be controlled, the network designer must carefully tradeoff among these factors. However, it would seem that for both TBG and IGF it would be desirable to select a reasonably small value of n just large enough that the *performance* and *efficiency* responses are in the *robust* region, and the *reliability* is as good as possible under this condition. The observation that the network designer should pick the smallest sensornet which is capable of supporting the proposed application is in keeping with other experimental results in the literature [202].

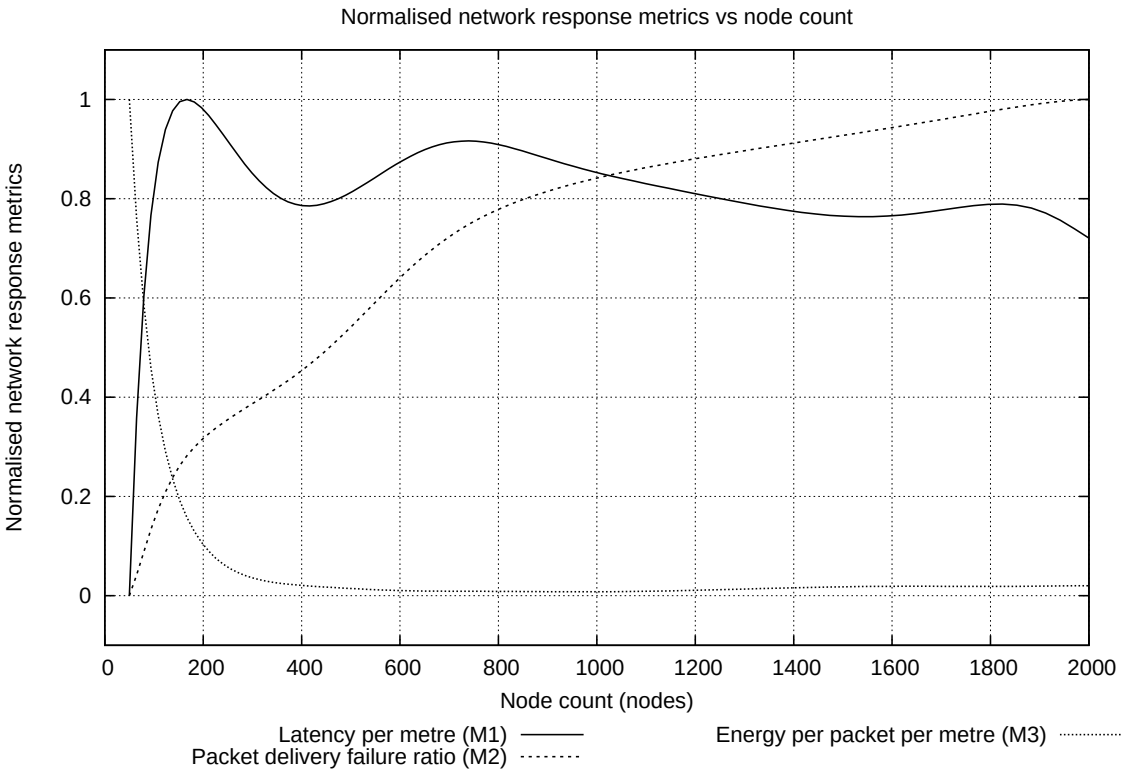


Figure 6.7: Network response metrics versus node count under TBG

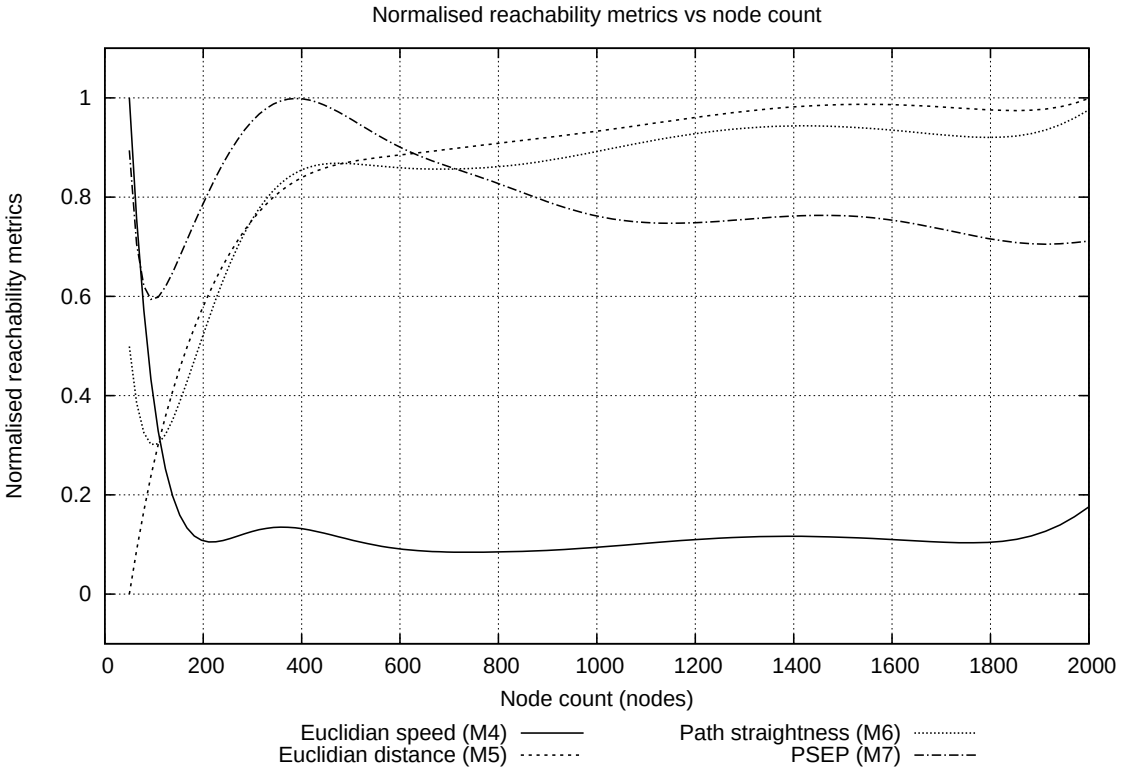


Figure 6.8: Reachability metrics versus node count under TBG

Secondly, consider the network reachability metrics $M_4 - M_7$ for which higher values correspond to more desirable network behaviour. Figure 6.8 shows the relationship between metrics $M_4 - M_7$ and node count n for the TBG protocol, and figure 6.10 shows the relationship between metrics $M_4 - M_7$ and node count n for the IGF protocol.

The relationship between M_4 and n is qualitatively similar for TBG and IGF, though quantitative differences are observed. The mean *Euclidian speed* increases with n until a maximum is reached, after which it varies only slightly around a constant value. We conclude that the protocol tunings are *robust* in M_4 in this near-constant region, but *fragile* for smaller values of n. The existence of this near-constant region suggests that an upper bound exists on the geographic region that is reliably reachable in unit time, with significant implications for distributed applications with real-time requirements.

The mean *Euclidian distance* M_5 between source and destination nodes of successfully delivered packets gradually increases with n across the entire range of network node counts considered in this chapter, though the rate of increase becomes less for higher values of n suggesting that an upper bound will eventually be reached. This is clearer for TBG, which has a clear point of inflection near the tuning network size $n_t = 500$, than for IGF but is evident for both. If n grows without bound in networks of fixed spatial density then the mean Euclidian distance between intended packet source and destination nodes will also grow without bound. However, the average Euclidian distance between *delivered* packets given by M_5 does not appear to grow without bound, suggesting that distributed applications in sensornets must be written with an awareness that successful pairwise communication becomes very unlikely as the distance separating the node pair increases.

A very similar $n - M_6$ relationship exists for the TBG and IGF protocols. Both show M_6 to grow rapidly with n until a point of inflection near the training network size n_t, at which point growth in M_6 levels off sharply to a near-steady value. This suggests that both protocols tend to deliver packets by somewhat sub-optimal routes in networks smaller than the training network, but deliver packets by routes consistently close to the optimal routes for networks larger than the training network.

Finally, we consider the $n - M_7$ relationship which appears substantially different for TBG and IGF. Under TBG the value of M_7 grows rapidly in n up to a point of inflection near the training network size n_t, then declines less quickly and appears to eventually converge on a steady value. Under IGF the value of M_7 falls consistently with growing n without any points of inflection near the training network size n_t.

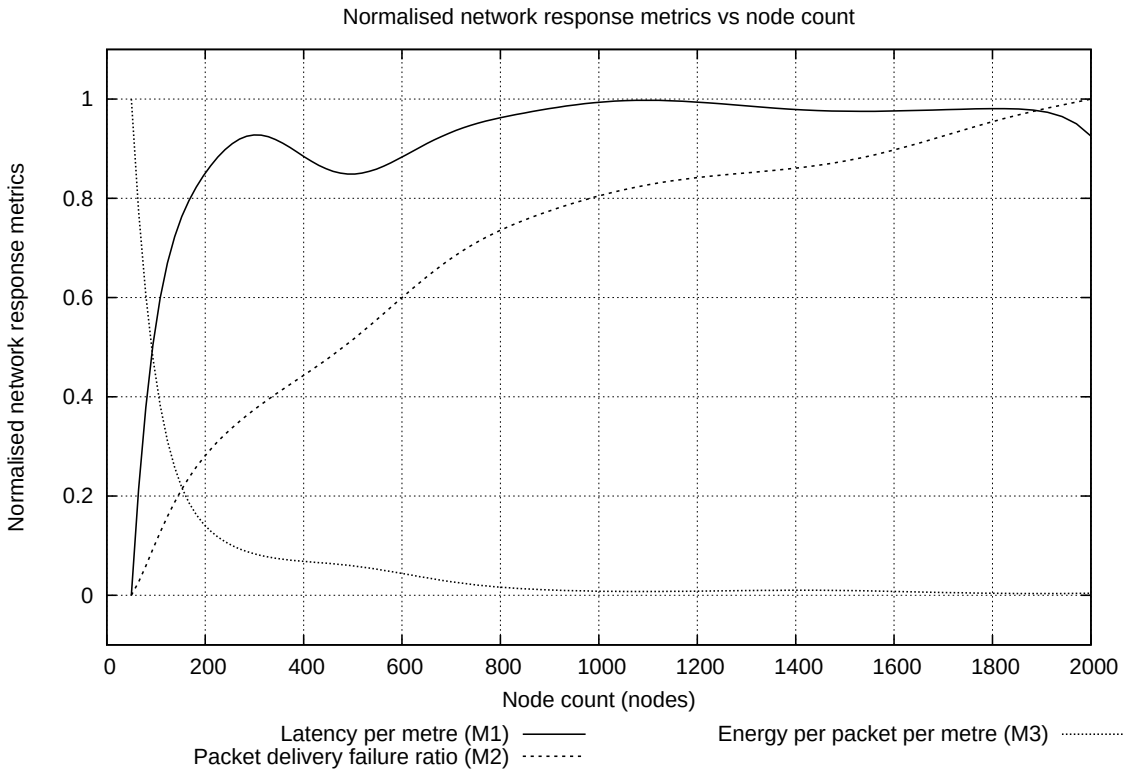


Figure 6.9: Network response metrics versus node count under IGF

We conclude that protocol tunings $I_{A\Xi}$ and $I_{B\Xi}$ are reasonably robust for metrics $M_4 - M_7$ under TBG and metrics $M_4 - M_6$ under IGF where $n > n_t$, and *fragile* for all other combinations of metric and network size n. It is interesting that points of inflection are often seen around the training network size n_t, suggesting that the choice of training network size is significant in determining the *robustness* of the resulting protocol tunings to changes in network size.

6.6 Summary

Section 4.3 demonstrated that a good protocol tuning solution derived in a given training environment may be similar, but not identical, to good protocol tuning solutions derived in different training environments. Results presented in chapter 6 show that a good protocol tuning derived in a given training environment is likely to perform effectively in different, but similar, deployment environments. However, as the deployment environment diverges from the training environment, this effectiveness may decline.

Sections 6.3 to 6.5 show that protocol tunings obtained in a given training environ-

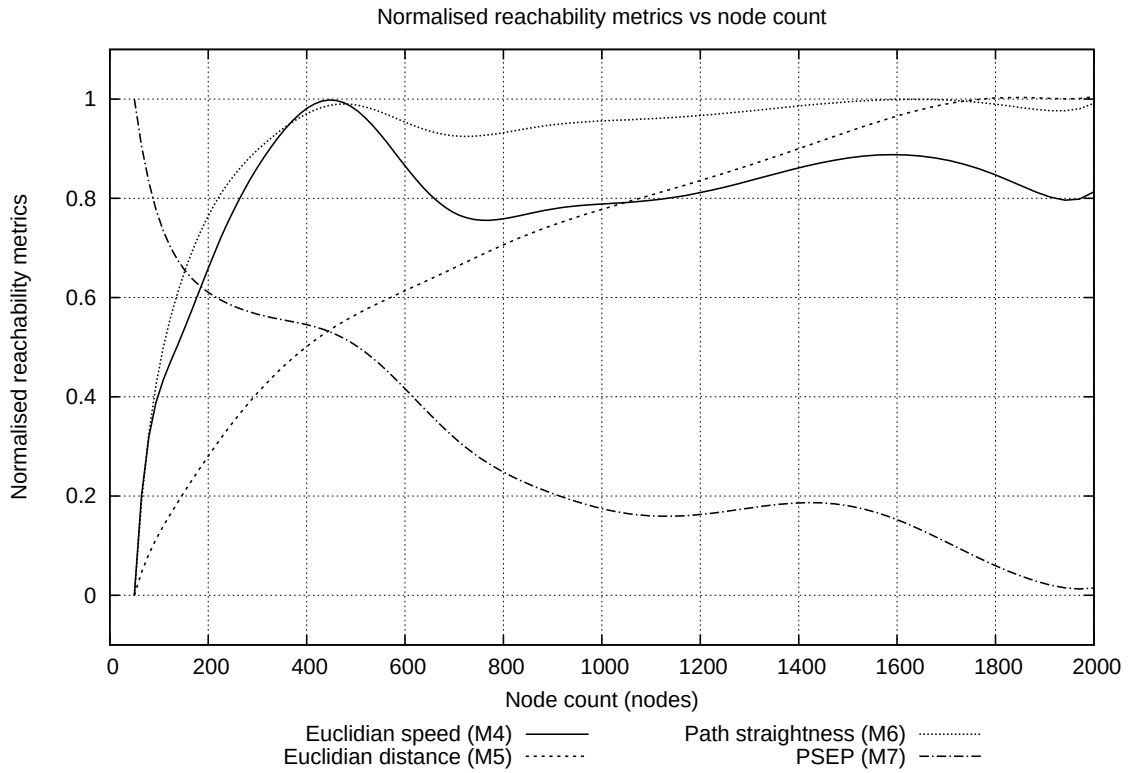


Figure 6.10: Reachability metrics versus node count under IGF

ment display differing levels of robustness to variation in the deployment environment. It is also shown that robustness can be measured for network response metrics which are not used in protocol training but nevertheless influence suitability for a given sensornet application. The IGF protocol tends to perform better than the TBG protocol under ideal circumstances, but the TBG protocol tends to offer greater robustness.

Chapter 7

Managing sensornets using cellular logical networks

The wireless communications channel is used for both application and infrastructure data exchange. Access to the shared medium is costly in renewable resources such as bandwidth, and costly in non-renewable resources such as battery power. By moderating wireless communication activity it is possible to conserve energy and enjoy improved network performance. This chapter considers lightweight and fault-tolerant protocols for synchronised node state management to maximise the useful lifetime of networks composed of unreliable nodes in a harsh environment. Unlike many network management approaches, these protocols offer an integrated approach which maximises the benefit derived from the implied costs, and minimises the overheads and network disruption implied by the management mechanisms.

7.1 Protocols for self-configuring sensornets

In chapter 3 we see that networks become harder to measure as they grow in size. In chapters 4 and 6 we see that suboptimal performance may occur when the deployment network is dissimilar to the training network, and performance tends to decline with increasing network size. Chapters 4 and 5 also demonstrate that, whereas protocol tuning can significantly improve performance, there are limits to the gains which can be achieved. If we cannot reduce the application requirements, and optimal protocol tunings cannot deliver the performance demanded by the application, we might instead reconfigure the deployment network. This allows its size to be minimised, addressing scalability issues of

protocols, and allows its characteristics to be controlled, addressing the unpredictability of deployment environments.

We assume we cannot change the physical environment, or the quantity or physical location of nodes after deployment. We can, nevertheless, control the size and structure of a logical virtual sensornet [144] by managing the duty cycle of individual nodes in the physical network. We can construct networks within which there is an adequate number and density of actively participating nodes in all physical regions to support distributed application requirements, without unnecessarily wasteful redundancy or duplication.

7.1.1 Managing sensornets over time

The expected lifetime of a given sensornet depends on its intended application. For a typical sensornet, system lifetimes in the order of months to years are typical [332]. This must be achieved using mote hardware platforms which may be capable of only a few days of continual operation [166]. This is obviously impossible unless the total number of individual motes is greater than that required to be simultaneously active to support the distributed application, and that some mechanism must switch motes on or off at appropriate times to maintain the required level of coverage. However, motes must nevertheless be available to extract sensor data from the physical environment at times implied by the physical phenomena of interest.

Sensornets must manage behaviour over a diverse range of timescales. Individual network packet transmissions may occupy periods of the order $10^{-3}s$ or less, and task scheduling decisions may consider much shorter periods for MHz- or GHz-clocked processors, whereas networks may be required to operate continually for periods of the order $10^{7}s$ if lifetimes extend into years. Different mechanisms are appropriate at different timescales, and as the duty management problem considers durations spanning at least 10 orders of magnitude it seems unlikely that a single mechanism could be optimal for all purposes.

We therefore give separate treatment to *short-*, *medium-* and *long-term* schedule management. We informally define *medium-term* tasks as being of the order of seconds to minutes, with shorter tasks of sub-second duration being *short-term* and larger tasks of hours or longer being *long-term*. This is based on the Burns-Hayes model of *time bands* [41], which is in turn based on Newell's model of *time bands* in human cognition [214]. We also use the Burns-Hayes notion of zero-length *events* as the foundation for synchronisation of *activities* and *behaviours* occurring within *time bands* (see section 2.6.2).

Our *short-term* band is equivalent to Newell's *biological band*, our *medium-term* band equivalent to Newell's *cognitive band*, and our *long-term* band equivalent to Newell's *rational* and *social* bands. This configuration places typical subsystem state transition latencies [81], and other platform-specific effects, into the *short-term* band, such that we need not consider these directly when considering behaviours within our *medium-* and *long-*term bands. This is similar to the low-level neural activity within Newell's *biological band*; essential for, but invisible to, higher-level *cognitive* activity. Our *medium-term* band supports instances of distributed sensing and processing behaviours, these being the sensornet equivalent of *cognition*. Our *long-term* band focuses on managing populations and ensuring fair distribution of work; parallels can be drawn between these issues and the *social* and *rational* aspects of Newell's model, although the equivalence is imperfect as we are considering computer- rather than human-based systems.

A more specific definition of timescales and bands requires ungeneralised knowledge of the intended application and deployment environment of a sensornet, which is beyond the scope of this document. Finer grained division of the time domain is possible, but this categorisation is sufficiently simple to avoid confusion and to provide a solid foundation for more elaborate schemes. Future work could, for example, subdivide the *long-term* band to provide targeted support for long-running or perpetually-running sensornet systems [145].

7.1.2 Synchronisation of time-sensitive behaviour

Sensornets are bound to the physical environment into which they are deployed, implying real-time requirements for the sensornet and the distributed application it supports [121]. Sensornet application designers must establish when network nodes are available to send, process or receive messages. This allows the cell to support distributed applications typically required in sensornets, such as distributed sensing applications or the provision of local storage redundancy to address node loss. Reliability is important in achieving probabilistic guarantees of real-time behaviour, as lost messages may imply delays, missed deadlines and wasted energy [6].

In a general network we might coordinate distributed behaviour through distributed scheduling and routing protocols, and to manage access to shared resources. This is typically difficult to achieve in a sensornet. Motes are generally equipped with the bare minimum resources required to support the distributed application. Lightweight protocols with probabilistic success guarantees are typically favoured over heavier but more reliable

alternatives [86,120]. No global clock exists, with distributed scheduling rendered difficult by many independent local clocks steadily drifting out of synchronisation [221].

7.1.3 LIPS: Lightweight Integrated Protocol Suite

Owing to the unpredictable nature of the physical environment and deployment methods, and imperfections in the reliability of network hardware, it will not suffice to construct a static logical network configuration. We must arrange for the network to continually self-monitor and self-manage to adapt to changing circumstances and cope with individual failures without compromising system success. The *Lightweight Integrated Protocol Suite* (LIPS) defines a set of protocols which cooperate harmoniously to achieve these goals. However, individual protocols can be applied in isolation if required for other network management purposes.

Firstly, we consider synchronisation in the time domain. Section 7.2 defines the *Lightweight Improved Synchronisation Primitive* (LISP), which produces a synchronised sequence of periodic timing events for coordination of time-sensitive distributed behaviour. LISP operates within a fully-connected set of nodes, each of which has its own unreliable timer but no global clock, such as a cluster or cell of sensornet motes. Chapter 9 defines the *Dynamic Cellular Accord Protocol* (DCAP) which synchronises multiple instances of LISP such that time-sensitive behaviour can be coordinated across hierarchical sensornets composed of multiple clusters or cells.

Secondly, we consider dynamically managed logical network configuration. A logical network of controlled size and physical density is created from a larger physical network, enabling lightweight protocols such as those considered in chapters 3 and 4 to function acceptably. This depends on the timing synchronisation provided by LISP and DCAP. Section 8.2 in chapter 8 defines the *Cyclic Duty Allocation Protocol* (CDAP) which constructs and maintains a distributed duty schedule within a closed system of active sensornet nodes, dividing responsibility between these nodes. Section 8.3 in chapter 8 defines the *Active Duty Control Protocol* (ADCP) which maintains the set of active nodes required by DCAP. A fixed number of nodes are selected at all times from a potentially larger pool of available nodes, with wear balancing implemented such that resource consumption is fairly distributed among all nodes but without excessive churn.

Dutta and Culler [81] identify the four main software techniques for energy efficiency improvement in sensornets as being *duty-cycling*, *batching*, *hierarchy*, and *redundancy re-*

duction. LIPS provides direct support for duty-cycling and redundancy reduction, as this is possible in an application-agnostic manner. Batching and hierarchy, under the definitions of Dutta and Culler, require application-dependent behaviour and are not directly exploited. However, these are not mutually incompatible with LIPS, and in fact could usefully be integrated by the sensornet designer.

7.2 Intracellular timing synchronisation

Consider a finite set of nodes with each node capable of broadcasting in a shared wireless medium, located such that they form a fully-connected network cell [44]. These broadcasts can be received by any member of the cell which is listening to the medium, or by any nearby external entities. Communication may occur with the cell, or with neighbouring cells and base stations to exchange data and tasking messages. We require some mechanism to synchronise time-sensitive activity where the sensornet application is distributed across multiple nodes.

Localised synchronisation protocols can support distributed applications that would otherwise fail without a global clock. We would like to arrange the timing of periodic events such that the delay between any two consecutive events is near-identical, and the ordering of events within each system epoch is identical. Under these conditions we can build protocols, such as CDAP [288] (discussed in section 8.2), upon this primitive which exhibit regular, predictable and fair allocation of duties. Long-term stability can be achieved despite imperfect clocks and connectivity.

7.2.1 LISP: Lightweight Improved Synchronisation Primitive

In this section we consider the elements of the *desynchronisation* primitive [226], and the properties of its converged equilibrium state. We use the standard definitions of pulse-coupled oscillator systems [201], but rephrase these from a global system viewpoint to a local node viewpoint as individual nodes do not have complete system knowledge.

We implement a lightweight feedback-driven primitive called LISP to build and maintain a cyclic duty schedule based on a variant of the biologically-inspired *synchronisation* phenomenon [201]. It has been observed that many naturally occurring systems will spontaneously self-synchronise [278]. This phenomenon has been extensively studied [201] and formalised [1]. However, the desired emergent synchronisation property requires all

interacting entities to share an appropriate set of behaviours. LISP defines behaviours appropriate to the domain of sensornets.

We show that this protocol works well under ideal network conditions, and propose improved versions that also perform well under adverse network conditions. We require that this protocol fulfils the following requirements:

- The protocol must reject timing error from clock drift and jitter.
- The protocol must cope with communications errors resulting in lost or spurious signals.
- The protocol must cope with dynamic mobile networks in which nodes join and leave cells unpredictably.
- The protocol must be sufficiently scalable to cope with network cell populations of realistic size.
- The protocol must make no assumptions about the low-level communications mechanisms employed by the network.
- The protocol must not assume nodes are implicitly aware of the behaviour or state of other nodes; all such knowledge must be acquired explicitly.
- The protocol must not require a global clock, or assume the existence of any mechanism for internode synchronisation of local clocks.

7.2.2 Building blocks

Assume we have a set Σ of nodes $S_1 \cdots S_n$ where $n \geq 2$; if $n = 1$, there is obviously no need for internode coordination. Each node S_i acts independently but shares an identical set of behavioural rules. The running time of the system is divided into a set of system *epochs* of equal period e such that $\forall j : E_j = e$. The sequence of system epochs E_j is defined by the natural ordering of $j \in \mathbb{N}$.

Within each system epoch E_j it is required that each node $S_i \in \Sigma$ shall execute a single instance of a periodic event V_i exactly once. All events V_i are periodic with identical period $p_i = e$. The occurrence of a specific event at a specific node i within a specific system epoch j is labelled V_{ij}. It is required that all events V_{ij} are executed within epoch E_j. These events need not be related to any functionality of the sensornet

application. However, if the application naturally produces periodic events of this type, perhaps as part of a distributed sensing function, then these application events can be reused for synchronisation.

Distributed protocols and applications can use the resulting stream of observable synchronisation events, occurring every t time units, as the foundation for coordinated activity. Periodic application events required to occur with frequency $f = 1/t$ can be triggered directly by observed synchronisation events. Application events specified at other frequencies may use harmonics of the synchronisation frequency f; other arbitrary relationships can be supported.

Between observed events nodes must use a local clock. During this period it is possible that the local timer of each node may drift by varying amounts, until the next observed event corrects the effects of this drift. However, it is reasonable to assume that commodity timers based on quartz crystals offer acceptably small and predictable drift between observed events [262]. Typical drift rates of 1×10^{-6} *seconds per second* [64] are very small compared to timings of the real-world phenomena of interest to sensornet operators. The impact of clock drift is evaluated in section 7.2.10.5.

7.2.3 Equilibrium state properties

For a *desynchronised* [226] system in a stable state, the ordering of events V_i is stable from epoch E_j to E_{j+1} and the elapsed time between any two consecutive events is equal to e/n. This is similar to *stable incoherence* under the Kuramoto model [1] with *order parameter* ≈ 0. A stable state conformant to these specification is known as an *equilibrium state*; as time is continuous, if there exists at least one equilibrium state there exists an infinite number of such states. Fortunately, all equilibrium states are equivalent and equally acceptable. Before the system reaches the equilibrium state it is possible that the inter-event time can change; when the equilibrium state is reached, it can not. The specific stable ordering is unimportant, though it is a deterministic consequence of the initial state of the system and the set of shared rules, but the inter-event time t is always $t = e/n$.

Although we have defined that the period of all events V_i is equal such that $\forall i : p_i = e$, we do not explicitly define the offset o_i of each periodic event V_i within a stable epoch; this is a deterministic consequence of running a coordination protocol based on the *desynchronisation* primitive as described below. The order of offsets o_i within an epoch E_j defines the order of events V_{ij}, but any ordering offers equivalent coordination behaviour

within the network cell.

As each epoch E_j is of equal length e, and each event V_i is periodic, for a given epoch we can define the *phase* of each event relative to the epoch start. If t_{ij} is the time from the start of epoch E_j to event V_{ij} then phase $\psi_{ij} = t_{ij}/e$. For a stable system the identity of the epoch is not relevant, so $\psi_i = t_i/e$. This gives phase measurements defined in the range $\psi \in [0, 1)$. Any value $\psi \notin [0, \psi_{max})$ is equivalent to $\psi \bmod \psi_{max}$ as a consequence of modular arithmetic inherent in phase calculations. Equivalent behaviour is observed if all values of ψ are scaled linearly with maximum phase ψ_{max} taking some arbitrary real value, so we will use ψ_{max} in the analysis but use the explicit $\psi_{max} = 1$ when presenting experimental results.

If we now consider the inter-event time t in terms of phase, we find that the phase difference between any two consecutive events V_x and V_y is $\Delta\psi = (e\psi_{max})/n$. To achieve this equal $\Delta\psi$ we must schedule the events V_i evenly in time throughout an epoch. This schedule must also ensure a margin of $\Delta\psi$ exists between the last event of the previous epoch and the first event of the given epoch, and between the last event of the given epoch and the first event of the following epoch. Within a given epoch the time before the first event and the time after the last event must sum to $\Delta\psi$ to provide sufficient margin.

Conversely, under *synchronisation* we would require that all periodic events occur simultaneously within each epoch. This is a particular form of *stable coherence* under the Kuramoto model [1] with *order parameter* ≈ 1. Whereas this would also be usable as the foundation for coordinated distributed activity, the duration between observable synchronisation events would be n times longer than under *desynchronisation*. This would increase the risk of errors deriving from clock drift between synchronisation events and other timing inaccuracies, but offers no saving in energy consumption or overhead. Given that the timing resolution defined by *synchronised* events would be coarser, and there would be no energy cost saving, there is little to recommend the *synchronisation* strategy over the *desynchronisation* strategy.

7.2.4 Attaining equilibrium state

Sections 7.2.2 and 7.2.3 describe the system from the viewpoint of an external observer with access to the entire system. Now consider the viewpoint of a participating node S_i which can observe events occurring at other nodes but has no other information. Each node tracks the passage of time using its internal clock, corresponding to a local measure

of phase ϕ_i in the range $[0, \phi_{max})$ where $\phi_{max} = \psi_{max}$ as given above. Each node S_i applies the algorithm independently, so we can define this algorithm using only locally-available data and assume that each participating node executes the same algorithm in parallel.

The difference between ψ and ϕ is that ψ gives a system-wide measure of the passage of time as measured in phase units, whereas ϕ_i gives the local measure of the passage of time as experienced by a single node S_i. This is significant because each node S_i does not have omniscient access to information available to any other node, and does not have access to any system-wide overview. As protocol designers we can use system-wide information to measure the effectiveness of a network design, but the nodes upon which the protocols are implemented have access only to information learned from their environment.

Consider an arbitrary epoch E_j. When $\phi_i = \phi_{max}$ the event V_i is triggered at node S_i and ϕ_i is reset to 0. Each node S_i is aware of the time at which its own event V_i executes, and the times at which the instantaneously preceding and following events $V_{i\beta}$ and $V_{i\gamma}$ occur. The node S_i does not know, and does not need to know, the identity of the other nodes $S_{i\beta}$ and $S_{i\gamma}$, the *phase neighbours* of S_i, at which $V_{i\beta}$ and $V_{i\gamma}$ occur respectively. However, S_i will influence and be influenced by its phase neighbours.

Assume a node S_i executes event V_i, and observes preceding event $V_{i\beta}$ and succeeding event $V_{i\gamma}$ which may or may not occur in the same epoch E_j. Node S_i measures the duration $t_{i\beta}$ between $V_{i\beta}$ and V_i, and the duration $t_{i\gamma}$ between V_i and $V_{i\gamma}$, using its internal clock. We convert these timings into relative phase differences as $\phi_{i\beta} = -(t_{i\beta}/\phi_{max})$, and $\phi_{i\gamma} = (t_{i\gamma}/\phi_{max})$. Note that $\phi_{i\beta}$ is negative as the predecessor phase neighbour event $V_{i\beta}$ must occur before V_i, but is nevertheless equivalent to the positive value $(\phi_{i\beta} \bmod \phi_{max}) \in [0, \phi_{max})$. This also implies that the successor phase neighbour event $\phi_{i\gamma}$ necessarily occurs in a different local epoch than the predecessor phase neighbour event $\phi_{i\beta}$.

In the equilibrium state described in section 7.2.3, all events V_i will be equidistant between preceding event $V_{i\beta}$ and succeeding event $V_{i\gamma}$. We define *phase error* for node S_i as $\theta_i = \phi_{i\beta} + \phi_{i\gamma}$, which is the phase amount by which the timing of event V_i differs from the desired stable state value. When a equilibrium state is attained, $\forall i : \theta_i = 0$.

The phase error for node S_i can also be found as $\theta_i = (t_{i\gamma} - t_{i\beta})/\phi_{max}$ by substituting the definitions of $\phi_{i\beta}$ and $\phi_{i\gamma}$ given above; this alternative notation is equivalent but may be easier to implement directly where nodes sleep for periods during which local phase ϕ_i is not monitored.

As soon as node S_i becomes aware of succeeding event $V_{i\gamma}$ during each epoch, node S_i

can execute the phase adjustment procedure. Recall that node S_i has an internal clock which it uses to maintain a measure of local phase ϕ_i. Node S_i evaluates its phase error θ_i when succeeding event $V_{i\gamma}$ is observed. We now use θ_i to adjust ϕ_i by the *phase change* amount $\Delta\phi_i$, which will either enlarge or contract the duration until the next execution of event V_i. This is achieved by immediately setting $\forall i : \phi_{i_{new}} = \phi_{i_{old}} + \Delta\phi_i$. Note that this $+\Delta\phi_i$ adjustment must also be applied to any phase measurements of other events stored within node S_i to ensure they retain their position relative to the local synchronisation event.

We define $\Delta\phi_i = -f_\alpha\theta_i$ where $f_\alpha \in (0, 1]$ represents the *feedback proportion*. Higher f_α values give faster convergence but less stability, whereas lower f_α values give a system which takes longer to reach an equilibrium state but is more stable to the deleterious effects of noise.

This local phase correction directly changes the behaviour of node S_i and indirectly changes the behaviour of phase neighbours $S_{i\beta}$ and $S_{i\gamma}$; during the following epoch all events V_i will be closer to their equilibrium-state equilibrium phase ψ_i. Given an otherwise unchanging network, $\forall i : |\Delta\phi_{ij}| \to 0$ as $j \to \infty$ in successive epochs [201]. If $\theta_i = 0$ then the phase change $\Delta\phi_i = 0$ as well; no special action needs to be taken. Note that systems implementing this primitive may be sufficiently converged to support useful application work before reaching full convergence.

Algorithm 1 defines the primitive behaviour executing at each node $S_i \in \Sigma$ under the original version of the primitive. Variables not defined within the algorithm itself take the standard meanings used elsewhere in this document.

Figure 7.1 illustrates the feedback mechanism in which the synchronisation transmission time of a given node influences, and is influenced by, its immediate timing neighbours. The line running left to right represents the progression of time within a system epoch. Each circle represents the synchronisation event associated with an individual node, fired at some time within the epoch. In subsequent epochs a similar pattern is observed, with node synchronisation events firing in the same order but potentially separated by different delays. All nodes are strict peers; no control hierarchy or precedence exists within a cell of peers. In figure 7.1 we consider one such node in detail, but the process described is implemented by all peers.

Consider the central filled black circle, representing the synchronisation event fired by

Algorithm 1 : Original primitive variant A at node S_i

Require: Observed predecessor sync phase, $\phi_{i\beta} = nil$
Require: Observed successor sync phase, $\phi_{i\gamma} = nil$
while monitoring local phase ϕ_i increasing over time **do**
  if sync event $\neq V_i$ observed **then**
    if $\phi_{i\gamma} = nil$ **then**
      $\phi_{i\gamma} \Leftarrow \phi_i$
      if $\phi_{i\beta} \neq nil$ **then**
        $\theta_i \Leftarrow \phi_{i\beta} + \phi_{i\gamma}$
        $\Delta\phi_i \Leftarrow -f_\alpha\theta_i$
        $\phi_i \Leftarrow (\phi_i + \Delta\phi_i) \bmod \phi_{max}$
        $\phi_{i\gamma} \Leftarrow (\phi_{i\gamma} + \Delta\phi_i) \bmod \phi_{max}$
      end if
    else
      $\phi_{i\gamma} \Leftarrow \phi_i$
    end if
  end if
  if $\phi_i \geq \phi_{max}$ **then**
    if $\phi_{i\beta} = nil$ **then**
      $\phi_{i\beta} \Leftarrow \phi_{i\gamma}$
    end if
    $\phi_{i\gamma} \Leftarrow nil$
    $\phi_i \Leftarrow 0$
    fire own sync event V_i
  end if
end while

node S as its local phase reaches ϕ_{max}. Node S has just observed the synchronisation event of its successor phase neighbour at ϕ_γ, and has earlier observed the synchronisation event of its predecessor phase neighbour at ϕ_β. The phase neighbour synchronisation events are represented by the grey filled circles to the left and right of the S synchronisation event.

Observe that the S synchronisation event at ϕ_{max} is located some distance from the midpoint of ϕ_β and ϕ_γ, as the *LISP* primitive has not yet reached equilibrium. Node S changes its local phase by $\Delta\phi_S = -\alpha(\phi_\gamma - \phi_\beta)$, in this case pushing the timing of this event towards that of the predecessor phase neighbour.

Note that similar activity is induced in the phase neighbours of S. The predecessor phase neighbour of S observes the node S synchronisation event as its own successor. The successor phase neighbour of S observes the node S synchronisation event as its own predecessor. Owing to the timing differences illustrated in figure 7.1, the predecessor phase neighbour's synchronisation event time is pulled toward that of node S, and the successor phase neighbour's synchronisation event time is pushed away from that of node S.

The ordering of synchronisation events within system epochs cannot change from one epoch to the next because the local phase adjustment can never exceed the midpoint $\phi_\gamma - \phi_\beta$ as $\alpha \leq 1$. Allowing $\alpha > 1$ would offer no improvement in time required to reach

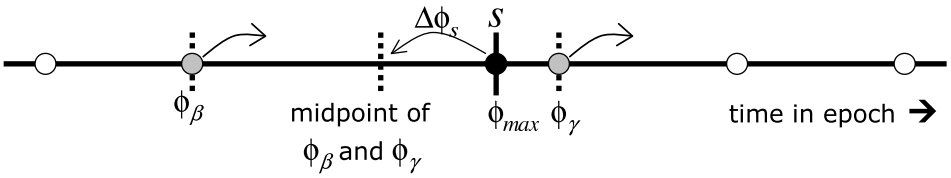


Figure 7.1: Influence of peer synchronisation timing feedback

equilibrium. It could, however, lead to unstable systems with timing behaviour that is deterministic but difficult to predict, as unstable event ordering between epochs prevents convergence.

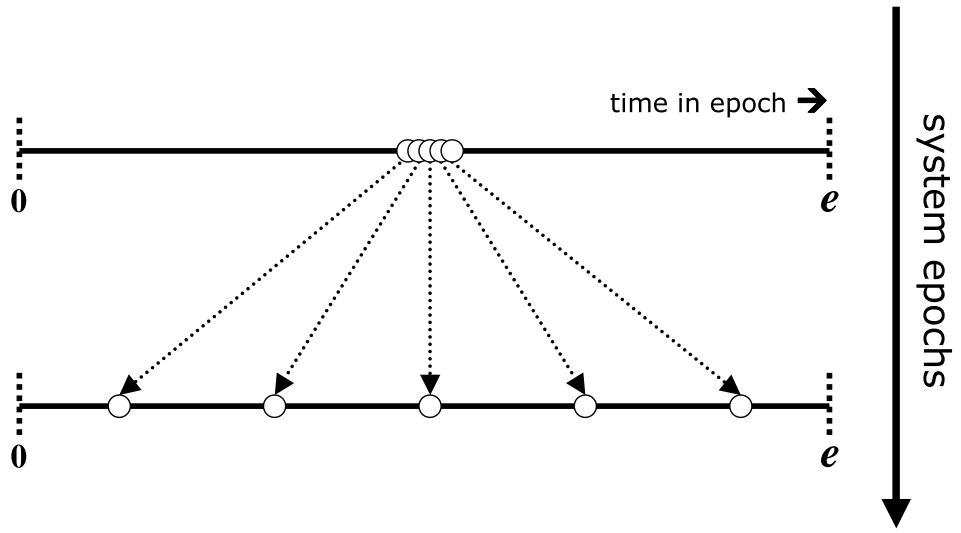


Figure 7.2: Worst case initial state transitioning to equilibrium state

Figure 7.2 illustrates the cumulative effect of this mutual influence for a system starting in the *worst-case* initial state, in which all nodes have very similar local phase. The feedback mechanism rapidly pushes the local phase values apart as the system approaches the equilibrium state, in which events are distributed evenly throughout time in each epoch. It is obvious that this equilibrium state also represents the *best-case* initial state. Note that under a *synchronisation*-based algorithm, as opposed to this *desynchronisation*-based algorithm, the definitions of *worst-case* and *best-case* initial state are reversed.

7.2.5 Measuring solution quality

Recall from section 7.2.3 that upon reaching a equilibrium state the set of events has an even temporal distribution. For a given node S_i we know that when local phase $\phi_i = 0$ the event V_i is exactly equidistant from both V_β and V_γ, and we know that the relative phase difference between V_β and V_γ is given by $2(\phi_{max}/n)$. It is therefore possible to measure the observed behaviour against this defined ideal to obtain estimates of solution quality at any given instant.

Each node can calculate these metrics using locally available data, perhaps using these to moderate local application behaviour. Ideally, all nodes would have the ideal value of all metrics.

M_1: *Allocated timeslot length.* In the equilibrium state each node is allocated communication duty for an equal proportion of each epoch. The metric is calculated for each node S_i as $M_{1i} = t_{i\beta} + t_{i\gamma}$ and is measured in *seconds*. The ideal value is $M_1 = e/n$.

M_2: *Asymmetry.* In the equilibrium state each node broadcasts its synchronisation pulse exactly equidistant from those of its phase neighbours with perfect symmetry. The metric is calculated for each node S_i as $M_2 = |t_{i\beta} - t_{i\gamma}|$ and is measured in *seconds*. The ideal value is $M_2 = 0$.

M_3: *Node population estimate.* In the equilibrium state each node has sufficient information to accurately estimate the cell population, and hence to decide whether it should participate. The metric is calculated for all nodes as $M_3 = [e/(t_{i\beta} + t_{i\gamma})]$, and is measured in *nodes*. The ideal value is $M_3 = n$.

7.2.6 Synchronisation transmissions

Networks in which this primitive is applied can be modelled as a fully connected graph $\mathcal{G} = (\Sigma, \mathcal{E})$, where Σ represents the set of network nodes and $\mathcal{E}$ represents the set of possible pairwise communication exchanges. We assume signal propagation, though not packet propagation, is instantaneous in the wireless medium. We cannot assume an atomic publisher-subscriber model in non-ideal networks.

The events V_{ij} executed by nodes $S_i \in \Sigma$ as described above are short *pulses* which are broadcast by a sender node S_α and received by all other nodes $S_i \in (\Sigma \setminus S_\alpha)$. The edges $\mathcal{E}$ of the graph $\mathcal{G}$ can be thought of as representing communication channels which are often

unused, but through which a single bit of information will periodically be transmitted when a *pulse* transmission occurs.

Recipients will use the time at which the *pulse* is received, rather than information encoded into the signal itself, as the source data for the coordination algorithm. Other protocols could encode additional information within these transmissions at the expense of higher overhead, but we do not require this. Similarly, if the application happened to produce application-level packets to a schedule conformant with these requirements, then these could be used instead of synchronisation pulse packets.

Precisely how this *pulse* is implemented is irrelevant to the content of this chapter, because any implementation which successfully distributes the single bit messages at the appropriate times would convey the same source information to the algorithm. However, a typical implementation would be the smallest valid packet achievable within a given network stack; the packet *data* is greater than one bit but conveys one bit of *information*.

The minimal time required for this stub packet to traverse the network stack of the sender and the receivers, κ, represents the limit of convergence of the *desynchronisation* primitive. Assuming cells contain n nodes the minimal overhead per epoch is $n\kappa$. For epochs of length e the proportion p of each epoch available for application data transmission is $p = 1 - (n\kappa/e)$. Note that $p \to 1$ as $e \to \infty$; longer epochs imply smaller overhead but greater stabilisation time as per section 7.2.4.

Some mote platforms offer hardware support for accurately determining the timing of incoming packets. For example, the *Mica* platform features a *synchronisation accelerator* which captures the timing of an incoming packet to within one CPU clock cycle (around 250ns) of reception starting [130]. This timing data is made available to higher levels of the network and application stacks, which is useful as it can otherwise be difficult to accurately determine transmission timing at high bit rates.

In an ideal system $\kappa = 0$ such that $M_1 - M_3$ approach their ideal values as system time $t \to \infty$. In a realistic non-ideal system $\kappa > 0$, so we expect M_1 and M_2 to converge within $\pm\kappa$. As M_3 is rounded to the nearest integer we would expect it to converge on the correct integer if κ is sufficiently small.

7.2.7 Tuning

There are three parameters of the *desynchronisation* primitive; the number of nodes, n, the system epoch length, e, and the feedback proportion, f_α. Achieving acceptable network

performance requires the setting of appropriate values of n, e and f_α. Appropriateness is defined in application-dependent and -independent factors.

The hardware in the deployment network may affect the possible range of n. This is particularly important where nodes are mobile or fragile; applications should continue to perform correctly when a single node leaves the cell. Application requirements may specify a minimum and/or maximum number of nodes to give a probabilistic guarantee of coverage of the physical region covered by the sensornet cell. n can never be higher than the number of nodes deployed into the environment, and can never be lower than 1 for any non-degenerate case, but between these bounds the appropriate value of n is application dependent. We conclude that n is significant but not tunable.

The network designer is largely free to set f_α to any defined value to obtain a reasonable tradeoff between responsiveness and stability. We examine the effect of different f_α values in section 7.2.10.2. Usually f_α is set to a high value to achieve good responsiveness, shortening the time to attain the *equilibrium state*. However, non-ideal network conditions can lead to inaccurate, noisy or missing inter-node synchronisation data. Unfortunately, the *desynchronisation* algorithm will respond as quickly to noise as to accurate data, harming solution stability. Network designers can reduce f_α, reducing feedback and increasing systemic *damping*, to minimise this effect at the cost of reduced responsiveness to real system changes. A better solution is given by the improved protocol variants defined in section 7.2.8.

The behaviour of the primitive is independent of e; virtually any value might be selected provided that $e \geq n\kappa$ to allow all n synchronisation messages to be transmitted within each epoch. Within each epoch, the proportion of time consumed by synchronisation is given by $n\kappa/e$. Larger values of e assign a greater proportion $p = 1 - (n\kappa/e)$ of each epoch for application usage rather than synchronisation; $p \to 1$ as $e \to \infty$. As the number of epochs required for the system to reach the required level of convergence is independent of e, if e is large then so is the wall time implied by these epochs. In highly mobile networks it is therefore useful to keep e relatively small, but sufficiently large for application-specified tasks to complete. However, synchronisation messages are typically very small; even relatively small e values are orders of magnitude greater than κ, such that p is insignificant and convergence is fast.

7.2.8 Improved variants of the primitive

In section 7.2.7 we observe that tuning the f_α parameter to increase responsiveness to timing signals has the unwanted side effect of increasing responsiveness to timing errors. Setting low values of f_α damps the response of the *desynchronisation* primitive, improving resilience to transient errors and network conditions at the expense of responsiveness to real network changes. It may be difficult to achieve an acceptable compromise through this single point of influence. In this section we propose an alternative approach in which, rather than selecting low f_α values, we improve the quality of data employed in the feedback calculation.

Recall from section 7.2.4 that each node S_i can disregard all observed synchronisation events other than the phase neighbours of its synchronisation event V_i, and that the sources of these phase neighbour events do not change between system epochs. Normally node S_i will use exactly one instance of the predecessor event $V_{i\beta}$ and the successor event $V_{i\gamma}$ in calculating $\Delta\phi_i$. These single instances are most recent observations, which will occur at $\phi_{i\beta} = -\phi_{max}(e/2n) \bmod \phi_{max}$ and $\phi_{i\gamma} = +\phi_{max}(e/2n) \bmod \phi_{max}$ respectively in the *equilibrium state* from the local viewpoint of node S_i.

Rather than use the most recently observed values of $\phi_{i\beta}$ and $\phi_{i\gamma}$, we propose that each node maintains a *moving average* over the most recent m complete epochs, stored in two queue buffers of size m at each node. Each queue is initially populated with *nil* values which do not contribute to the moving average. During each epoch the new value is pushed onto the head of the appropriate queue, and the oldest value is popped off the end of the queue. If no phase neighbour events are observed in a given epoch, a *nil* value is pushed on the queue instead of a measurement. This is required for well-defined behaviour in the degenerate case where node movement temporarily implies $n = 1$.

For a queue containing ν non-*nil* values, the *fill ratio* $\pi = \nu/m$ increases in $[0, 1]$ as $\nu \to m$. The minimum *fill ratio* π_{min} required to calculate meaningful moving averages is specified by the application designer; larger values imply a greater delay until noise rejection behaviours are active, but have more data with which to work and hence are less susceptible to the influence of outliers.

When the node S_i is required to amend its local phase, as per section 7.2.4, the relative phases $\phi_{i\beta}$ and $\phi_{i\gamma}$ of phase neighbour events are calculated as the arithmetic mean of the associated buffer of recent historical values if $\pi \geq \pi_{min}$; otherwise, we revert to the original strategy of using the most recent observations directly. The underlying primitive remains

fundamentally unaltered in this improved algorithm and hence retains its convergence properties, but operates on higher-quality source data. The network designer must still set an appropriate value of f_α.

It is still possible to set $f_\alpha \in [0, 1]$ but there is now little reason to set $f_\alpha < 1$ as the moving average process implicitly damps the effect of noisy data. Under the original primitive, if $f_\alpha < 1$ historical values are weakly and implicitly influential but all specific information is lost at each epoch. Under the improved primitive, the historical values are stored explicitly and their influence on calculation of new values is strong and explicit; specific information is discarded slowly in a controlled and predictable manner.

To improve responsiveness we use variants of the plain moving average that give greater weighting to more recent values, but can still operate effectively when the value for the current system epoch is undefined as a result of a lost pulse. Assume we label the non-null historical data values in each buffer as $x_1, \ldots, x_m$ where x_m is the most recent. We employ an *exponentially weighted moving average* in which the weighting w of historical data point x_y is given as $w_y = y^z$ where $z \in \mathbb{R}$ is the *scaling exponent*. If $z = 1$ then we have the plain moving average. If $z > 1$ then newer data are more significant, whereas if $z < 1$ then older data are more significant. Usually $z > 1$ will be selected to give higher priority to newer data.

Algorithm 2 defines the primitive behaviour executing at each node $S_i \in \Sigma$ for the improved primitive variants B and C. Function Π returns the fill ratio of a given buffer. Function *avg* returns average of the values stored in a given buffer, where the type of average is appropriate to the selected primitive variant. Other variables and functions not defined within the algorithm itself take the standard meanings used elsewhere in this document.

7.2.9 Cost analysis

The plain version of the *desynchronisation* primitive defined in section 7.2.4 requires only two items of data to be stored. As the local phase ϕ_i increases from 0 to ψ_{max} for some given node S_i any number of pulse events might be observed, but only the first and last are retained. The first corresponds to the successor pulse event $V_{i\gamma}$, and the last corresponds to the predecessor pulse event $V_{i\beta}$, that surround the local pulse event V_i. We require storage space for exactly two such timing data, as each value will be overwritten with new

Algorithm 2 : Primitive variants $B - C$ at node S_i

Require: Most recent observed sync phase, $\phi_\alpha = nil$
Require: Predecessor sync phase queue buffer, $Q_{i\beta} = \varnothing$
Require: Successor sync phase queue buffer, $Q_{i\gamma} = \varnothing$
Require: Peer sync event counter, $c = 0$

while monitoring local phase ϕ_i increasing over time **do**
  if sync event $\neq V_i$ observed **then**
    $c \Leftarrow c + 1$
    $\phi_\alpha \Leftarrow \phi_i$
    if $c = 1$ **then**
      $Q_{i\gamma} \Leftarrow Q_{i\gamma} \cup \{\phi_\alpha\}$
      if $\Pi(Q_{i\beta}) > \pi_{min} \wedge \Pi(Q_{i\gamma}) > \pi_{min}$ **then**
        $\phi_{i\beta} \Leftarrow -(\phi_{max} - \text{avg}(Q_{i\beta}))$
        $\phi_{i\gamma} \Leftarrow avg(Q_{i\gamma})$
        $\theta_i \Leftarrow \phi_{i\beta} + \phi_{i\gamma}$
        $\Delta\phi_i \Leftarrow -f_\alpha\theta_i$
        $\phi_i \Leftarrow (\phi_i + \Delta\phi_i) \bmod \phi_{max}$
        $\phi_\alpha \Leftarrow (\phi_\alpha + \Delta\phi_i) \bmod \phi_{max}$
        for all $q_{i\beta} \in Q_{i\beta}$ **do**
          $q_{i\beta} \Leftarrow (q_{i\beta} + \Delta\phi_i) \bmod \phi_{max}$
        end for
        for all $q_{i\gamma} \in Q_{i\gamma}$ **do**
          $q_{i\gamma} \Leftarrow (q_{i\gamma} + \Delta\phi_i) \bmod \phi_{max}$
        end for
      end if
    end if
  end if
  if $\phi_i \geq \phi_{max}$ **then**
    $Q_{i\beta} \Leftarrow Q_{i\beta} \cup \{\phi_\alpha\}$
    if $c = 0$ **then**
      $\phi_\alpha = nil$
      $Q_{i\gamma} \Leftarrow Q_{i\gamma} \cup \{nil\}$
    end if
    $\phi_i \Leftarrow 0$
    $c \Leftarrow 0$
    fire own sync event V_i
  end if
end while

data during each epoch. Therefore, the storage overhead is O(1) in node population, n. The algorithmic complexity is also O(1) in n because the algorithm requires a small fixed number of steps to be executed during each epoch; there are no loops or other recursive constructs. This low overhead is highly desirable in sensornet systems which have few resources to allocate.

Now consider the moving average variants defined in section 7.2.8. Storage and computation overheads remain O(1) in node count as the algorithm continues to consider only the two phase neighbour nodes, irrespective of any number of other participating nodes which might be present. However, we must now consider the number of event observation timing values, m, which contribute to the moving average on each execution of the algo-

rithm. Note that this applies only to the calculation of the effective phase of events $V_{i\beta}$ and $V_{i\gamma}$; the phase adjustment algorithm is unaffected.

There exist algorithms to calculate simple moving averages that are O(1) in storage and computation overhead [39], and if these are employed it is obvious that the moving average offers significantly improved performance with minimal increased overhead. However, a general moving average algorithm may be worse than O(1) but no worse than O(m) in storage and computation and overhead, the latter being observed if the algorithm must consider all m contributing data on each iteration.

We observe that each execution of the algorithm at each node is guaranteed to terminate in O(1) time. However, the algorithm is executed once at each node during each epoch, so in this sense the algorithm never terminates. This latter condition is essential if the algorithm is to remain responsive to changing network conditions; it is obvious that no algorithm could respond after terminating.

For systems expected to be deployed into highly predictable and rarely changing environments, non-terminating algorithms may not be the most efficient choice. However, sensornets are typically deployed in highly unpredictable and changeable environments, and mobile ad-hoc networks are characterised by continual change; the algorithms described in this chapter are an appropriate choice. For moderately changing environments, these primitives can be executed until equilibrium is reached, then cyclically suspended for significant periods then executing for short periods. During suspended periods the extant event schedule can be reused without incurring overhead, with schedule repair and recalibration occurring during execution periods.

7.2.10 Experimental results

We model the Crossbow MICA2 mote in our experiments. We set $\kappa = 1 \times 10^{-3}s$ as the time required for a synchronisation pulse transmission-reception pair to complete, and hence take this as the threshold deviation from the ideal value of metrics M_1 and M_2 within which we consider a system *converged*. As metric M_3 is inherently rounded we require the measured value to exactly match the ideal value. Each metric is measured at all nodes $S_i \in \Sigma$. We count the elapsed time in system epochs from network initialisation to the point at which the mean, minimum and maximum values measured across the participating nodes all fall within the defined threshold.

Unless stated otherwise we use a fixed cell population $n = 10$ nodes, because this

is an energy-efficient cluster size for typical 1000-node sensornets [312]. We label the plain desynchronisation algorithm as A, the *basic moving average* variant as B, and the *exponentially weighted moving average* variant as C. We select epoch length $e = 10s$ so that epochs are large compared to k and long enough for realistic tasks to complete between synchronisation events in time e/n. We select feedback $f_\alpha = 0.9$ yielding similar fast convergence under all variants $A - C$ (see section 7.2.10.2).

For variants B and C we set buffer size $m = 10$ to ensure that sufficient captured synchronisation data contributes to moving averages to reject the effect of outliers and timing error, but does not contain unacceptably stale historical data which may no longer be representative of current network conditions. We set fill ratio $\pi_{min} = 0.5$ assuming that synchronisation timing data extracted from fewer than half of the system epochs may be unrepresentative, although the protocol would continue to function under this condition. For variant C we specify scaling exponent $z = 2$ such that newer data exert more influence than older data.

We do not claim that these parametric values are optimal. Selecting the most appropriate values for a given specific network is an optimisation problem which is beyond the scope of this chapter; the tuning methods described in chapter 4 could be applied to this problem. However, these values are typical and illustrative, and we show that useful behaviour is observed over broad ranges of the defined parameters.

To model other hardware platforms substitute a different κ, and to model other networks different values of n, e and f_α can be used; the results are qualitatively equivalent but quantitatively different. Note that metrics M_1 and M_2 approach their κ convergence limits asymptotically; it is possible to achieve a looser but acceptable degree of convergence in significantly shorter time. Network designers must tradeoff solution quality against algorithm efficiency when specifying network requirements.

Section 7.2.10.1 models coordinated and uncoordinated network deployment scenarios. Section 7.2.10.2 models networks of differing cell size and responsiveness requirements. Section 7.2.10.3 models situations in which mobile nodes enter or leave the physical region covered by a network cell, suspend or wake in response to duty cycle management protocols, or leave the network owing to hardware failure. Section 7.2.10.4 models networks where malfunctioning hardware, environmental obstacles or deliberate sabotage disrupts inter-node communications. Section 7.2.10.5 models networks where malfunctioning hardware, poor application design or extreme ambient temperature induces local timing errors.

7.2.10.1 Cell initial configuration

We define *initial configuration* as the set of initial node phases relative to the start of the first system epoch. In *random* initial configurations these starting phases are randomly distributed in the interval $[0, \psi_{max})$. In *ideal case* initial configurations these starting phases are evenly distributed in time, identical to the *desynchronised equilibrium state*. In *worst case* initial configurations all starting phases are equal, identical to the *synchronised equilibrium state*.

We begin by illustrating convergence of metrics from a *random* initial configuration. Figure 7.3 shows the mean values of metrics $M_1 - M_3$ across all nodes, with all measured values normalised to the range $[0, 1]$. Metrics were sampled at the end of each of the first 100 system epochs under the original algorithm variant A. Similar plots are obtained for variants B and C.

All metrics $M_1 - M_3$ can be approximated by sequences of the form $f(j) = 1/j + c$ in epoch j where c is some constant. We observe that M_1 very quickly approaches its limiting value. As epoch j increases the value M_{1j} alternates between higher and lower than the limit $M_{1\infty}$ with the difference $|M_{1j} - M_{1\infty}|$ quickly becoming small. M_3 also approaches its limiting value $M_{3\infty}$ quickly, though not as quickly as M_1, with relatively large perturbations from the idealised hyperbolic form explained by the quantisation of individual measurements to integral values (see section 7.2.5). M_2 converges more slowly than M_1 or M_3 but declines smoothly and monotonically toward the limit $M_{2\infty}$.

Table 7.1 presents the time required for metrics $M_1 - M_3$ to converge. Consider the behaviour when the system starts in the *best-case* configuration, equivalent to an *equilibrium state* of the algorithm. We see that the system maintains this ideal configuration for all metrics $M_1 - M_3$. This simply, but importantly, indicates that the algorithm will not take the system from an *equilibrium state* to a *non-equilibrium state*. We need not consider the *best-case* configuration further.

Now consider the M_1 metric. We see that M_1 reaches its converged value very quickly for all algorithm variants and all initial configurations. We conclude that all variants are highly capable in this regard under ideal network conditions and need not consider this metric further.

For all variants $A - C$, we see that all metrics $M_1 - M_3$ will converge in finite time starting from a *randomised* or *worst-case* initial configuration. In all experiments, reaching the convergence limit required more epochs from a *worst-case* initial configuration. This

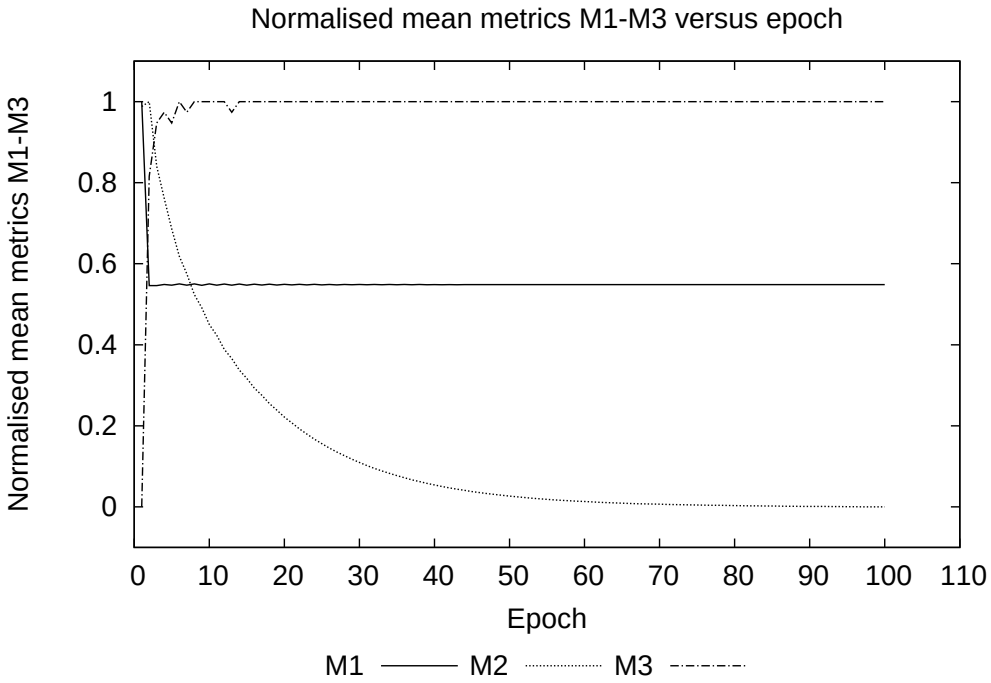


Figure 7.3: Normalised metrics for variant A

is unsurprising as the *worst-case* configuration is further from the *best-case* configuration than almost every *randomised* configuration, except for *randomised* configurations that are also *worst-case*.

The number of epochs required to reach the convergence limit $M_{3\infty}$ is nearly the same for each algorithm variant $A-C$. This is a consequence of the calculation of M_3 rounding intermediate values to the nearest integral value, an effect which will dominate small variation in pre-rounded intermediate values as these converge.

Now consider the M_2 metric, which in all cases is the slowest to reach the convergence limit and therefore defines the point at which cells reach an *equilibrium state*. Starting with a *randomised* initial state we observe the epochs required for convergence is of the same order of magnitude for each algorithm variant, but convergence is reached somewhat faster under variant A than B or C; a smaller difference exists between values for variants B and C. This is explained by hysteresis effects; variant B calculates new values using historical data and variant A does not, so the output of variant B lags behind that of A. Variant C is somewhere between A and B both in the influence of historical data and the corresponding measured responsiveness.

We conclude that all algorithm variants $A-C$ are effective under ideal network conditions.

Initial state	**Algorithm variant**	**Epochs to convergence**			
		M_1	M_2	M_3	MAX
Random	A	3	25	11	*25*
	B	3	38	21	*38*
	C	3	37	21	*37*
Best	A	1	1	1	*1*
	B	1	1	1	*1*
	C	1	1	1	*1*
Worst	A	3	35	21	*35*
	B	3	56	24	*56*
	C	3	54	24	*54*

Table 7.1: Convergence times for metrics

7.2.10.2 Cell composition

In this section we measure the epochs required for all metrics $M_1 - M_3$ to converge to an *equilibrium state*. Figure 7.4 illustrates the relationship between f_α and the number of system epochs, y, which must elapse before the system reaches an *equilibrium state* under algorithm variants A and B; the trace for variant C is very similar to that of B and is omitted for clarity. Each value of f_α was evaluated with an identical *worst-case* initial configuration

Traces A and B are similar, though not identical, for $f_\alpha \in (0, f_{critical})$ where $f_{critical} \approx 0.91$. Up to this point, both A and B describe approximately hyperbolic traces such that the relationship between f_α and epoch of *equilibrium state* can be approximated by the form $f(j) = 1/j + c$ in epoch j where c is some constant. A difference in behaviour is noted for $f_\alpha > f_{critical}$; trace B continues its original hyperbolic path, whereas trace A grows quickly with $f_\alpha \in [f_{critical}, 1]$. Two distinct effects must be considered to understand this relationship.

In each epoch, each node S_i amends its local phase by $\Delta\phi_i = -f_\alpha\theta_i$ where θ_i is the perceived phase difference between the local synchronisation event at $\phi_i = \psi_{max}$ and the midpoint of the phase neighbour events. The greater the value of f_α, the greater the proportion of perceived difference that is fed back into the system, pushing the system toward the *equilibrium state* more quickly. This explains the shape of trace B for $f_\alpha \in [0, 1]$, and the shape of trace A for $f_\alpha \in [0, f_{critical}]$.

Now consider trace A for $f_\alpha \in [f_{critical}, 1]$. θ_i is continuously variable but κ, the time for a pairwise exchange of synchronisation event, is constant. Converting κ from time units to phase units, the magnitude $|\Delta\phi_i|$ becomes small compared to the magnitude $|\kappa\phi_{max}|$. As the magnitude $|\kappa\phi_{max}|$ defines the uncertainty of the phase neighbour event midpoint measurement, it follows that the magnitude of the measurement error becomes significant compared to the magnitude $|\Delta\phi_i|$. This causes convergence to slow as the limit is approached. Each iteration of the procedure must attempt to correct for previous measurement errors within the new phase difference measurement.

If f_α is small, the proportion of this measurement error fed back into the system is also small, so its effect is insignificant. As f_α grows so does the proportion of measurement error feedback. Under variant B the measurement error is found in all stored samples. Although the error values are not explicitly available, as they derive from consecutive system epochs they are likely to be of similar magnitude, and they are as likely to be positive as to be negative. Taking the average of the samples will approximately cancel the measurement errors, so the effect of these errors does not become dominant. Under variant A there is no such cancellation effect, hence the effect of these errors becomes dominant. Defining convergence limits of significantly larger magnitude than κ would hide this phenomenon without actually addressing the underlying issue.

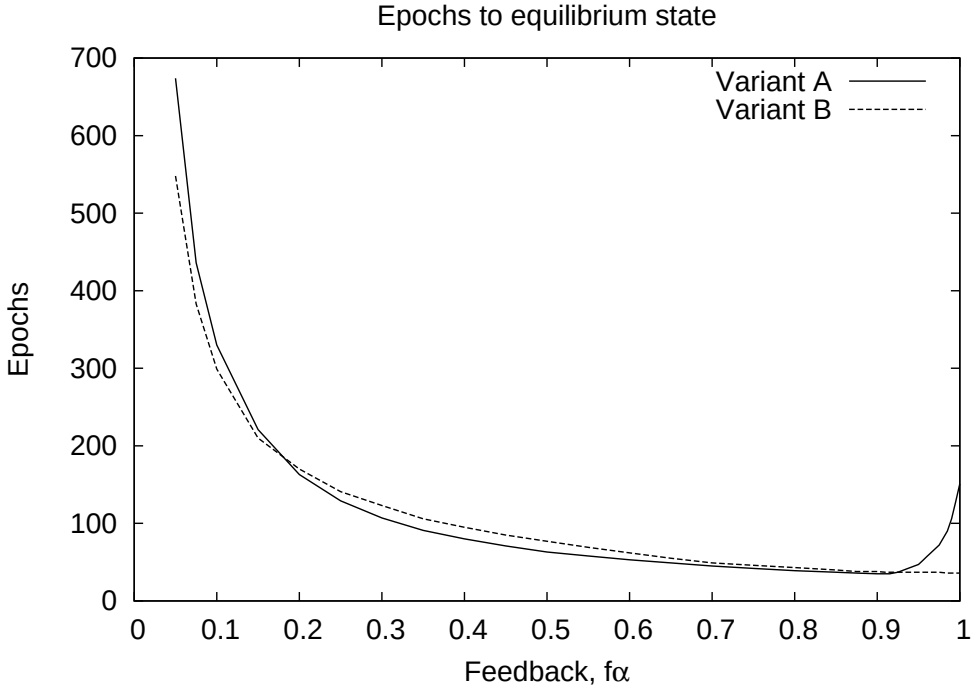


Figure 7.4: f_α versus epochs to equilibrium

Figure 7.5 illustrates the relationship between n and the number of system epochs, y, which must elapse before the system reaches an *equilibrium state* under algorithm variant A. Similar plots are observed for variants B and C. As the cell n increases the general trend is that y increases too. It is notable that this increase is not monotonic, and does not conform readily to any well-known relationship. Despite the guarantee that the system will converge [201] it is difficult to predict the time required. This is a consequence of algorithm variants $A - C$ defining feedback-driven systems, in which the relationship between input and output is deterministic yet difficult to predict [77].

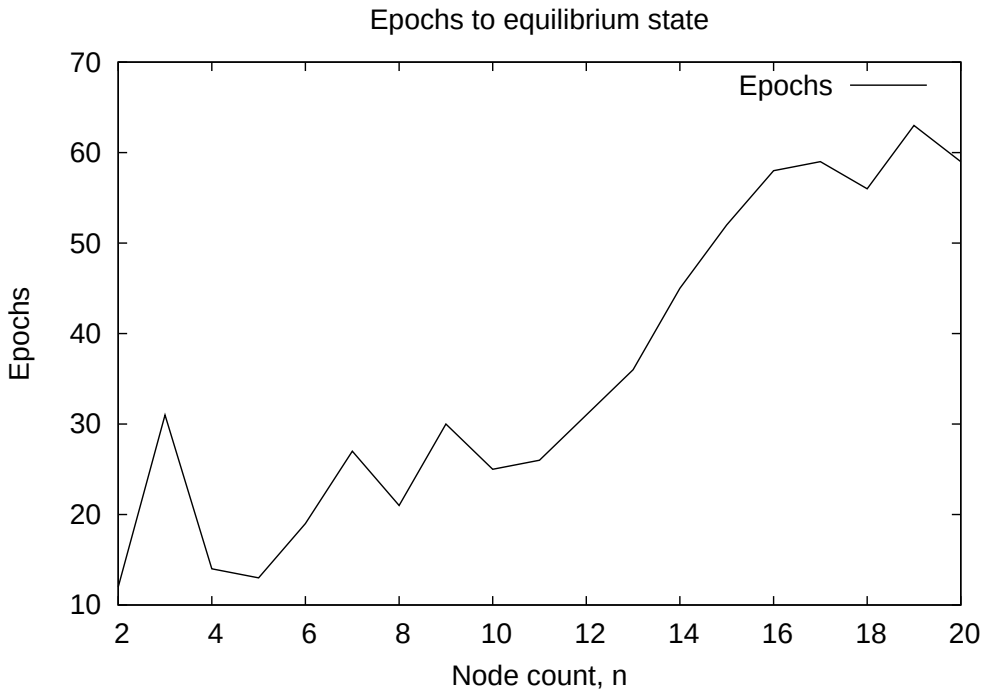


Figure 7.5: n versus epochs to equilibrium

The primitives considered in this section operate exclusively within individual network cells. Larger networks may be divided into multiple cells. Separate instances of the primitive operating in adjacent cells may interact if the communication range of some nodes extends beyond their own cell, if nodes cannot determine the cell from which a given transmission originates. If not managed, these interactions could cause disruption, similar to the *phantom pulse* effect examined in section 7.2.10.4.

These interactions can be exploited for beneficial effect. Extensions based on *entrainment* have been implemented which progressively synchronise equivalent transmissions in adjacent cells. This enables intercellular cooperation, mitigates the risk of clashing behaviour, and enables efficient handover of mobile nodes between cells. A detailed de-

scription is given in chapter 9.

7.2.10.3 Cell population change

In this section we consider algorithm performance for cells starting in a stable *best-case* where a node is either added or removed from the cell population. We then measure the time required to reach a new *equilibrium state* where all metrics $M_1 - M_3$ are converged. We plot metric M_2 against epoch as this is the slowest to converge. Integrating the area under each plot gives a quantitative estimate of deviation from desired system behaviour over the system epochs indicated on the x-axis, accounting for both short- and long-term effects of varying magnitude.

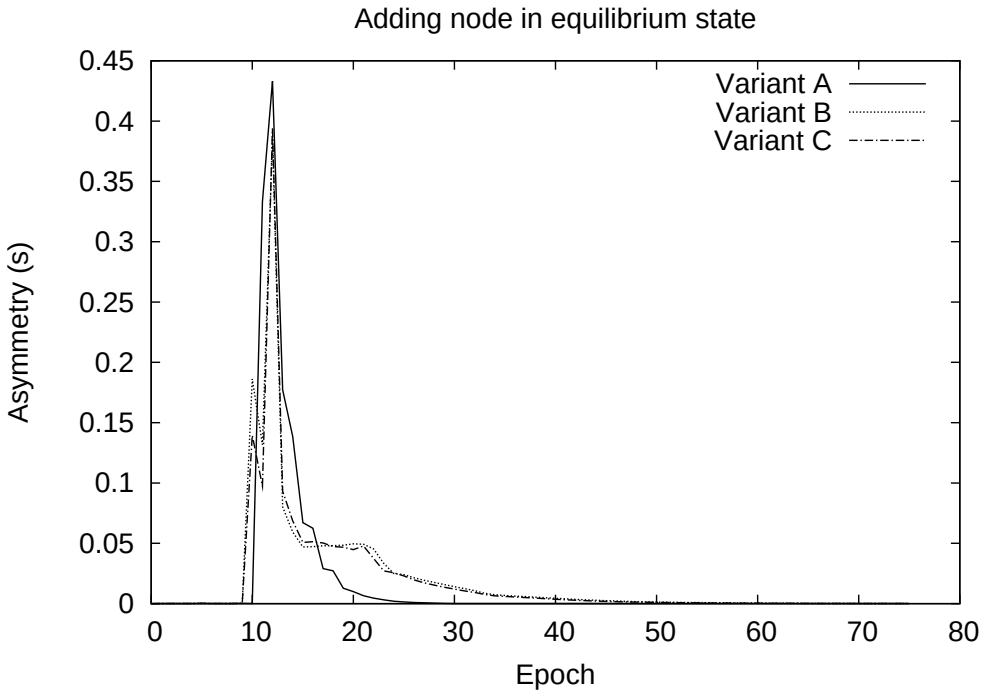


Figure 7.6: Adding node to stable system

Figure 7.6 shows a node being added to a stable 5-node system. Variant A requires 21 epochs to re-establish the *equilibrium state*, variant B requires 58 epochs, and variant C requires 57 epochs. Figure 7.7 shows a node being removed from a stable 5-node system. Variant A requires 16 epochs to re-establish the *equilibrium state*, variant B requires 47 epochs, and variant C requires 46 epochs. The node removal experiments re-establish the *equilibrium state* more quickly because the new stable system is smaller than the new stable system in the node addition experiments.

In all cases the *equilibrium state* is re-established in finite time. Note that decreasing

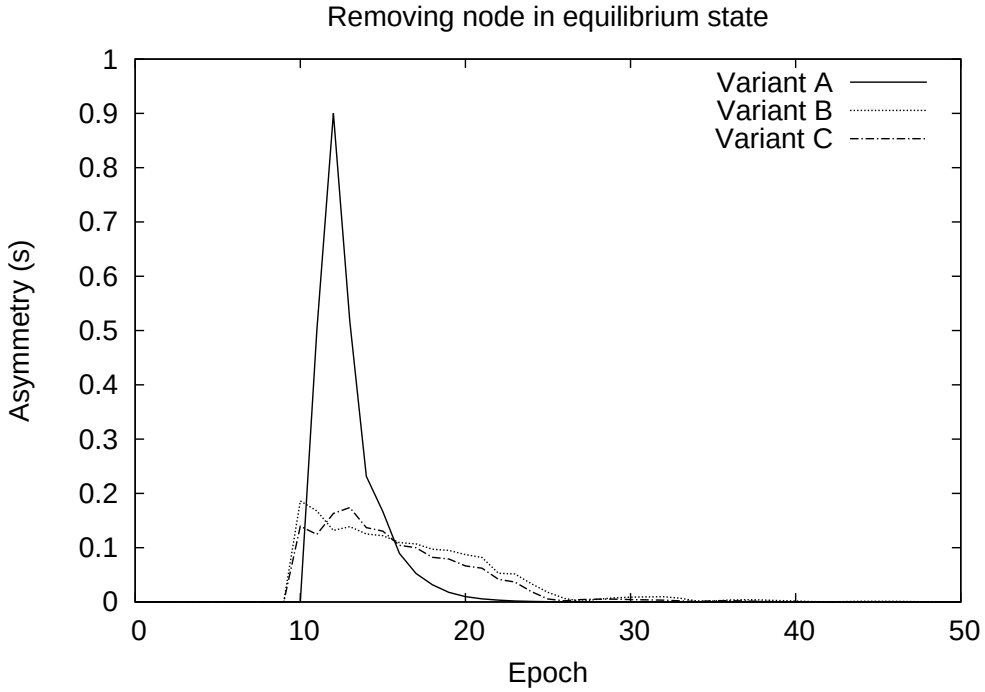


Figure 7.7: Removing node from stable system

e reduces this time linearly. As the algorithm is capable of restabilising the cell schedule when a single node is added or removed, it is capable of dealing with multiple additions or removals as these can be decomposed into an equivalent temporally ordered sequence of single additions and removals. This is particularly helpful in networks of highly mobile nodes, in which cell membership is expected to change frequently.

7.2.10.4 Radio error resilience

In this section we consider algorithm performance for cells starting in a stable *best-case* where network conditions are non-ideal. It is possible that a synchronisation pulse transmission V_{ij} may fail to be heard at one or more of the intended recipients; we call each instance a *lost pulse*. Reception will either succeed or fail independently and atomically at each potential recipient. We measure performance where reception of an arbitrary pulse at an arbitrary node fails stochastically with probability $p \in [0, 1]$. Integrating the area under each plot gives a quantitative estimate of deviation from desired system behaviour over the system epochs indicated on the x-axis, accounting for both short- and long-term effects of varying magnitude.

In figure 7.8 we set $p = 0.05$. For each variant $A - C$ exactly the same synchronisation pulse transmitter-receiver pairs were lost. We see that variants $B - C$ significantly

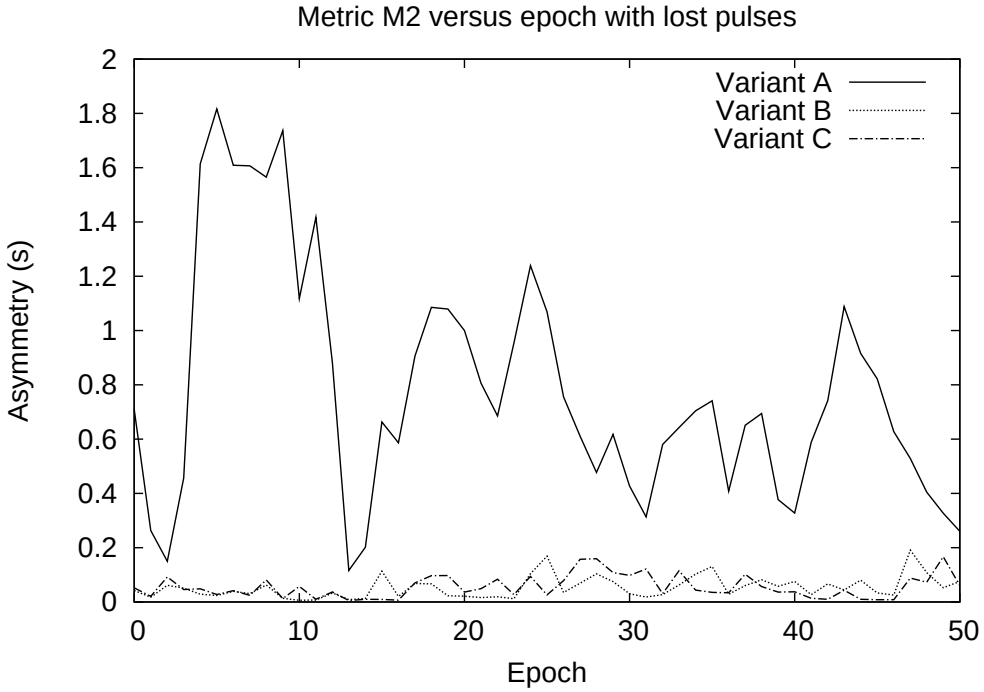


Figure 7.8: Lost pulses

outperform the original variant A significantly, with a much smaller deviation in metric M_2 from the ideal value of $M_2 = 0$. Although neither variant B nor C cope perfectly with pulse loss, and there is little to pick between them, they offer substantially improved performance and stability.

Synchronisation pulses have minimal length and content; a *phantom pulse* is feasible where radio noise or corrupted packets are interpreted as a synchronisation pulse. We measure performance where nodes observe *phantom pulses* distributed randomly in time with rate r given in s^{-1}. It is possible to reduce r by increasing the information contained in the synchronisation pulse, perhaps by encoding a hash derived from a shared secret key and the transmission time; this would also provide resilience to attack by rogue synchronisation pulse transmission [272]. However, this would increase the overhead of the *desynchronisation* primitive and is beyond the scope of this section.

In figure 7.9 we set $r = 0.1\ s^{-1}$. For each variant $A - C$ exactly the same phantom pulses were heard by nodes. Again, we observe that variants B and C offer significantly better stability and performance than variant A.

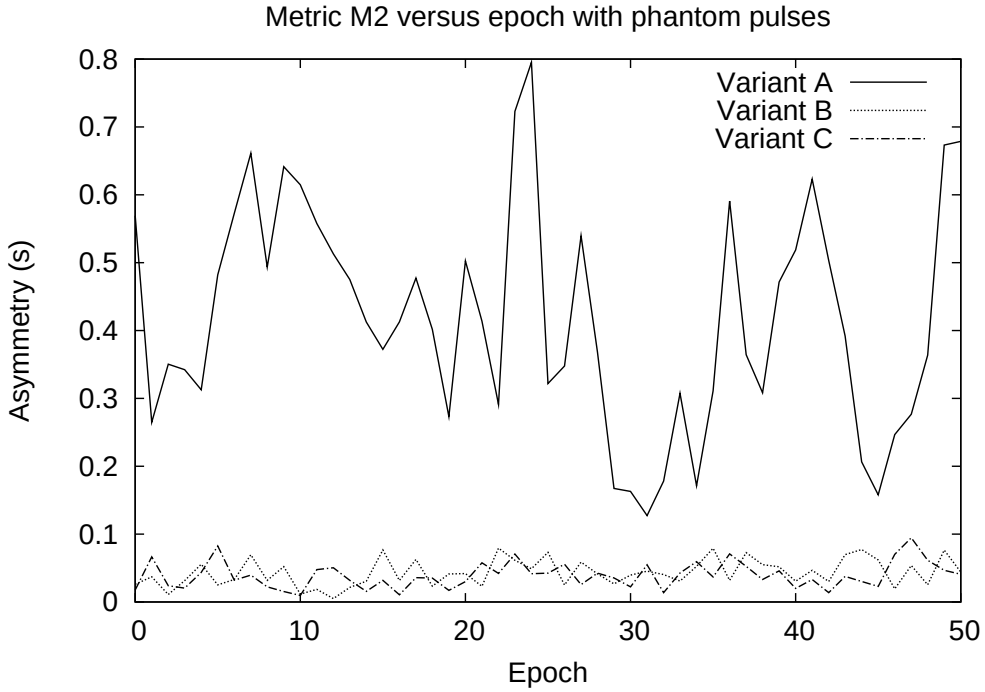


Figure 7.9: Phantom pulses

7.2.10.5 Clock error resilience

In this section we consider algorithm performance for cells starting in a stable *best-case* where timings are not accurate. *Jitter* in synchronisation pulse transmission times may result from non-ideal task scheduling algorithms or preemption by higher priority tasks at the sender node. Although many definitions are possible [185] we define the jitter ι of a given synchronisation pulse as the difference between the intended and actual transmission times, where ι is distributed normally as $\iota \sim N(\mu_\iota, \sigma_\iota^2)$. Transmission jitter affects both phase neighbours of the transmitter node, whereas an individual radio error affects only a single receiver. Integrating the area under each plot gives a quantitative estimate of deviation from desired system behaviour over the system epochs indicated on the x-axis, accounting for both short- and long-term effects of varying magnitude.

In figure 7.10 we set $\mu_\iota = 0s$, as early transmission is as likely as late transmission, and $\sigma_\eta = 0.1s$. For each variant $A - C$ pulse transmission times are subject to exactly the same jitter. We observe that variants $B - C$ show significantly better stability and performance than variant A. Under variants $B - C$ the uncorrected error component is of the same order of magnitude as the standard deviation of jitter.

Clock drift is observed if local node clocks are imperfect. As one second passes in the physical world the clock may measure more or less than one second passing, governed by a

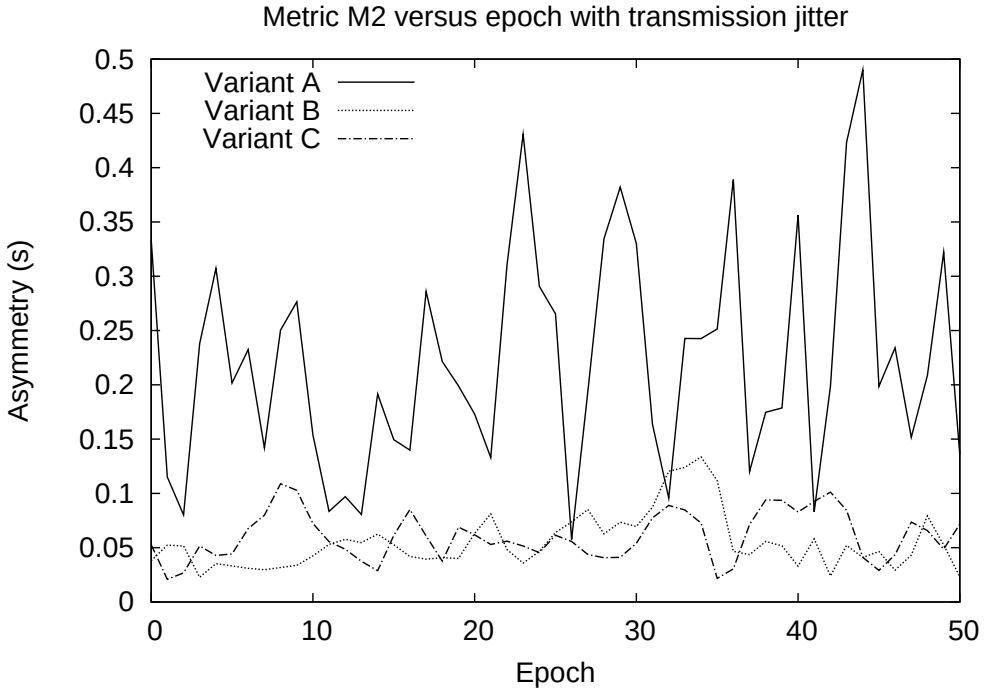


Figure 7.10: Jitter

scaling factor $\eta > 0$. Perfect clocks have $\eta = 1$; manufacturing imperfections and variation between calibration and operational temperature tend to give $\eta \neq 1$ [200]. We assume each node clock has constant η [262], distributed normally as $\eta \sim N(\mu_\eta, \sigma_\eta^2)$. We set $\mu_\eta = 1$ to model clocks equally likely to run fast as to run slow, as compared to the notional global clock. We set $\sigma_\eta = 1 \times 10^{-3}$, modelling drift rates with standard deviation several orders of magnitude greater than the 1×10^{-6} *seconds per second* drift typical of commodity quartz crystal timers [64]. Figure 7.11 shows variants $A - C$ perform acceptably in rejecting drift effects, with uncorrected error of the same order of magnitude as the drift. For variants $B - C$ we see some initial stabilisation as drift-laden measurements fill the buffers.

7.3 Summary

The *Lightweight Integrated Protocol Suite* (LIPS) coordinates time-sensitive activity, and regulates network size and density, in self-managing sensornets. Although each protocol can be implemented in isolation, each contributes part of a larger, integrated solution to the problem of automated low-level infrastructure management for self-managing sensornets. It follows that implementing the full suite offers the greatest potential for improvement over the base case of an unmanaged, or manually managed, sensornet infrastructure.

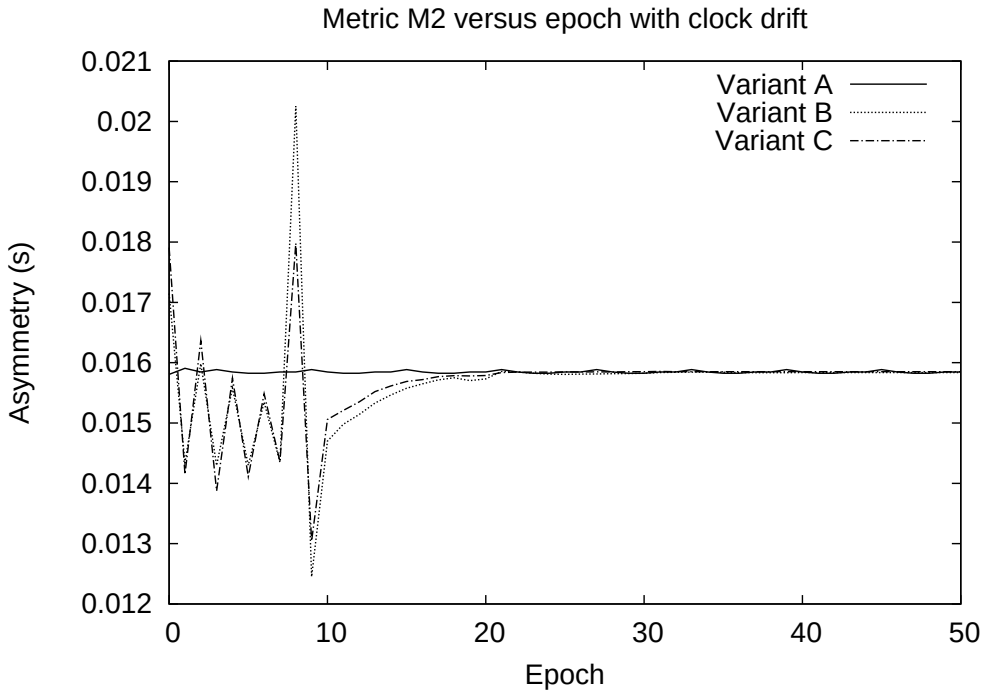


Figure 7.11: Clock drift

An intracellular timing coordination primitive based on the *desynchronisation* principle is lightweight and effective when operating within sensornet cells. However, the original version is prone to instability arising under common non-ideal timing and network conditions. This is a consequence of the feedback control loop; it is key to attaining a desired steady state condition, but does not reject input signal noise.

Improved versions of the primitive were defined in section 7.2.8 and form the basis of the *Lightweight Improved Synchronisation Primitive* (LISP). Significant and measurable improvements in stability were obtained without sacrificing performance, as shown in section 7.2.10. Algorithmic and storage overheads are of the same order, O(1), in cell population n as the original. The synchronisation functionality of this protocol is directly usable in itself, but also constitutes the foundation for other protocols in the suite.

Chapter 8

Distributed state management

Section 7.1 introduces the idea that we can control the size and structure of a logical virtual sensornet [144] by managing the duty cycle of individual nodes in the physical network. We can construct networks within which there is an adequate number and density of actively participating nodes in all physical regions to support distributed application requirements, without unnecessarily wasteful redundancy or duplication.

This chapter considers mechanisms with which a distributed duty schedule for a sensornet cell can be constructed and maintained over *short-*, *medium-*, and *long-term* timescales, as discussed in section 7.1.1. At any given time, the set of nodes assigned to be on duty implicitly defines the members of the logical virtual sensornet.

8.1 Short-term duty schedule coordination

Numerous MAC protocols suited to sensornets have been proposed [76,155,256] at the *Data Link Layer* of the OSI protocol stack model [343] which manage contention for a shared medium. These define the times within which a network entity can legitimately transmit into a shared medium, the times at which the entity is expected to listen for transmissions, and the circumstances under which the entity can stop listening to a message transmission which is not required or is irrelevant [284]. These protocols generally define and impose order over only a short period, and over a short distance in the case of a wireless shared medium. The medium- and long-term coordination protocols described in sections 8.2 and 8.3 place no special requirements on the MAC protocol employed for short-term coordination. It is assumed that a MAC protocol will be selected and implemented in any sensornet, but the details will not be considered here.

8.2 Medium-term duty schedule coordination

Sensornets compose many small, low-cost computing nodes into distributed systems deployed into physical environments of interest. Nodes have restricted energy, computation and storage resources and therefore limited utility in isolation; cooperation and coordination is necessary to address realistic problems.

Consider a large sensornet consisting of many nodes, divided into cells containing smaller numbers of nodes in close geographic proximity [44]. Within a cell each node has a similar view of the physical environment, and similar connectivity to nearby base stations or surrounding cells [225]. It follows that all nodes within a cell are approximately equivalent with respect to extracellular entities and environmental context.

Suppose that an external entity broadcasts a message received by all members of a cell. Unless the message is intended for a specific member of that cell, it is unclear which cell member or set of cell members should respond. Data packets to be forwarded to remote destinations need only be rebroadcast once; if all cell members rebroadcast this wastes energy, increases contention for the wireless medium, and risks collisions [217]. If a tasking message requests that a sample value be read from the physical environment then all cell members will produce equivalent readings [103]. Consequently, energy and network capacity may be wasted in delivering multiple redundant messages.

Any of a number of similarly positioned nodes are equally valid candidates to handle specific tasks. Some mechanism is required to avoid wasteful repetition, and mitigate ambiguous or unpredictable multiple responses to stimuli, by enforcing mutual exclusion [335]. If all nodes in a cell periodically sample the same aspect of their physical environment, each node will obtain a similar dataset over time without exchanging sample data, provided that the sampling rate is sufficiently high to track changes in the observed physical phenomenon. If exactly one node of multiple redundant candidates is on duty at any given time there is never ambiguity as to which node must respond to external stimuli.

By deterministically assigning responsibility for response, we implicitly identify the nodes which will not be required to respond. These nodes can switch unused energy-hungry subsystems into low power modes. The consequent energy saving extends the useful lifetime of sensornets composed of nodes with finite energy resources [184]. Sensornets can run indefinitely if duty cycle allocation allows nodes to scavenge energy from the environment at the rate of consumption [145].

8.2.1 CDAP: Cyclic Duty Allocation Protocol

The *Cyclic Duty Allocation Protocol* (CDAP) is an application- and platform-agnostic lightweight protocol to cycle duty between the nodes of a network cell. System *epochs* are divided into portions of equal length and allocated fairly among nodes, such that each node is assigned responsibility for one portion during each epoch. Exactly one node is deterministically assigned this responsibility at any arbitrary time, removing ambiguity as to which node must respond to stimuli. Applying well-understood *synchronisation* phenomena observed in nature [201], inter-node coordination is achieved by cells acting as closed systems of pulse-coupled oscillators. As cells approach stable equilibrium states, nodes can identify periods in which energy-saving states can safely be entered.

The *desynchronisation*-based LISP primitive described in section 7.2 above generates a periodic sequence of synchronisation transmissions spaced evenly in time. We use this as the basis of a duty allocation protocol, although any functionally equivalent source of periodic synchronisation events could be substituted with equivalent results.

8.2.1.1 CDAP protocol states

A simple Finite State Machine runs at each node. The states define the communication responsibilities of a given node at a given time with regard to peer nodes within the cell and external entities beyond the cell. As the local phase ϕ_i of node S_i increases from 0 to ϕ_{max} the protocol state may be changed by detected synchronisation events, or by state timeouts.

ONDUTY - Node is responsible for communications duties of the cell, and is responsible for handling any incoming packets. Node can hear both application messages and synchronisation messages. Radio modules are switched on and ready for bidirectional exchange with neighbouring entities, and transmit their own synchronisation message in the middle of this period.

SCAN - Node is listening for synchronisation messages but has not yet collected sufficient data to predict times of phase neighbour peer node synchronisation transmissions. Node can hear synchronisation messages but does not expect application messages. Node radio module is switched to the lowest-power mode that can detect the synchronisation messages, except when transmitting its own synchronisation message.

SYNC - Node is waiting for synchronisation messages around the predicted times of phase neighbour peer node synchronisation transmissions. Node can hear synchronisation messages but does not expect application messages. Node radio module is switched to the lowest-power mode that can detect the synchronisation messages.

OFFDUTY - Node has no communication responsibilities and is free to switch radio modules off or into other low-power modes. Nodes can hear neither application nor synchronisation messages.

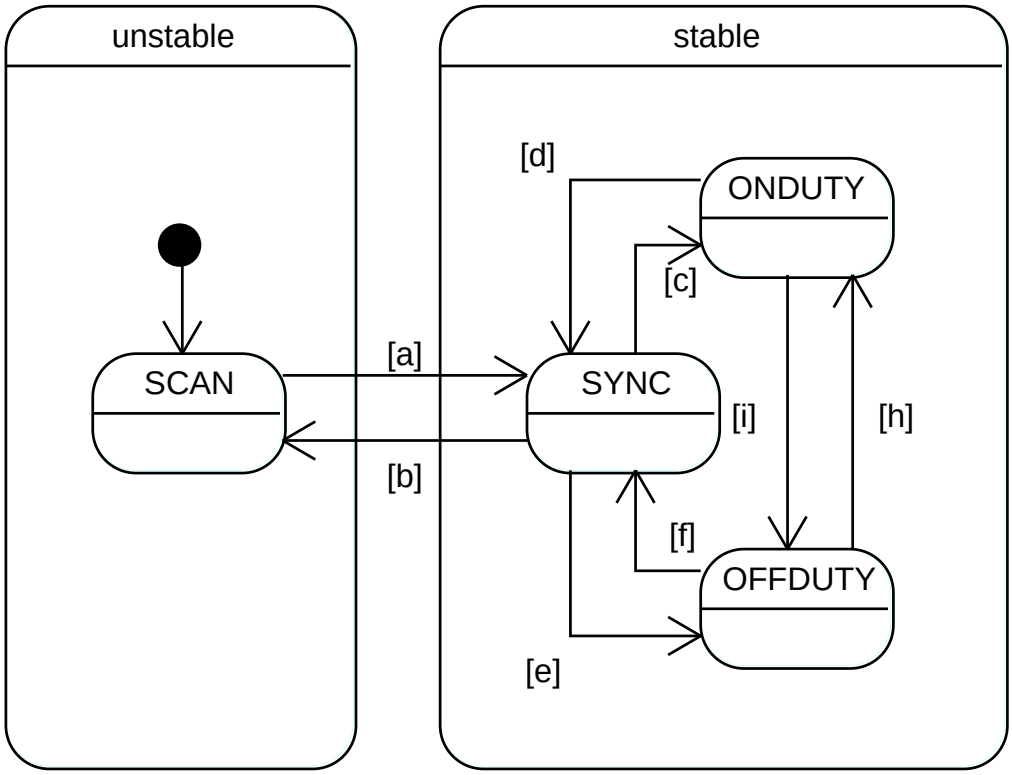


Figure 8.1: Finite State Machine for CDAP states

Figure 8.1 illustrates CDAP states and state transitions. The associated guard conditions are discussed in section 8.2.4 when the necessary terminology has been introduced.

The CDAP protocol builds a distributed schedule which defines the periodic duty cycles for each node in each system epoch. We will consider the mechanism by which this schedule is constructed in section 8.2.1.2 but these details are irrelevant at this point. The duty period in state *ONDUTY* is obtained for each node, with that node's own synchronisation transmission occurring at the midpoint of the *ONDUTY* period and the phase neighbours' synchronisation transmissions occurring at times outside the *ONDUTY*

period. In the most basic form of CDAP it follows that, by default, nodes are in the *SYNC* state listening for synchronisation messages, transitioning temporarily to *ONDUTY* when the duty period begins and transitions back to *SYNC* when the duty period ends.

This provides the desired mutual exclusivity property in which there is exactly one node in the *ONDUTY* state at all times except for a small handover period between nodes. The *ONDUTY* state takes precedence over all others if timing errors lead to conflict at any node. However, this basic schedule is not energy efficient. Listening to the wireless medium for synchronisation messages often consumes energy as quickly as listening for application messages, and always consumes more energy than a low-power standby state. We would prefer nodes to spend time in the *OFFDUTY* state to conserve energy when possible.

We address this issue by observing that phase neighbours' synchronisation transmissions are generally short in comparison to the system epoch, and occur at reasonably predictable times as the system converges on the equilibrium state if signal noise and timing error is moderate. For example, transmission times are subject to jitter with magnitude that is usually (but not always) small compared to epoch length e. Imperfect node clocks will drift out of synchronisation such that the relative phase offset of phase neighbour synchronisation events will inevitably change over time if the protocol is not continually adaptive.

It is therefore sufficient to limit listening in the *SYNC* state to relatively small *synchronisation windows* during which there is a reasonable expectation, though no guarantee, that peer nodes will transmit their synchronisation messages. The details are irrelevant at this point but are considered in section 8.2.3.2. In stable systems predictions will usually be reasonably accurate; prediction failures can be handled when they arise with lower total cost than under the basic policy.

Participating nodes can therefore employ *OFFDUTY* as the default state rather than *SYNC*. Nodes transition from *OFFDUTY* to *SYNC* shortly before the predicted synchronisation transmission from the preceding phase neighbour, and then back to *OFFDUTY* shortly afterwards. Some time passes in *OFFDUTY* until the assigned duty period begins, at which point the node transitions to *ONDUTY* until the duty period ends and the node transitions back to *OFFDUTY*. A further transition to and from *SYNC* occurs around the predicted synchronisation transmission from the succeeding phase neighbour. The node is then able to remain in *OFFDUTY* until the next predicted time of the preceding

phase neighbour synchronisation event. This cyclical pattern of transitions repeats with the same periodicity as the system epoch, e.

It is mentioned above that synchronisation events are not guaranteed to occur at the predicted times. This may be due to failure of the node which was due to transmit its synchronisation message, signal noise at the receiver, timing error at the transmitter or receiver, or an overly optimistic truncation of the *SYNC* state time. The protocol determines if sufficient timing data exists to make predictions in subsequent system epochs. If yes, the protocol proceeds as before, and is hence able to reject small or transient errors. If no, the node transitions from the *stable* composite state to the *unstable* composite state, and the *SCAN* simple state in particular. The protocol then restarts listening for peer nodes' synchronisation messages and transmitting its own synchronisation messages at the scheduled times.

This allows a given node to temporarily drop out of active service, without adversely affecting the cell's other nodes or the distributed schedule, rejoining soon after. If a node fails completely its disappearance will be noted in the same way, but as it will no longer transmit synchronisation messages the distributed schedule will eventually reconverge on a new equilibrium state.

8.2.1.2 Allocating duty periods

The LISP primitive described in section 7.2 obtains an equilibrium state in which a sequence of synchronisation events is evenly distributed throughout time. We now use these synchronisation events to allocate *ONDUTY* state periods. We use a method similar to that employed by the *DESYNC-TDMA* protocol [74] to mediate access to a shared wireless medium.

Recall that $\forall i : |\Delta\phi_{ij}| \to 0$ as $j \to \infty$ in successive epochs. As the system converges on the desynchronised equilibrium state, at each node S_i the phase differences $\phi_{i\beta}$ and $\phi_{i\gamma}$ between the local synchronisation event v_i and the phase neighbour synchronisation events $V_{i\beta}$ and $V_{i\gamma}$ will converge on $-\phi_{max}e/2n$ and $+\phi_{max}e/2n$ respectively.

It follows that, as the system converges toward the equilibrium state, each node can predict the time of its phase neighbours' synchronisation events with increasing accuracy. This is important as we must predict the timing of successor phase neighbour event $V_{i\gamma}$ from historical values in order to allocate duty periods that extend beyond the local synchronisation event V_i. Otherwise, we must end duty periods at the occurrence of V_i,

preventing allocation of more than 50% of each system epoch to active duty among nodes.

We define the duty period of node S_i in terms of phase offsets of phase neighbours' synchronisation events $V_{i\beta}$ and $V_{i\gamma}$, measured from the local synchronisation event V_i at local phase $\phi_i = \phi_{max}$. The duty period starts halfway between the occurrence of $V_{i\beta}$ and V_i, and ends halfway between the occurrence of V_i and $V_{i\gamma}$. To reduce the risk of two nodes' duty periods overlapping as a consequence of clock error we can scale down the duty period length l to ηl. Higher values of the scaling constant $\eta \in (0, 1]$ give longer duty periods and a greater proportion of each epoch allocated to active duty, but less unallocated inter-periodic buffer time.

Consider node S_i. Recall from section 7.2 that $\phi_{i\beta}$ is the phase offset of $V_{i\beta}$, and $\phi_{i\gamma}$ is the phase offset of $V_{i\gamma}$. The p most recent measured values of each phase offset are retained, a *null* value being stored if an expected synchronisation event is not observed. Predicted values of $\phi_{i\beta}$ and $\phi_{i\gamma}$ are taken as moving averages over historical values to reject timing noise and to cope with some missing measurements.

Within the p stored values, there are p_a non-null values in total, and the largest consecutive sequence of null values is of length p_b. There are *sufficient* measurements to predict timings of phase neighbours' synchronisation events if the ratio of non-null to null measurements $\frac{p_a}{p} > q$, and $p_b < r$. These conditions establish that predictions are based on acceptably complete and timely observations. If *insufficient* historical values have been collected, the node must re-enter the $SCAN$ state to capture more prediction data as per section 8.2.1.1.

The duty period starts at phase $\phi_i = -\lambda|\phi_{i\beta}|$ and stops at phase $\phi_i = +\lambda|\phi_{i\gamma}|$. As phase $\phi \in [0, 1)$ we apply modular arithmetic to convert the start phase value to the equivalent $\phi_i = \lambda|\phi_{max} + \phi_{i\beta}|$. We set the scaling constant $\lambda = (1 + \eta)/2$ to split the unallocated buffer time defined by η between the beginning and end of the duty period.

As the system converges on the equilibrium state defined in section 7.2 the start and stop phase offsets of duty periods will converge on $-\lambda\phi_{max}e/2n$ and $+\lambda\phi_{max}e/2n$ respectively. However, we cannot simply use these convergence limits from the outset as the protocol must align the local phase of each node with that of its phase neighbours, and hence indirectly with all nodes in the cell. The cell population n is not necessarily known by any node owing to the vagaries of initial deployment, node failures, or cell population changes. Furthermore, before convergence the relative phase of synchronisation events is in flux.

In section 8.2.1.1 we state exactly one node is *ONDUTY* at any given time, and is implicitly responsible for sending application packets if an immediate response to observed stimuli is required. Nodes in other states which create application packets must wait $\leq e/n$ time units to regain the *ONDUTY* state before transmitting.

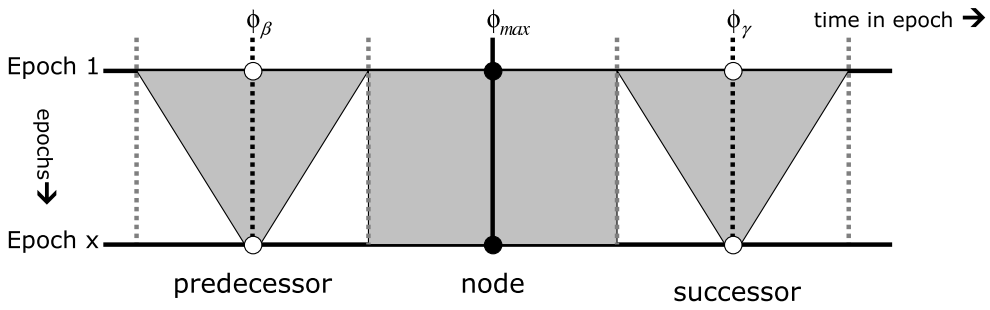


Figure 8.2: Synchronisation window management

Figure 8.2 illustrates this mechanism for a system of 3 nodes in which the LISP primitive described in section 7.2 begins in a stable equilibrium state. We focus on one specific node, the synchronisation event of which is shown at the horizontal centre of the diagram. Passage of time within a system epoch flows from left to right along the x-axis. Passage of time on a larger scale, representing the sequence of system epochs, is represented by progress from top to bottom in the y-axis. Shaded areas represent periods during which the node must be active, and unshaded areas represent periods during which a node can become inactive to conserve energy. Observe that the node must remain active throughout its assigned duty period, but the search periods during which phase neighbours fire their synchronisation events are progressively reduced. Section 8.2.3.2 considers the detail of how this reduction is achieved.

8.2.2 Measuring effectiveness

CDAP is a state management protocol but has implications for packet routing. Sensornets are composed of unreliable nodes deployed into hazardous environments. It is therefore inappropriate to route application packets by unique node identifier; some sensornets do not allocate globally unique identifiers and any individual node along delivery routes may fail.

A data-centric geographic routing policy is thus appropriate, in which packet routing decisions are based on the physical location of nodes rather than logical network topology.

As packets are routed between physical locations, and there is no guarantee that any live node is located at the exact specified destination, it follows that any node sufficiently near the specified destination is equally acceptable. CDAP determines which redundant candidate actually takes responsibility, independently of the packet content or application type.

The rôle of CDAP is to construct and dynamically maintain a duty schedule such that exactly one node is in the $ONDUTY$ state at any given time. If zero nodes are in the $ONDUTY$ state then communication between the cell and external entities will fail. If two or more nodes are in the $ONDUTY$ state then it is undefined which is responsible for external communications.

We define the following metrics: P_0, P_1 and P_2. The sensornet executes the protocol as runtime t increases in the interval $[0, \infty)$. Each metric $P_0 - P_2$ measures the proportion of time during a measurement period $[t_{start}, t_{stop}]$ which a given number of nodes are in the $ONDUTY$ state.

P_0: Proportion of time in which zero nodes are in the $ONDUTY$ state. Unitless. Defined in the range $[0, 1]$. The ideal value of $P_0 = 0$.

P_1: Proportion of time in which one node is in the $ONDUTY$ state. Unitless. Defined in the range $[0, 1]$. The ideal value of $P_1 = 1$.

P_2: Proportion of time in which two or more nodes are in the $ONDUTY$ state. We do not record the exact number of such nodes, only that there are ≥ 2. Unitless. Defined in the range $[0, 1]$. The ideal value of $P_2 = 0$.

CDAP is a dynamic protocol and hence requires some time to stabilise, attaining the equilibrium state at time t_{eq} as described in section 7.2. The system will remain within this steady state until the network changes, for example where a node joins or leaves the network cell. If $t_{start} \geq t_{eq}$ then all measurements are taken in the equilibrium state, and the values of $P_0 - P_2$ will approximate the theoretical optimal values given below. The longer the measurement period $p = t_{stop} - t_{start}$, the better the approximation as the influence of measurement granularity diminishes.

If, however, $t_{start} < t_{eq}$, the measured values of $P_0 - P_2$ will be influenced by the stabilisation period of sub-optimal behaviour prior to the system reaching equilibrium state at t_{eq}. Although this accurately reflects network performance during the measurement period, it does not necessarily reflect the long-term stable performance as the influence

of this pre-*equilibrium* period becomes insignificant as $t \to \infty$. Both measurement scenarios are correct and useful but must be interpreted appropriately; the former describes the long-term stable behaviour, and the latter describes the short-term behaviour during initialisation.

8.2.3 Energy efficiency

CDAP switches off radio modules when nodes are not on duty. Other components such as CPU, memory or sensors may optionally be powered down if this is compatible with application requirements. *Synchronisation windows* suppress this during peer synchronisation transmissions to ensure correct CDAP behaviour.

8.2.3.1 Radio module states

We define an abstract model of sensornet radio models in terms of a finite set of permitted states. We assess the energy efficiency of a sensornet based on a specific hardware platform by binding a specific power value to each defined radio module state, and measuring the time spent in each state over the runtime of a sensornet. From these measurements we can trivially calculate the energy consumed in each radio module state, and hence the average energy consumption rate for a participating sensornet node. We assume antenna gain and transmit power is fixed for all transceivers.

STANDBY - Low power mode in which nodes can neither transmit nor receive.

LISTENLOW - Low power mode in which nodes can detect nearby transmissions but not receive data.

LISTEN - Node is listening to wireless medium but not currently receiving data.

RECEIVE - Node is listening to wireless medium and currently receiving data.

TRANSMIT - Node is transmitting into the wireless medium.

Transitions between permitted states is controlled by the CDAP protocol. The lowest power state which supports required functionality is selected. In many systems idle radio listening dominates the system power budget [82], so we would like to have nodes spend more time in *STANDBY* than *LISTEN* or *LISTENLOW* when network participation does not force the *RECEIVE* or *TRANSMIT* states.

Nodes in *OFFDUTY* keep radio modules in *STANDBY*. Nodes in *SCAN* keep radio modules in *LISTENLOW*, switching temporarily to *TRANSMIT* to transmit synchronisation messages. Nodes in *SYNC* keep radio modules in *LISTENLOW*. Nodes in *ONDUTY* keep radio modules in *LISTEN*, switching temporarily to *RECEIVE* when receiving application messages or to *TRANSMIT* when transmitting synchronisation or application messages.

If a given hardware platform does not explicitly support an abstract model state we substitute the lowest cost supported state that provides the same functionality. For example, some hardware platforms support *LISTENLOW* in which nodes cannot exchange data but can detect transmissions [81], which is sufficient for synchronisation. *Zero-power* secondary radio subsystems have been proposed [112] which couple simple passive RF filter and detector circuits to CPU interrupt lines, asynchronously waking the node upon radio activity. These subsystems do not directly consume the energy reserves of the receiving node; all energy consumed by such subsystems is from the RF energy received at the antenna, and ultimately from the transmitting node. If *LISTENLOW* is not available then *LISTEN* is substituted, switching temporarily to *RECEIVE* when receiving synchronisation messages.

In *LISTENLOW* nodes cannot inspect packet contents to differentiate between application packets and unexpected synchronisation packets. The decision can be made using timing data. In section 7.2.6 we specify that synchronisation packets are as short as possible; longer transmission times imply application packets.

8.2.3.2 Window management

We cannot predict the actual times of synchronisation events with certainty. We can, however, give probabilistic guarantees that they will occur within defined finite periods. We exploit this fact by limiting costly wireless activity to these periods. In this section we describe mechanisms by which the length of the synchronisation windows is gradually reduced as the system converges on the desynchronised equilibrium state toward a final state in which the synchronisation window length reaches a specified minimum.

Within the duration of the system epoch each node has two synchronisation windows; one pertaining to the predecessor phase neighbour synchronisation event, and the other pertaining to the successor. Note that although both window lengths are likely to reduce simultaneously there is no guarantee that this will happen. For example, a given node

might transmit its synchronisation event with abnormally high jitter, or might be poorly positioned in the wireless medium landscape and hence frequently fail to be heard by its phase neighbours. We therefore track the predecessor and successor synchronisation windows separately.

It is permitted for these synchronisation windows to overlap; this overlap has no effect as both simply place a node into the *SYNC* state to listen for synchronisation transmissions of which all occurrences are equivalent.

The actual time of a synchronisation event may be earlier or later than the predicted time with equal likelihood as a result of the desynchronisation primitive described in section 7.2. We also assume that transmission time jitter and timing errors from imperfect clocks is equally likely to be positive as negative. We therefore define the synchronisation window of length μ phase units as centred on the predicted synchronisation event time, extending symmetrically by $\mu/2$ phase units in either direction.

If the synchronisation pulse requires time κ (see section 7.2.6) then we restrict μ to the interval $[2\kappa, \phi_{max}]$ to prevent the window length becoming smaller than the transmission length κ with a reasonable safety margin, and to prevent the window length becoming longer than the system epoch length e. When μ reaches the 2κ threshold, and remains there during subsequent epochs, we measure the steady state energy profile.

We provide estimates of the proportion of time nodes spend in CDAP states defined in section 8.2.1.1. For a system converged on the equilibrium state we know that the proportion of time spent in *SCAN* is 0, and the proportion of time spent in $ONDUTY = 1/n$ so we need not consider these further, but will compare experimental measurements in section 8.2.5. The proportion of time spent in *SYNC* is given as T_{sync} and the proportion of time spent in *OFFDUTY* is given as $T_{offduty}$. $T_{scan} = \mu/\phi_{max}$ and $T_{offduty} = 1-(\frac{\mu}{\phi_{max}}+\frac{1}{n})$.

8.2.3.3 Policy A: Null policy

Under the *null policy* nodes never enter the *OFFDUTY* state. The synchronisation window length is always $\mu = \phi_{max}$ such that the radio module is always in the *SYNC* state, except for assigned duty periods in which nodes temporarily assume the *ONDUTY* state. We use this policy as a baseline against which to compare the other policies as the resulting energy consumption is the upper boundary of all possible CDAP policies.

8.2.3.4 Policy B: Hyperbolic Decline policy

A counter is maintained of the number of consecutive successful synchronisation event predictions, σ. Initially, $\sigma = 0$. Every time a synchronisation event is detected within the appropriate synchronisation window σ is incremented; if the no event is detected in the window then the counter is reset as $\sigma = 0$. The window length is taken as $\mu = \phi_{max}$ until at least $\sigma = \chi$ consecutive successful predictions occur, after which $\mu = max\big(\phi_{max}/(\sigma + 1), 2\kappa\big)$. This allows the policy to rapidly shrink the synchronisation window but not until the system begins to stabilise. Increasing χ delays window shrinking for longer, reducing premature shrinking but also reducing potential energy savings.

8.2.3.5 Policy C: Moving Average Error policy

A buffer Ξ records the ξ most recent *phase prediction error* magnitudes, which are the unsigned magnitude of the difference between the predicted and measured phase for a phase neighbour synchronisation event. If the predicted event was not observed, a *null* value is recorded. Taking the average of the non-null members of Ξ provides a moving average prediction error, ϵ. If no non-null members of Ξ exist we take $\epsilon = \phi_{max}$.

We set $\mu = max(\nu\epsilon, 2\kappa)$ during each system epoch. This defines the window size in terms of actual observed prediction errors; in essence the network nodes *learn* the local timing uncertainty and adapt dynamically. $\nu \geq 1$ is a scaling constant which determines the extent to which the next phase error can be bigger than recent historical phase errors and still allow nodes to reliably detect synchronisation events. A value of $\nu \in [1, 2]$ is typical.

8.2.3.6 Measuring efficiency

We define the metrics Q as the rate at which a node consumes energy. The sensornet executes the protocol as runtime t increases in the interval $[0, \infty)$. Q measures the mean energy consumed per node per unit time during a measurement period $[t_{start}, t_{stop}]$. Unlike $P_0 - P_2$ (see section 8.2.2) these measurements apply to individual nodes rather than populations, so we measure Q for each node and take the mean to normalise metrics in cell population size n.

Q: Rate of energy consumption of a node. Measured in *Watts*. Defined in the range $(0, \infty)$. The ideal value of $Q = 0$.

As per metrics $P_1 - P_3$ defined in section 8.2.2 we observe that the measured values will differ if the measurement period considers the entire runtime of the network or is restricted to the equilibrium state. The former measures the system in transition from the initial state to the stable state, whereas the latter measures the long-term behaviour of the stable system. Both options are valid but non-equivalent, and both are measured in the experiments described in section 8.2.5.

8.2.4 CDAP state transitions

Figure 8.1 illustrates CDAP states and state transitions. The associated guard conditions are listed in table 8.1 for clarity as they cannot fit in figure 8.1. The conditions utilise *duty allocation* definitions given in section 8.2.1.2 and *window shrinking* definitions given in section 8.2.3.2.

Transitions $a - b$ describe CDAP startup and recovery from conditions in which there is insufficient timing data for normal CDAP behaviour, as defined in section 8.2.1.2. Transitions $c - h$ describe normal CDAP behaviour. If *window shrinking*, defined in section 8.2.3.2, is not implemented then transitions labelled $\{c, d\}$ are possible. If *window shrinking* is implemented then transitions labelled $\{e, f, g, h\}$ are possible. State transitions at node S_i occur at times defined by neighbour synchronisation events measured at $\phi_{i\beta}$ and $\phi_{i\gamma}$, illustrated in figure 8.2, and the parameters λ and μ, defined in sections 8.2.1.2 and 8.2.3.2 respectively.

Label	**Condition**	
	Without WS	With WS
a	$\frac{p_a}{p} \geq q \wedge p_b \leq r$	$\frac{p_a}{p} \geq q \wedge p_b \leq r$
b	$\frac{p_a}{p} < q \vee p_b > r$	$\frac{p_a}{p} < q \vee p_b > r$
c	$\phi_i = +\lambda\lvert\phi_{i\gamma}\rvert$	–
d	$\phi_i = -\lambda\lvert\phi_{i\beta}\rvert$	–
e	–	$\phi_i = \phi_{i\beta} + \frac{\mu}{2} \vee \phi_i = \phi_{i\gamma} + \frac{\mu}{2}$
f	–	$\phi_i = \phi_{i\beta} - \frac{\mu}{2} \vee \phi_i = \phi_{i\gamma} - \frac{\mu}{2}$
g	–	$\phi_i = +\lambda\lvert\phi_{i\gamma}\rvert$
h	–	$\phi_i = -\lambda\lvert\phi_{i\beta}\rvert$

Table 8.1: Guard conditions for CDAP Finite State Machine in figure 8.1

Algorithm 3 defines the behaviour of CDAP executing at each node $S_i \in \Sigma$, without window shrinking as defined in section 8.2.3.2. We assume that each node implements

LISP, as defined in section 7.2, maintaining the buffer of $\phi_{i\beta}$ and $\phi_{i\gamma}$ measurements. Variables not defined within the algorithm itself take the standard meanings used elsewhere in this document.

Algorithm 3 : CDAP at node S_i without window shrinking

Require: Duty period scaling constant, λ
Require: Predecessor phase neighbour synchronisation measure, $\phi_{i\beta}$
Require: Successor phase neighbour synchronisation measure, $\phi_{i\gamma}$
Require: Current CDAP state, s_i=$SCAN$

```
1: while monitoring local phase φ_i increasing over time do
2:    Acquire φ_iβ and φ_iγ measurements from LISP
3:    if s_i =SCAN then
4:       if p_a/p ≥ q ∧ p_b ≤ r then
5:          s_i ⇐ SYNC
6:       end if
7:    end if
8:    if s_i ≠SCAN then
9:       if p_a/p < q ∧ p_b > r then
10:         s_i ⇐ SCAN
11:      else
12:         if s_i =SYNC ∧ φ_i ≥ −λ|φ_iβ| then
13:            s_i ⇐ ONDUTY
14:         else if s_i =ONDUTY ∧ φ_i ≥ +λ|φ_iγ| then
15:            s_i ⇐ SYNC
16:         end if
17:      end if
18:   end if
19: end while
```

Algorithm 4 defines the behaviour of CDAP executing at each node $S_i \in \Sigma$, with window shrinking as defined in section 8.2.3.2.

8.2.5 Experimental results

We implemented CDAP in a modelled unicellular sensornet. We assess whether the empirical measurements match the theoretical predictions of sections 8.2.1.1 and 8.2.3.

8.2.5.1 Experimental configuration

We consider a set of homogeneous sensornets which are identical in all respects except for hardware platform. We use energy profile data for the MICA2 and MICAz motes extracted from manufacturer product data sheets [65] and two independent sets of experimentally measured energy profile data for the MICA2 mote [71, 260]. We label these energy profiles

Algorithm 4 : CDAP at node S_i with window shrinking

Require: Duty period scaling constant, λ
Require: Predecessor phase neighbour synchronisation measure, $\phi_{i\beta}$
Require: Successor phase neighbour synchronisation measure, $\phi_{i\gamma}$
Require: Current CDAP state, s_i=$SCAN$
Require: Window size, $\mu = \phi_{max}$
while monitoring local phase ϕ_i increasing over time **do**
  Acquire $\phi_{i\beta}$ and $\phi_{i\gamma}$ measurements from LISP
  if s_i =$SCAN$ **then**
    if $\frac{p_a}{p} \geq q \wedge p_b \leq r$ **then**
      $s_i \Leftarrow SYNC$
    end if
  end if
  if $s_i \neq SCAN$ **then**
    if $\frac{p_a}{p} < q \wedge p_b > r$ **then**
      $s_i \Leftarrow SCAN$
    else
      if $s_i = OFFDUTY \ \wedge \ \phi_i \geq -\lambda|\phi_{i\beta}| \ \wedge \ \phi_i < +\lambda|\phi_{i\gamma}|$ **then**
        $s_i \Leftarrow ONDUTY$
      else if $s_i = OFFDUTY \ \wedge \ \phi_i \geq \phi_{i\beta} - \frac{\mu}{2} \ \wedge \ \phi_i < \phi_{i\beta} + \frac{\mu}{2}$ **then**
        $s_i \Leftarrow SYNC$
      else if $s_i = OFFDUTY \ \wedge \ \phi_i \geq \phi_{i\gamma} - \frac{\mu}{2} \ \wedge \ \phi_i < \phi_{i\gamma} + \frac{\mu}{2}$ **then**
        $s_i \Leftarrow SYNC$
      else if $s_i = ONDUTY \ \wedge \ \phi_i \geq +\lambda|\phi_{i\gamma}|$ **then**
        $s_i \Leftarrow OFFDUTY$
      else if $s_i = SYNC \ \wedge \ \phi_i \geq \phi_{i\beta} + \frac{\mu}{2} \ \wedge \ \phi_i < -\lambda|\phi_{i\beta}|$ **then**
        $s_i \Leftarrow OFFDUTY$
      else if $s_i = SYNC \ \wedge \ \phi_i \geq \phi_{i\gamma} + \frac{\mu}{2} \ \wedge \ \phi_i > +\lambda|\phi_{i\gamma}|$ **then**
        $s_i \Leftarrow OFFDUTY$
      end if
    end if
  end if
  Recalculate μ at node S_i by selected policy defined in section 8.2.3.2
end while

$E_1 - E_4$, in this order. Each energy profile specifies the average rate of energy consumption in each of the states defined in section 8.2.3.1.

We assume that no application data packets traverse the network during the test period. The behaviour induced by CDAP is fully independent of any distributed or localised application running on the sensornet infrastructure. It is therefore unnecessary to model any sensornet application as it would have no impact on CDAP, and furthermore it is unhelpful to do so as this results in confounding of the CDAP and application influences on system energy profile.

A cell population of $n = 10$ is selected because this is an energy-efficient cluster size for typical 1000-node sensornets [312]. All experiment nodes are located in the same cell. All experiments begin with initial node phases in the same randomised distribution.

We set $\kappa = 0.01s$ as this an order of magnitude greater than the shortest complete

packets, and an order of magnitude greater than the startup latency for wireless communication subsystems transitioning from sleep states to active states [81], for the selected mote platforms. It follows that this offers a substantial safety margin. The exact value of the system epoch size e is irrelevant as we measure the passage of time in complete epochs so we set $e = 10s$ as this is orders of magnitude greater than κ. For the desynchronisation moving average parameters we set buffer size $p = 10$, required non-null proportion $q = 0.5$, and maximum consecutive nulls $r = 5$. There are no policy-specific parameters for policy A experiments. We select $\chi = 5$ for policy B experiments. We select window scaling factor $\nu = 1.5$ for policy C experiments.

In the experiments we measure the number of system epochs, j, required for each policy to reduce synchronisation window size to $\mu = 2\kappa$. We set duty period scaling factor $\eta = 1$ to evaluate worst-case duty period overlap. We measure the metrics $P_0 - P_2$ and Q for two periods; the first being the time from network start-up to CDAP during convergence, and the second being a longer duration after convergence.

8.2.5.2 State timing

Figure 8.3 shows synchronisation window shrinking against time under policies $A - C$. The response for successor and predecessor synchronisation events are very similar, but for clarity we display only the former. We see that Policies B and C significantly outperform policy A in minimising synchronisation window length, and converge to $\mu = 2\kappa$, but perform identically until the algorithms are permitted to begin window shrinking. This happens at epoch $j = \chi$ for policy B, and at epoch $j = p$ for policy C.

Policy A is trivially converged at epoch 0, policy B reaches convergence at epoch 94, and policy C reaches convergence at epoch 23. Whereas policy B induces a smoother and more predictable window size decline, policy C generally offers a smaller window size after the respective algorithms are allowed to begin. This highlights the advantage conferred by policy C *learning* network characteristics as opposed to policy B *assuming* network characteristics, where these assumptions must be pessimistic to prevent unacceptable synchronisation prediction misses. More significantly, each prediction miss requires policy B to restart at $\mu = \phi_{max}$ whereas policy C can tolerate some prediction misses before resetting $\mu = \phi_{max}$.

Note that Policies B and C confer significant efficiency improvements prior to convergence; this is simply the point at which the window size hits the predefined minimum

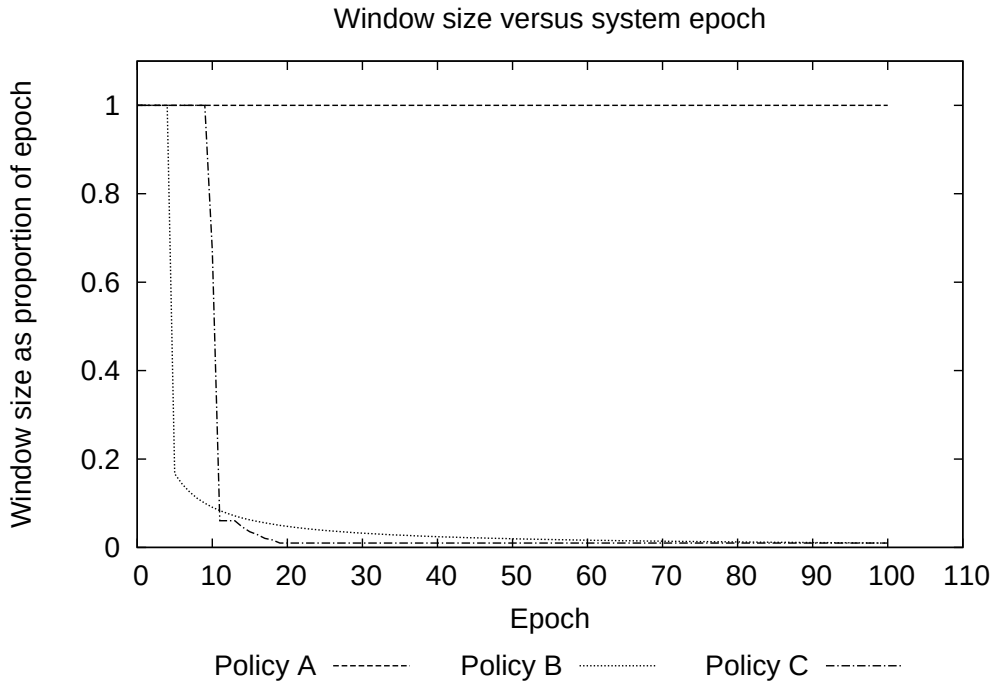


Figure 8.3: Window size

threshold, preventing the algorithms from reducing window size further. If this threshold was not enforced the policies would shrink the synchronisation window length to zero as $j \to \infty$. This is ideal in a theoretical system in which synchronisation transmissions are instantaneous, but infeasible for real systems in which $\kappa \neq 0$.

Policy	Convgd.	SCAN	SYNC	ONDUTY	OFFDUTY
A	-	0.0000	0.9000	0.1001	0.0000
B	N	0.0378	0.0590	0.1056	0.7976
	Y	0.0000	0.0200	0.0999	0.8800
C	N	0.3462	0.1536	0.1244	0.3757
	Y	0.0000	0.0203	0.0999	0.8796

Table 8.2: Proportion of time in protocol states

Table 8.2 illustrates the proportion of time spent in each CDAP state (see section 8.2.1.1). Figure 8.4 illustrates the proportion after reaching convergence. We see that under all policies $A - C$ the measured proportions match the theoretically predicted proportions. As predicted, policies B and C produce very similar results in which the majority

of time is spent in the *OFFDUTY* state. The proportion of each epoch spent by nodes in *ONDUTY* is 0.1, which is the expected value of $1/n$. When each policy has converged the time spent in *SCAN* is 0, demonstrating that the reduced synchronisation window size is compatible with accurate synchronisation event observation.

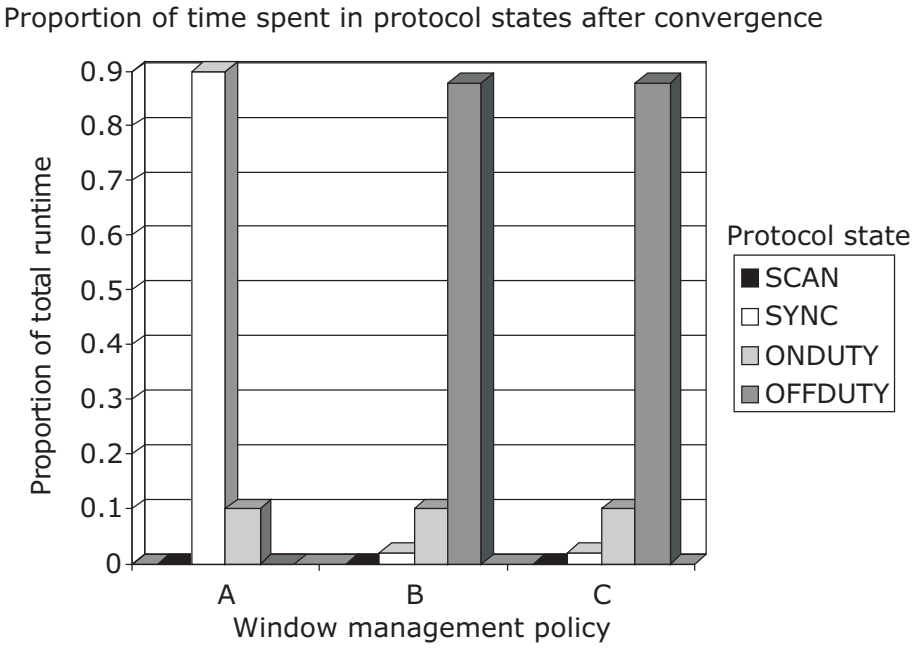


Figure 8.4: Proportion of time spent in CDAP states after convergence

For policy A the proportion spent in *SYNC* is 0.9 and the time spent in *OFFDUTY* is 0 as predicted. In contrast, under policies B and C the proportion in *SYNC* is 0.02, the minimum window size threshold 2κ, with the remaining 0.88 in *OFFDUTY*.

Prior to convergence, policies B and C display behaviour that is better than that of A but not as good as the converged behaviour. We conclude that Policies $A - C$ all assign duty periods of appropriate length, but Policies B and C can achieve this while assigning the majority of time to a low energy state.

8.2.5.3 Cell coverage

Table 8.3 illustrates the proportion of time in which 0, 1, or $\geq$2 nodes are in the *ONDUTY* state (see section 8.2.1.1). Figure 8.5 illustrates the proportion after reaching convergence. We see that under all policies $A-C$, P_0 and P_2 are very close to zero and P_1 is very close to 1 in the converged state, and hence are very close to the ideal values. Each policy $A - C$ is highly effective at maintaining mutual exclusion with exactly one node undertaking cell-wide duties at any given time.

Policy	Convgd.	P_0	P_1	P_2
A	-	0.0001	0.9948	0.0051
B	N	0.0001	0.9890	0.0109
	Y	0.0001	0.9948	0.0051
C	N	0.0100	0.9223	0.0677
	Y	0.0001	0.9945	0.0040

Table 8.3: Proportion of time for cell coverage

For policy C we see slightly poorer behaviour prior to reaching convergence. We attribute this to the relatively short convergence period of the window shrinking algorithm coinciding with the settling period of the underlying network; when convergence is attained the values of $P_0 - P_2$ are excellent.

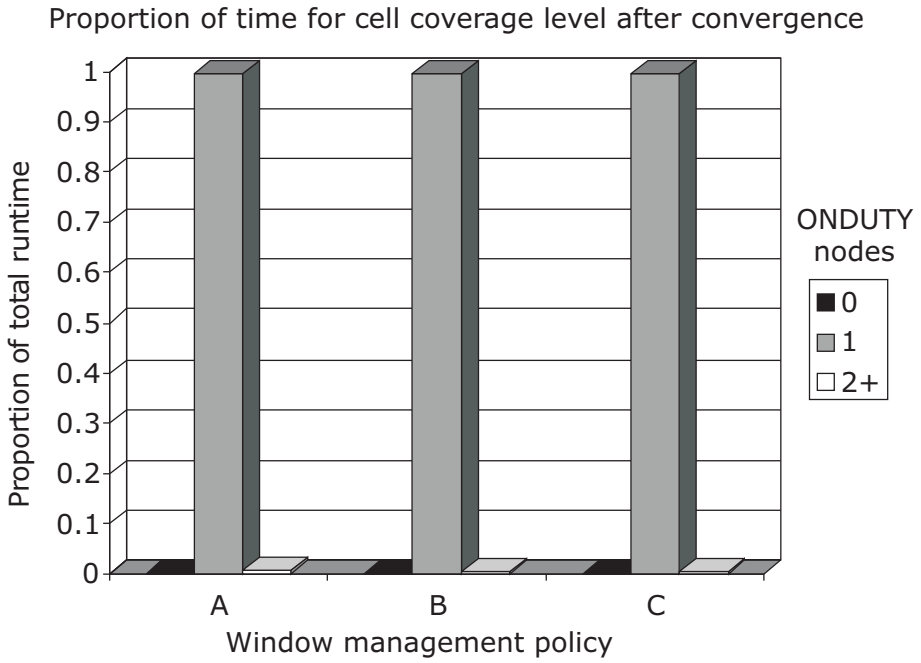


Figure 8.5: Proportion of time with cell coverage levels after convergence

8.2.5.4 Energy efficiency

Table 8.4 states values of Q, the mean energy consumption rate at each node (see section 8.2.3.6), for policies $A - C$. Figure 8.6 illustrates the relative energy consumption rates after reaching convergence. Taking policy A, the least energy efficient option, as a baseline

for comparison, we observe that policies B and C offer lower energy consumption. This is true before and after CDAP reaches convergence, with both B and C offering similar energy efficiency.

Policy	Convgd.	E_1	E_2	E_3	E_4
A	-	0.0540	0.0831	0.0441	0.0470
B	N	0.0301	0.0360	0.0168	0.0193
	Y	0.0187	0.0252	0.0125	0.0148
C	N	0.0427	0.0609	0.0317	0.0345
	Y	0.0194	0.0275	0.0135	0.0159

Table 8.4: Mean energy consumption rates (Watts)

Under energy model E_1 there are improvements of 65% and 64% under policies B and C respectively. Under E_2 the improvements are 70% and 67%, under E_3 the improvements are 72% and 69%, and finally under E_4 the improvements are 69% and 66%. Although energy models $E_1 - E_4$ differ in composition, we see a recurring qualitative outcome. Policies B and C offer significant improvement in energy efficiency over the baseline policy A, with policy B offering a slight advantage over policy C. Network designers must, however, balance this against the better reliability and shorter convergence time of policy C.

8.2.6 Managed redundancy

Section 8.2.1.2 describes the mechanism for the dynamic construction and maintenance of a duty schedule. A mutual exclusion condition is enforced such that exactly one node is in the *ONDUTY* state at any given time, excepting those brief periods in which synchronisation event transmissions occur for which there are no *ONDUTY* nodes. Each epoch of length e is divided into equal n *slots* of equal duration $\frac{e}{n}$ for a system of n nodes. Figure 8.7 illustrates a duty schedule of this type for a network cell containing 5 peer nodes. Shaded grid cells indicate that a specific node is *ONDUTY* for a specific *slot* in the dynamic schedule.

Some applications may require that two or more nodes be *ONDUTY* at any given time [103]. For example, it may be desirable to observe some physical phenomenon with

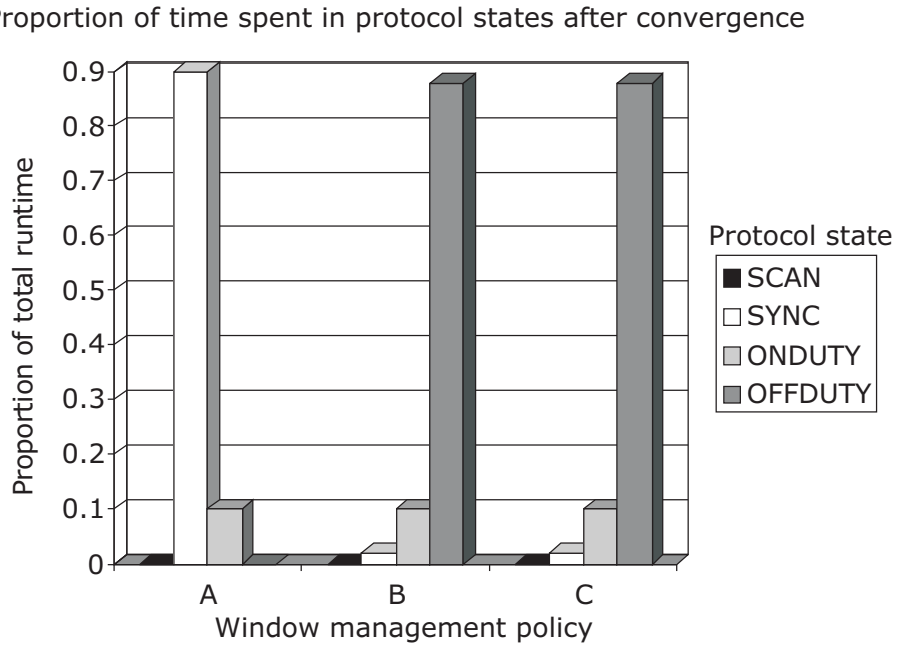


Figure 8.6: Mean energy consumption rates after convergence

multiple sensors to provide greater sampling resolution, or to capture multiple copies of some network packet to provide protection against loss from a single failed node. The duty period allocation mechanism described in section 8.2.1.2 can easily be extended to support requirements of this type.

Consider a scenario in which we wish to guarantee that m nodes are in the *ONDUTY* state at any given time. As all nodes are equal peers we do not need to consider the individual identities of the nodes sharing the *ONDUTY* state at any given time. However, we do wish to ensure that the number of active nodes is no greater or smaller than m at all times, and that all nodes share an equal burden of the overall duty workload. We assume that $m \leq n$, as it is impossible for a greater number of nodes to be active than exist.

A simple strategy would be to employ a probabilistic approach in which each node independently chooses to be *ONDUTY* during any given *slot* with probability $\frac{m}{n}$. If the probability distribution of the independent choice is well-defined it is possible to estimate the proportion of system runtime for which the required condition of m simultaneously *ONDUTY* nodes holds. However, it is highly probable that the number of *ONDUTY* nodes would be greater or lesser than m for a substantial proportion of system runtime, resulting in an unacceptable platform for the distributed application.

A better strategy builds upon the duty schedule constructed in section 8.2.1.2, retaining the principle that the duty responsibility should be exchanged cyclically between

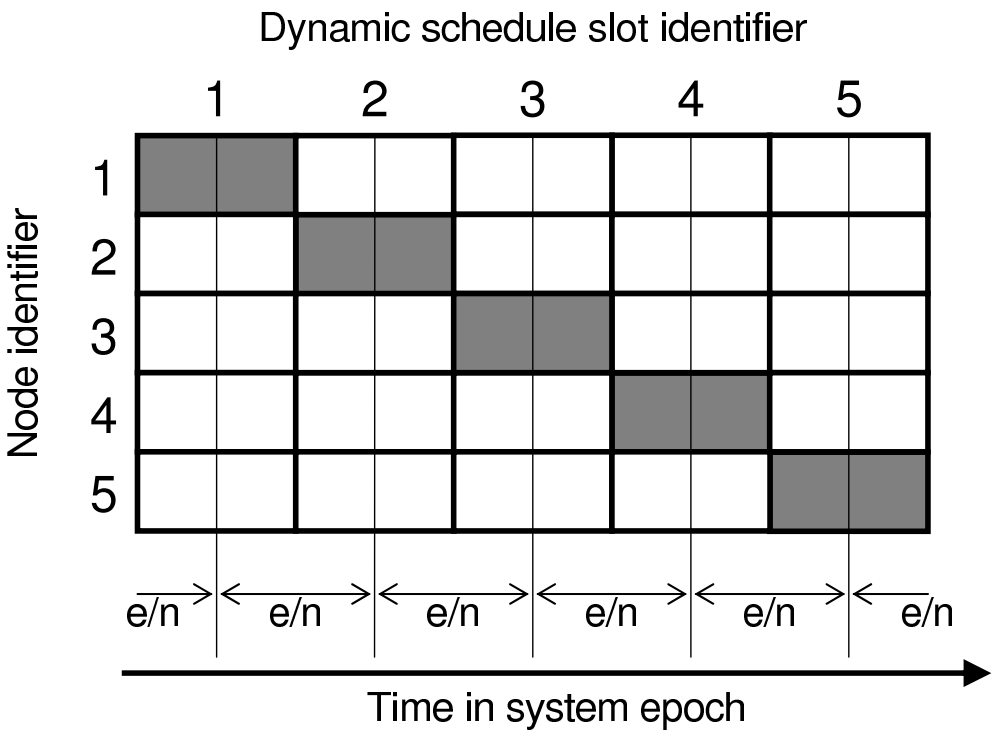


Figure 8.7: Dynamic schedule slots with duty period mutual exclusion

participating nodes. This is achieved by constructing a schedule such that each of the n nodes is active for m *slots* in each epoch. If the LISP primitive has converged to a stable equilibrium state within a network cell, as per section 7.2.3, synchronisation events are evenly distributed in time with time $\frac{e}{n}$ between consecutive examples.

Under the original algorithm described in section 8.2.1.2 each node is $ONDUTY$ for a single schedule *slot* centred around its own synchronisation event. Instead of shutting off completely at the end of this *slot*, each node remains $ONDUTY$ for a total of m consecutive *slots*. It is trivial to predict the midpoint of the xth subsequent *slot*, which follows the local synchronisation event after a delay of $\frac{x}{e}$ time units, and to assume the length of each such *slot* as being equal to that centred around the local synchronisation event. Each slot also requires the buffer periods for synchronisation transmissions discussed in section 8.2.3.2. As clock drift is negligible within a system epoch (see section 7.2.10.5) it is safe for nodes to predict the relative timing of synchronisation events for all nodes, not just the phase neighbour nodes.

Figure 8.8 illustrates a duty schedule for a network cell containing 5 peer nodes, in which we require exactly 3 nodes to be $ONDUTY$ at any given time. As in figure 8.7, shaded grid cells indicate a specific node is $ONDUTY$ for a specific *slot* in the dynamic

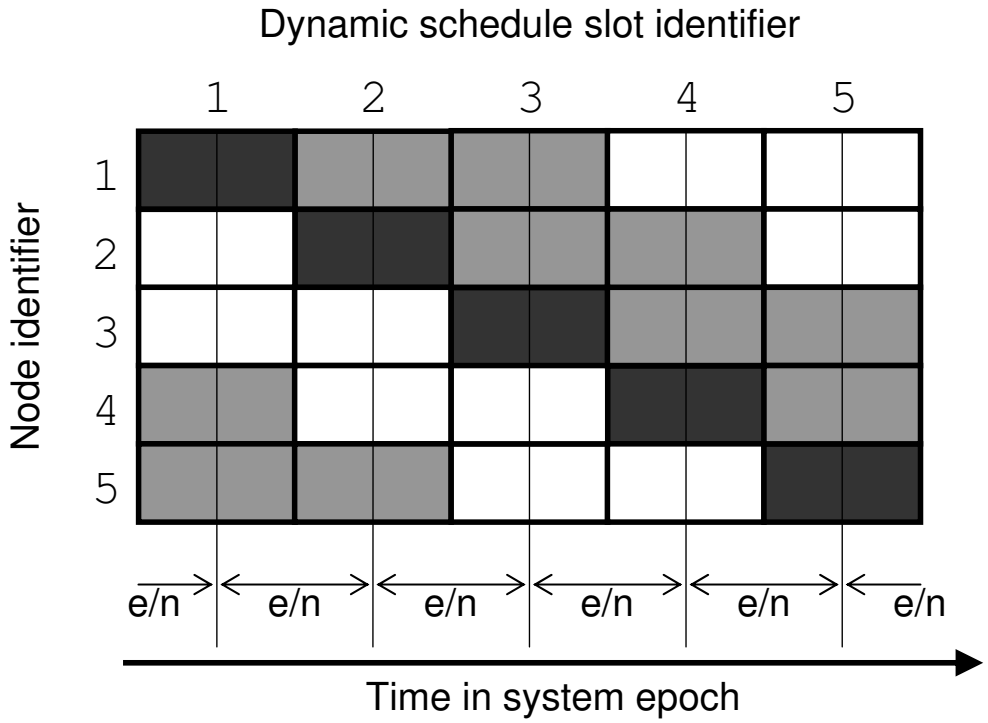


Figure 8.8: Dynamic schedule slots with managed duty period redundancy

schedule. Grid cells with darker shading represent the duty periods shared with the original version which enforces mutual exclusion; grid cells with lighter shading represent the additional *ONDUTY slots* which provide the required degree of redundancy.

At the end of each *slot*, exactly one node transitions into the *ONDUTY* state and exactly one node transitions from the *ONDUTY* state. This property holds for any values of n and m. As this is the smallest possible number of node state transitions at the boundary between *slots*, it follows that this policy implies the minimal disruption to distributed application running within the network.

8.3 Long-term duty schedule coordination

Section 8.2 describes the CDAP protocol which handles medium-term duty schedule coordination. Nodes are switched into low power modes to conserve energy when not actively required to participate in the network. Although useful, this does not address all sleep schedule coordination concerns that arise in a moderate-to-large sensornet.

Consider a situation in which the network application functions optimally with cells of n nodes, but the physical distribution of nodes is such that each geographic cell region

contains $m > n$ nodes. Some mechanism is required to regulate which nodes actively participate in the network. Too many or too few active nodes may be detrimental to the correct functioning of the distributed network application. Even if it is not harmful for too many nodes to be simultaneously active, there is little to be gained from draining the energy reserves of nodes whose input is not required; it would be better to keep such nodes in reserve for future use.

Consider some given cell containing m nodes. A static assignment of n nodes as *active* and $m - n$ nodes as *inactive* achieves the correct number of *active* nodes but is inflexible. If one or more *active* nodes should fail then this correctness property no longer holds. This is likely in sensornets deployed in hazardous environments, or composed from unreliable nodes. If the same subset of nodes is active at all times, and cycled to exhaustion despite the presence of the additional $m - n$ *inactive* nodes, then network lifetime is determined by the lifetime of individual nodes.

If the $m - n$ surplus can be considered as a pool of *spares*, a suitable mechanism can dynamically assign n nodes as *active* at any given time, with a different set of n nodes being drawn from the pool of m nodes as time progresses. This allows the fairness property that the duty burden to be shared evenly between all m nodes, while retaining the correctness property that n nodes actively participate in the network at all times. The lifetime of a single node does not determine the lifetime of the network, as a failed node is replaced in time by another node drawn from the set of spares. This also allows the operator to add more nodes after the sensornet begins operation to extend the network lifetime without the need to specify the point at which new nodes begin participation.

If a cell contains exactly $m = n$ nodes then the control mechanism should maintain n nodes active at all times. If a cell contains $m < n$ nodes then it is not possible for any control mechanism to achieve a population of n simultaneously active nodes, but keeping all m nodes active at all times yields behaviour as close as possible to the desired behaviour.

8.3.1 ADCP: Active Duty Control Protocol

The *Active Duty Control Protocol* (ADCP) regulates the population of a network cell to maintain an active population of fixed size. If too many nodes are active, some are made inactive. If too few nodes are active, some of the pool of spares are made active. It is assumed that each node is aware of the target active cell population n, and the total

number of nodes which are available for selection for active duty m, but does not have omniscient access to the current active cell population δ. In a cell of total population l there may also be nodes which do not participate in ADCP-regulated activity at some given time, but can later be brought into the reserve pool to replace failed nodes. It is assumed that any active node is capable of communicating with any other active node within its cell.

Whereas CDAP is defined in section 8.2 to allocate duty and rest periods in the medium term, with perhaps only the most energy consuming mote subsystems being powered down in off-duty periods, under ADCP nodes can be powered down more fully and for longer periods. This is important where nodes require non-trivial time to enter or leave some low-power modes. For example, during startup a node might need to reboot an operating system or recalibrate physical sensors. During shutdown, a mobile node may need to become stationary, and instantaneous acceleration to velocity of zero magnitude is impossible.

Each node acts independently, and each decision is made without reference or consultation with other nodes. Information about other nodes within a network or network cell is not actively shared or solicited, but instead is *learned* by observation. Non-active nodes do not transmit in the wireless medium. This allows ADCP to be lightweight, and guarantees that non-active nodes can not interfere with network activity supported by active nodes.

ADCP is somewhat similar in principle to the PEAS approach [335], but more sophisticated. ADCP maintains an active population of fixed size, which can be greater than one, from a specific sensornet subpopulation such as that defined by membership of a network cell. In contrast, PEAS aims to maintain only one active node within physical regions, where these regions are defined in terms of communications range rather than geography, and therefore are not consistent between nodes. Having been activated by PEAS, the single node remains active until it fails, at which point there is an unpredictable and unbounded delay before replacement occurs. Prior to replacement, there is no coverage in that region. Under ADCP, multiple nodes are active within a given region; if one node should fail, the remainder may remain able to support the application with reduced but non-zero capability. ADCP therefore prioritises the maintenance of constant active population size over strictly minimising duty cycle, which is necessary to support applications with well-defined QoS requirements, and actively seeks to share the duty burden equally

and fairly among the available node population.

8.3.2 Cell subpopulations

Each node of the network population is a member of exactly one subpopulation. The *active set*, Δ, contains nodes which are currently active within the network and are available for participation in wireless communication. The *reserve set*, Λ, contains nodes which are not currently active within the network, but which may become active in the future if required. The *inactive set*, Γ, contains nodes which do not participate in the ADCP mechanism, for example because energy reserves have become sufficiently depleted that the node cannot fulfil the requirements of the protocol.

Nodes in Γ neither transmit nor receive messages in the wireless medium. Nodes in Λ may passively listen to the wireless medium from time to time, but are not assumed to be listening at all times; when not listening, radio modules can be switched into low-power modes for energy conservation. Nodes in Λ do not transmit into the wireless medium except to announce leaving Λ to join Δ, as described in section 8.3.3. Nodes in Δ may transmit or receive messages in the wireless medium, but ADCP does not define the nature of this communication activity beside that which is required for nodes to estimate the size of cell subpopulations as described in section 8.3.5.

8.3.3 ADCP protocol states

A simple Finite State Machine runs at each node. Each node is in exactly one state at any given time. The states define the communication responsibilities of a given node at a given time with regard to peer nodes within the cell and external entities beyond the cell. As the local phase ϕ_i of node S_i increases from 0 to ϕ_{max} the protocol state may be changed by detected synchronisation events, or by state timeouts. Most state transitions occur when $\phi_i = \phi_{max}$ in response to observed network conditions.

Figure 8.9 illustrates the ADCP states and state transitions in UML statechart format. There are three composite states in which a given node can exist, corresponding to the three cell subpopulations described in section 8.3.2. Nodes in the *inactive* composite state are members of the *inactive set*, Γ. Nodes in the *reserve* composite state are members of the *reserve set*, Λ. Nodes in the *active* composite state are members of the *active set*, Δ. Section 8.3.6 defines the formulae with which the probabilities of nondeterministic transitions are calculated.

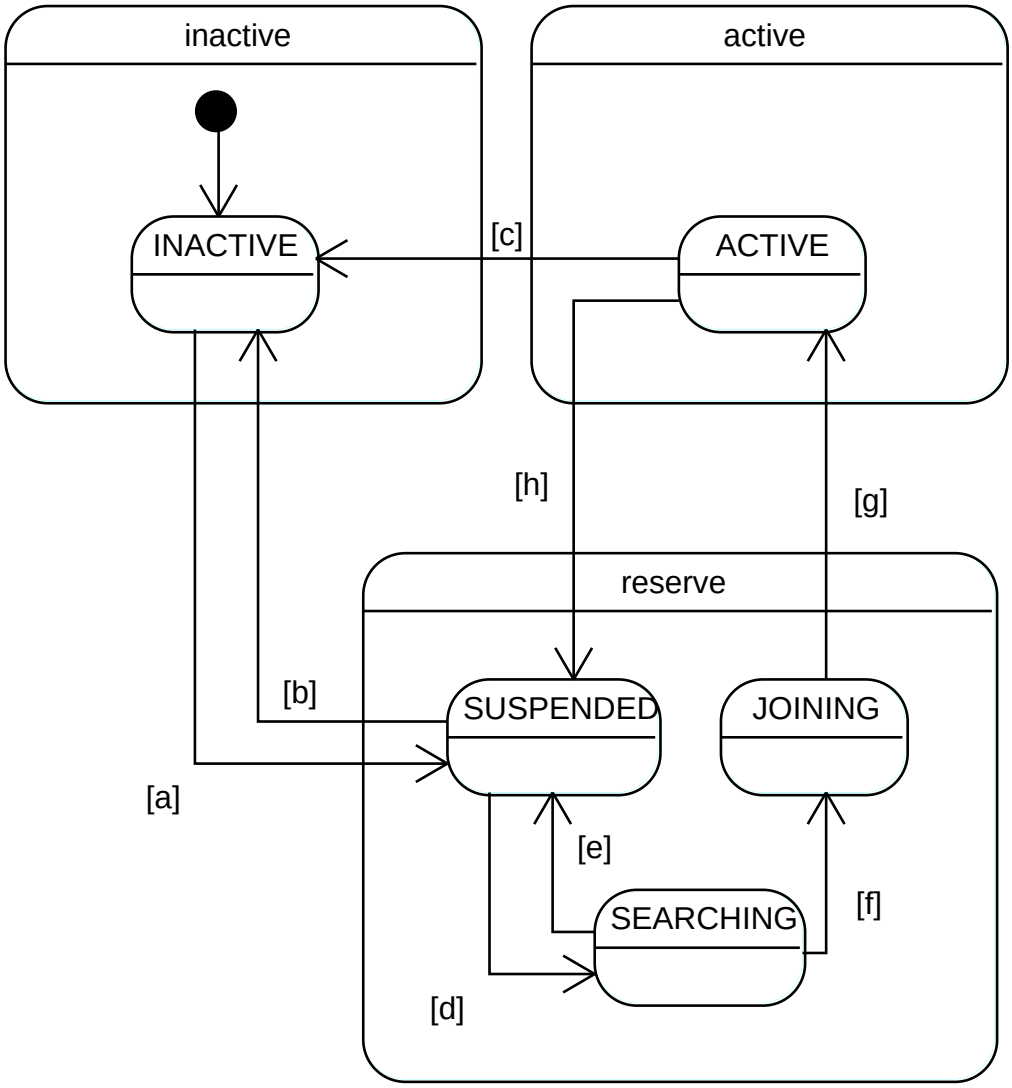


Figure 8.9: Finite State Machine for ADCP states

Now consider the simple states within the *inactive*, *reserve* and *active* composite states.

8.3.3.1 Simple states of the *inactive* composite state

INACTIVE - Node does not participate in network activity confined to the *ACTIVE* state, and is not a candidate for selection to make up the shortfall in an underpopulated network cell.

Each node starts in the *INACTIVE* state. The only possible transition is *INACTIVE*→*SUSPENDED* which moves the node into the *reserve set*, Λ, hence making the node a candidate for becoming an active participant in the network. Note that there is no direct transition *INACTIVE*→*ACTIVE*; this allows the ADCP mechanism to

stagger and regulate the release of nodes to prevent the cell becoming flooded with too many active participants.

Nodes can stay in the *INACTIVE* state for any arbitrary duration as determined appropriate by the sensornet operator. If the operator wishes to make all nodes immediately available as candidates for selection for active duty, this duration can be set to zero. This is generally a good option where the number of available nodes, m, is of the same order of magnitude as the target active population, n. It is also a good option where maximising the total lifetime of the network is of less importance than maintaining a fixed cell population.

However, if there are many more nodes in a network cell than n it may be advantageous to stagger the release of nodes into the *reserve set*, Λ, by delaying the *INACTIVE→SUSPENDED* transition. For example, it may be desired to make a subset of all nodes available immediately, but hold back the remainder for later release to replace failed nodes. If the *mean time to failure* (MTTF) is known for nodes, it is possible to stagger this release such that candidates enter the *reserve set*, Λ, at a similar rate to that at which failed nodes leave the network entirely. The primary benefit of this strategy is that the *inactive set* nodes can enter very low power modes, perhaps switching off all subsystems other than a simple timer, whereas *reserve set* nodes must occasionally switch on energy-hungry wireless communication modules to listen for network activity. It follows that keeping nodes in the *inactive set* for as long as possible maximises network lifetime.

8.3.3.2 Simple states of the *reserve* composite state

SUSPENDED - A node in the *reserve set*, Λ, is not currently active in the network but is held in reserve for future active duty.

SEARCHING - A node listens to the wireless medium for a period of equal length to one system epoch, to determine how many peers are in the *active set*, Δ. At the end of this period, the node must transition to *JOINING* if it is able and willing to rejoin Δ, or otherwise transition back to *SUSPENDED* and remain in the *reserve set* Λ.

JOINING - A node waits for an opportunity to rejoin the *active set*, Δ. The transition occurs as soon as a suitable opportunity arises, or after waiting for the duration of one epoch if no suitable opportunities arise, for example in an otherwise empty cell.

A node in the *SUSPENDED* state not currently an active participant in the network, but is held in reserve until it is required. The main difference between *SUSPENDED* and *INACTIVE* is that nodes in the former state are eligible to become members of the *active set*, Δ, if this should push a network cell toward the desired state where $\delta = n$, whereas nodes in the latter state are not eligible. Otherwise, the node is free to switch off any energy-hungry subsystems, unless they are required for some other purpose by the distributed application.

A *SUSPENDED* node occasionally transitions to *SEARCHING* to determine if its participation is required. The local timer of each node measures the progress of its local phase from 0 to ϕ_{max}, at which point the node randomly decides with *suspended-searching probability*, p_ω, to undergo the *SUSPENDED→SEARCHING* transition. If the node does not elect to undergo this transition it simply remains in the *SUSPENDED* state.

If ADCP is used to manage energy-hungry wireless communications duty, nodes in the *SUSPENDED* state will occasionally be selected for active duty, whereas nodes in the *INACTIVE* state will never be selected. Controlling the balance of nodes between the *reserve set* and the *inactive* set controls the frequency at which nodes undertake active duty and hence deplete energy reserves. If the distributed application stores data from observed traffic, retaining either raw data or some derivative, this can also control the proportion of total application data managed by any single node. If a node has depleted its energy reserves to the extent it is no longer able to be considered for active duty, it can undertake the *SUSPENDED→INACTIVE* transition. The circumstances within which this is appropriate are beyond the scope of the protocol definition.

A node entering the *SEARCHING* state as described above will remain in that state for the duration of one system epoch, e. During that time, the node produces an estimate of δ, the cardinality of the *active set*, Δ; the method by which this estimate is produced is described in section 8.3.5. The node determines whether the cell in which it resides is *underpopulated*. If *false*, the *SEARCHING→SUSPENDED* transition is always taken and the node remains in the *reserve set*, Λ. If *true*, the node *may* join the *active set*, Δ; the probability p_ψ of taking this action depends on the extent to which the cell is *underpopulated* and is calculated for each instance by the method given in sections 8.3.6 and 8.3.6.2. If this probabilistic test succeeds the node takes the *SEARCHING→JOINING* transition, and will eventually join the *active set*, Δ, otherwise it takes the *SEARCHING→SUSPENDED* transition and remains in the *reserve set*.

A node entering the *JOINING* state as described above will remain in that state until a suitable opportunity arises to join the *active set*, Δ. When this opportunity arises the node will immediately undertake the *JOINING→ACTIVE* transition, and signal this intent to its new *ACTIVE* peers.

Firstly, consider the scenario in which the LISP mechanism defined in section 7.2 is implemented in the network. Assume that all *ACTIVE* nodes listen to the wireless medium at all times. For a given node to signal its intent to become *ACTIVE*, all that is required is that it begins to transmit one *synchronisation pulse* per system epoch as described in section 7.2. The existing *ACTIVE* peers will accommodate the newcomer by the usual mechanism, starting when the first *synchronisation pulse* is transmitted and the *JOINING→ACTIVE* transition is taken.

However, *ACTIVE* nodes may not listen to the wireless medium at all times; for example, the wireless communications subsystems may be switched off periodically to reduce energy consumption. In the general case, the initial *sync pulse* transmission and *JOINING→ACTIVE* transition must occur at a time when existing *ACTIVE* peers are listening to the wireless medium.

In the specific case of a network in which CDAP is implemented, as defined in section 8.2, it is possible to make a minor change to CDAP which guarantees the *JOINING* node can identify a suitable time. Recall from section 8.2.3.2 that the *search window* size has a lower bound of $\mu = 2\kappa$, where κ is the length of one LISP *sync pulse*. We amend CDAP to require that nodes listening for *phase neighbour sync pulses* to remain listening for at least time κ after receiving any *sync pulse*. We then arrange for the *JOINING* node to listen to the wireless medium until any *sync pulse* is detected. When this observed *sync pulse* transmission completes, the *JOINING* node immediately transmits its own *sync pulse* and undertakes the *JOINING→ACTIVE* transition. This latter *sync pulse* transmission will be observed by any current *ACTIVE* peer which was listening for the former *sync pulse* as either a *predecessor* or *successor*, or which had just transmitted its own *sync pulse*.

As each node in the *active set*, Δ, must broadcast a *synchronisation pulse* exactly once per system epoch, if a node remains in the *JOINING* state for a duration of one epoch, e, it is possible to conclude that $\delta = 0$. At this point the node should immediately broadcast its first *synchronisation pulse* and undertake the *JOINING→ACTIVE* transition.

If the network does not implement the LISP primitive then the *JOINING* node should announce its intention to join the *active set*, Δ, to its new *ACTIVE* peers by some mecha-

nism appropriate to the networking regime selected by the sensornet designer. The details of any such mechanism are not within the scope of ADCP, but would typically take the form of a broadcast announcement packet as described above.

8.3.3.3 Simple states of the *active* composite state

ACTIVE - Node is in the *active set* Δ. At the end of each epoch the node either stays in Δ, or elects to transition to *SUSPENDED* and joins the *reserve set* Λ.

Nodes in the *ACTIVE* state are active participants in the network, typically implementing whatever network and computation duties are required by the distributed application running within the sensornet. The *ACTIVE* state is the only ADCP state which is not an energy-saving state, although sensornet designers will naturally wish to minimise energy consumption by application-dependent means for *ACTIVE* nodes. If a network implements both ADCP and CDAP, as defined in section 8.2, LISP- and CDAP-regulated behaviour is implemented only within the nodes assigned to the *ACTIVE* state by ADCP. This achieves the goal of ADCP providing the long-term duty schedule coordination as specified in section 7.1.1.

For maximum stability of the membership of the *active set*, Δ, we would require that any node in the *ACTIVE* state would remain as such until forced into the *INACTIVE* state, for example as a consequence of energy reserve depletion. Other nodes in the *reserve set*, Λ, would periodically determine whether to join Δ, but in each instance would determine this to be unnecessary.

However, this regime would lead to an uneven distribution of responsibility among the node population. For example, the semi-permanent members of Δ would generally deplete their energy reserves substantially quicker. Perhaps more significantly, if nodes were required to store raw sensor data, derived partial or complete computation results, or to learn the characteristics of nearby traffic flows by observation, the semi-permanent members of Δ would shoulder a disproportionate burden. If such a node were then to fail, the damage impact would also be disproportionately large.

If the characteristics of the sensornet prioritise an equitable distribution of responsibility over active population stability, we can arrange for *ACTIVE* nodes to occasionally elect to undertake the *ACTIVE→SUSPENDED* transition from the *active set*, Δ, to the *reserve set*, Λ. Each *ACTIVE* node periodically implements an independent decision to either undertake the *ACTIVE→SUSPENDED* transition with *voluntary suspension prob-*

ability, p_τ, or to remain *ACTIVE*. Any period could be employed, though if the network implements LISP as defined in section 7.2 one viable approach is for each node to measure its local phase from 0 to ϕ_{max} and implement the probabilistic decision at ϕ_{max}.

Assuming a node elects to undertake the *ACTIVE→SUSPENDED* transition, a member of the Λ will then take its place by the normal mechanism to correct an *underpopulated* cell described in section 8.3.6. The node which has just joined Λ may rejoin Δ at some point; perhaps immediately, if not specifically prohibited. Note that there is little point in setting $p_\tau > 0$ if $\delta = n$.

If a node has depleted its energy reserves to the extent it is no longer able to be considered for active duty, it can undertake the *ACTIVE→INACTIVE* transition. The circumstances within which this is appropriate are beyond the scope of the protocol definition.

8.3.4 Relationship with other protocols

ADCP is designed to integrate with the LISP primitive defined in section 7.2 and the CDAP protocol defined in section 8.2. In this section we consider the motivation for this integration, and the circumstances under which these dependencies can be removed.

ADCP manages a pool of nodes, maintaining a fixed population of n *ACTIVE* nodes. ADCP does not define other network behaviour of these; it simply delivers an *ACTIVE* population of correct size. Non-*ACTIVE* nodes do not participate in the network. Other protocols, such as CDAP, manage the behaviour and interactions of *ACTIVE* nodes which do participate in the network. Informally, we could therefore consider all CDAP states defined in section 8.2.1.1 to be substates of the ADCP *ACTIVE* state defined in section 8.3.3, and likewise for all other stateful protocols running on *ACTIVE* nodes. However, neither CDAP nor ADCP strictly requires the other, and these substates are merely an artefact of the superposition of multiple protocols running simultaneously.

Correct functioning of the ADCP mechanism is predicated on the assumption that each participating node can estimate the active subpopulation of a cell, and thus calculate the surplus or shortfall of observed active nodes as compared to the required number. In section 8.3.5 we describe how this is achieved in networks implementing the LISP primitive, defined in section 7.2, by counting the number of unique nodes broadcasting at least once within a given duration of length e. The requirement for behaviour similar to that implemented by LISP can be relaxed if some other mechanism exists through which

a node can determine how many of its peers are actively participating in the network.

It is assumed that nodes moving from the *reserve set* Λ to the *active set* Δ are able to signal their intention to their peers. In section 8.3.3 we describe how this is achieved by a node broadcasting a LISP synchronisation transmission. It is obvious that this will be unsuccessful if the currently active peers are not listening to the wireless medium at the time this broadcast occurs. Under CDAP it is known that nodes will be listening around the expected times of peer synchronisation transmissions. Joining nodes aim to transmit around the expected time for some other node when peers can be expected to be listening, as defined in section 8.3.3.3. The requirement for behaviour similar to that implemented by CDAP can be relaxed if some other mechanism exists through which a node can announce to its peers that it intends to join the set of active nodes.

8.3.5 Determining size of subpopulations

Assume we have a network, or a network cell, with a total population of l nodes, of which m are in any state other than *INACTIVE*, and a target active population of n nodes. At any given time the activity distribution of the population is such that γ nodes are in the *inactive set*, Γ, λ nodes are in the *reserve set*, Λ, and δ nodes are in the *active set*, Δ. The total number of nodes which are available, but not necessarily selected, for active duty is given by $m = \lambda + \delta$ at any given time. We need only δ, m and n to calculate the correct *state change probability* as described in section 8.3.6. We must therefore determine the absolute values associated with parameters λ and δ at some arbitrary point during the lifetime of the network.

Assume that network nodes are running the LISP primitive defined in section 7.2. We know that during any arbitrary period of length e each active node will broadcast a synchronisation transmission exactly once. Each node counts the number of synchronisation transmissions, d, observed within an epoch of length e as measured between pairs of LISP synchronisation transmissions. d estimates the number of active peers. $\delta = d+1$ estimates the total active subpopulation size, including the node making the estimate. If the value of m is known then it is trivial to find $\lambda = m - \delta$, such that the proportion of *active* nodes is $\frac{\delta}{m}$ and the proportion of *non-active* nodes is $\frac{\gamma+\lambda}{m}$.

The above works well for nodes in the *SEARCHING* state where the node is listening to the wireless medium for a complete system epoch. However, if the network implements the CDAP mechanism defined in section 8.2, nodes in the *ACTIVE* state may have their

wireless communication modules inactive for part of each system epoch to conserve energy. Under this condition the *ACTIVE* node may not be listening to the wireless medium during the synchronisation transmissions of some peers. Instead, we exploit the fact that LISP places these synchronisation transmissions evenly throughout the system epoch.

Recall from section 7.2.3 that each pair of LISP synchronisation transmissions is separated by a delay of $d = \frac{e}{n}$, where e is the epoch length and n is the target cell population. We substitute δ for n as we wish to find the current population size, which does not necessarily equal n at the time of measurement. Under CDAP each node can observe at least three synchronisations transmissions: that of its *predecessor phase neighbour*, that of its *successor phase neighbour*, and its own synchronisation transmission halfway between the other two (see section 8.2.3.2). The total time between the *predecessor phase neighbour* and *successor phase neighbour* occupies two CDAP *timeslots* in time $2d$, and can be measured directly at each node. As we know that $d = \frac{e}{\delta}$, and we have measured the value of $2d$, trivial rearrangement gives $\delta = \lceil \frac{e}{d} \rceil$. We round up to the next integer as a non-integral node count is meaningless, as is an estimate of $\delta = 0$ for a cell which must contain at least one node to implement the calculation. If no phase neighbour synchronisation transmissions exist, a node can deduce it is a *singleton* in its cell and assume $\delta = 1$.

Other methods of determining the size δ of the *active set*, Δ, exist which do not assume the LISP primitive, defined in section 7.2, is running within the network or the cell. For example, each node could listen to the wireless medium continuously. Within each period of length e, each node can record the identity of any nodes which happen to transmit a packet. At the end of the period of length e, the cardinality of the set of all unique transmitter nodes can be taken as an estimate of the *active set*, Δ, subpopulation. This approach has the advantage of zero overhead; no additional packet transmissions are required, minimising disruption and overloading of the wireless medium. However, it is not guaranteed to count all nodes in Δ unless all nodes are guaranteed to transmit periodically with frequency $f > \frac{1}{e}$. Furthermore, data packets are likely to be longer than the minimal synchronisation packets described in section 7.2.6, and the requirement to listen continually may increase the rate of energy consumption.

8.3.6 Probabilistic state transitions

Section 8.3.3 describes the state transitions possible under ADCP. Where possible these state transitions are *deterministic*. However, nodes will often have incomplete knowledge

of the network and its constituent elements. ADCP must employ some *probabilistic* state transitions, defined such that the most likely outcome is that which would be selected if complete knowledge was available.

In some cases the probability of a given state transition implemented by a given node is dependent on the perceived state of the network cell in which that node resides. More specifically, the probability that a given node either joins or leaves the *active set*, Δ, is dependent on the current number of active nodes, δ, as estimated by the methods outlined in section 8.3.5.

This probabilistic approach is required because each node is an equal peer and does not have an omniscient view of the network cell. It follows that there is no central controller, or distributed coordination mechanism, to decide which of the eligible nodes should make a given transition. Consider a network cell in which $\delta \neq n$. The greater the difference between observed and required active population size, $\epsilon = \delta - n$, the greater the number of nodes which will need to undergo a state transition to push the network cell toward the desired condition of $\delta = n$ where $\epsilon = 0$.

Ideally, we would like all nodes undertaking these transitions to implement the appropriate change as soon as possible. From section 8.3.5 we know that each node is equipped with the value of n and an estimate of δ. Over a period of length e, each node will produce its estimate of ϵ and will decide independently whether it shall contribute toward the change in active cell population. If nodes could cooperate and coordinate their decisions, it would be trivial to define a simple algorithm to decide which eligible nodes undergo a given state transition during some system epoch.

However, without some centralised or resource-consuming distributed control mechanism, it is possible to achieve similar results by implementing probabilistic methods. The probability that a given node undergoes a given state transition is dependent on the number of nodes required to undergo this transition. A feedback loop is created in which nodes observe the network as their local phase increases from 0 to ϕ_{max}, decide whether to take action, with the results of this decision then being observed during the next epoch.

As each node does not cooperate or coordinate its decision with that of its peers, we apply a probabilistic approach in which state transition probabilities are calculated as a function of the *overpopulation* or *underpopulation* of a network cell. Each node has an equivalent view of the cell such that, within a system epoch, each node will independently calculate the same probability p for each state transition. It follows that the number

of nodes which independently decide to undergo a given state transition, x, in a given system epoch observes a *binomial distribution* [38]. x is distributed as $x \sim B(y, p)$ where y is the number of nodes which are eligible to undergo a given state transition and p is the probability that an eligible node independently chooses to do so.[1]

It is obvious that probabilistic decision making of this type is not guaranteed to correct the active cell population $\delta = n$ within the duration of a single system epoch of length e. For any given probability p, it is entirely possible that the number of nodes from a set of c candidates which independently decide to implement state transition could be any value in the interval $[0, c]$. Furthermore, as the *local phase* of nodes is not equal (see section 7.2.2), and each node α makes the probabilistic decision as its *local phase* $\phi_\alpha = \phi_{max}$, the estimate of δ at a given node may become inaccurate as other nodes independently leave or join the *active set*, Δ.

Assume that all nodes in $\Lambda \cup \Delta$ are listening to the wireless medium and hence are eligible to undergo state transition to correct the active cell population. If each node has an independent source of reasonably random numbers from which to make the probabilistic decision, and x observes a binomial distribution, it follows that the *expected value* is given by $E(x) = yp$ with variance $V(x) = yp(1 - p)$ [38]. We define the formulæ with which to calculate p below, such that $yp = |\epsilon|$. It follows that the *expected value* of x within any given system epoch is equal to the number of nodes which must undergo a state transition to restore the *active set* cardinality to the desired value such that $|\Delta| = n$. By the *Law of Large Numbers* [38], the average value of x tends to approach and stay close to the expected value across multiple system epochs.

As a consequence, the most likely outcome at the end of a system epoch is that the composition of all independent decisions yields a network cell with the desired number of *ACTIVE* nodes, regardless of the values of δ and ϵ at the start of that epoch. It is possible that there is a mismatch between x and ϵ such that the number of nodes which independently decide to undergo the state transition is greater or lesser than the

[1] If ADCP was modified to allow a *leaving* or *joining* node to immediately signal the outcome of its probabilistic decision to its peers, and those peers immediately took account of this information in determining the probability of their own non-deterministic decision taking, the number of nodes independently undertaking a given probabilistic decision would observe a *hypergeometric distribution* rather than a *binomial distribution*. However, the binomial distribution is a good approximation of the hypergeometric distribution, particularly for large populations, and in any case sensornets cannot generally support immediate state information distribution where nodes are sometimes not listening to the wireless medium.

ideal. However, as the binomial distribution can be approximated well by a Gaussian distribution, it is likely that any such mismatch is small; the probability of a given x value falls away quickly either side of the *expected value*, $|\epsilon|$. Any remaining discrepancy between δ and n is corrected during subsequent system epochs by the same mechanism. Because each decision is probabilistic and independent, it is not possible to give a *hard guarantee* that δ will ever be equal to n, but it is possible to define the probability distribution function for the number of epochs taken for δ to reach n.

Good sources of pseudorandom numbers, essential for each node to make an independent probabilistic decision, can be obtained by algorithms such as *Mersenne Twister* [197] with acceptable overhead. Appropriate seed values can be obtained from entropy sources such as physical sensor hardware or network packet timing data. To prevent excessive volatility it is possible to *damp* the effect by applying a scalar in the interval $(0, 1]$ to the state transition probabilities as described below. This may be useful in sensornets where *lost* or *phantom* synchronisation transmissions (see section 7.2.10.4) tend to give unreliable estimates of δ, but is not strictly necessary. It may also be useful if the network operator wishes to ensure that *underpopulated* and *overpopulated* cell conditions are corrected at the same rate by retarding the correction of an *overpopulated* cell. If nodes in the *reserve set*, Λ, spend significant time in the *SUSPENDED* state in which they are not eligible to undergo the *SEARCHING*→*JOINING*→*ACTIVE* sequence, *overpopulation* might otherwise be corrected faster than *underpopulation*.

Recall that we wish to obtain a situation in which the cell population δ equals the target population n. It follows that the cell can be *overpopulated*, *underpopulated*, or *correctly populated*. We calculate the value of p under each possibility below in sections 8.3.6.1, 8.3.6.2, and 8.3.6.3. To recap, the probability calculation uses several measures of cell population. n is the target *active* population, δ is the current *active* population, and m is the total usable *active* and *reserve* population. Where a number of nodes must undertake a given state transition to move δ toward n, $\epsilon = \delta - n$ is the desired change in *active* population, y is the number of nodes eligible to undergo this transition, and x is the number of nodes which independently choose to undergo this transition during a system epoch.

8.3.6.1 Overpopulated cell: $\delta > n$

For an *overpopulated* cell we know that the appropriate state transition to correct the cell population is the *ACTIVE→SUSPENDED* transition. Only nodes in the *active set*, Δ, are considered eligible to undergo this transition, and all members of Δ are eligible during each system epoch. It follows that $x \sim B(y_a, p_\chi)$ where the number of eligible *ACTIVE* nodes $y_a = \delta$. $p_\chi = 0$ unless the cell is *overpopulated*.

p_χ is the calculated probability that a given node will independently choose to take the *ACTIVE→SUSPENDED* transition. Recall from section 8.3.6 that we wish to calculate p_χ such that $y_a p_\chi = |\epsilon|$. A trivial rearrangement yields $p_\chi = \frac{|\epsilon|}{\delta}$ where $y_a = \delta$.

$\pi_\chi \in (0, 1]$ is the *node suspension coefficient*, which is a scaling factor applied to p_χ to reduce the rate at which nodes move from the *active set*, Δ, to the *reserve set*, Λ. To utilise the *node suspension coefficient* we extend the definition of p_χ given above to $p_\chi = \pi_\chi \frac{|\epsilon|}{\delta}$. As $\pi_\chi \neq 0$ it is always possible for the cell to progress toward the target active population n, and as $\pi_\chi \leq 1$ the *expected value* $E(x) = \pi_\chi \frac{|\epsilon|}{\delta} y_a = \pi_\chi |\epsilon|$ never exceeds that of the simple case in which $\pi_\chi = 1$. A typical use of π_χ is to balance the rate at which *underpopulated* and *overpopulated* cell conditions are corrected where $p_\omega \neq 1$.

Given $\delta > n$, it is trivially always true that there are sufficient candidates in Δ which can undertake the *ACTIVE→SUSPENDED* transition, so it is always true that we can find a suitable $p_\chi \in (0, 1]$ such that the *expected value* $E(x) = |\epsilon|$.

8.3.6.2 Underpopulated cell: $\delta < n$

For an *underpopulated* cell, the appropriate state transitions to correct the cell population are those of the *SEARCHING→JOINING→ACTIVE* sequence. Only nodes in the *reserve set*, Λ, can undertake this sequence. However, at any given time it is not guaranteed that any given node in Λ is in the *SEARCHING* state; listening to the wireless medium in the *SEARCHING* state is less energy efficient than the *SUSPENDED* state in which the communications hardware can be switched off, so it is desirable for nodes to spend more time in the latter state than the former.

Recall from section 8.3.3.2 that a *SUSPENDED* node independently decides with probability p_ω whether to enter the *SEARCHING* state at the end of each local epoch. This decision is independent of the probabilistic state transition considered in this section. The proportion of *reserve set* nodes in the *SEARCHING* state is also p_ω. It follows that $x \sim B(y_s, p_\psi)$ where the number of eligible *SEARCHING* nodes $y_s = \lambda p_\omega$. From section

8.3.5 we know that $\lambda = m - \delta$. $p_\psi = 0$ unless the cell is *underpopulated*.

p_ψ is the calculated probability that a given node will independently choose to take the *SEARCHING→JOINING* transition, which inevitably leads to the *JOINING→ACTIVE*. Recall from section 8.3.6 that we wish to calculate p_ψ such that $y_s p_\psi = |\epsilon|$. A trivial rearrangement yields $p_\psi = \frac{|\epsilon|}{(m-\delta)p_\omega}$ where $y_s = \lambda p_\omega$.

$\pi_\psi \in (0, 1]$ is the *node activation coefficient*, which is a scaling factor applied to p_ψ to reduce the rate at which nodes move from the *reserve set*, Λ, to the *active set*, Δ. To utilise the *node activation coefficient* we extend the definition of p_ψ given above to $p_\psi = \pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega}$. As $\pi_\psi \neq 0$ it is always possible for the cell to progress toward the target active population n, and as $\pi_\psi \leq 1$ the *expected value* $E(x) = \pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega} y_s = \pi_\psi |\epsilon|$ never exceeds that of the simple case in which $\pi_\psi = 1$. A typical use of π_ψ is to balance the rate at which *underpopulated* and *overpopulated* cell conditions are corrected where $p_\omega \neq 1$.

Given $\delta < n$, it is not always true that there are sufficient candidates in Λ which can undertake the *SEARCHING→JOINING* transition such that the *expected value* $E(x) = |\epsilon|$. If $y_s < |\epsilon|$ this would imply $p_\psi > 1$, which is obviously impossible.[2] As probabilities must be in the range $[0, 1]$ we require that the value of p_ψ used by a given node when deciding whether to undergo the state transition is given by $p_\psi = \max(\pi_\psi \frac{|\epsilon|}{(m-\delta)p_\omega}, 1)$. If $\lambda p_\omega < |\epsilon|$, forcing $p_\psi = 1$, the *expected value* is trivially $E(x) = \lambda p_\omega$ as the binomial distribution is effectively reduced to a uniform distribution. It follows that the *expected value* of the *active set* population size $\delta < n$ after a single system epoch passes, although $\delta \to n$ in successive system epochs as system time $t \to \infty$ provided that $(\lambda + \delta) \geq n$.

8.3.6.3 Correctly populated cell: $\delta = n$

The probability that an *ACTIVE* node volunteers to undertake the *ACTIVE→SUSPENDED* transition is fixed at $p_\tau \in [0, 1]$; this is independent of δ. Only nodes in the *active set*, Δ, are considered eligible to undergo this transition, and all members of Δ are eligible during each system epoch. Each eligible node independently decides, with the same probability p_τ, whether to undergo the transition. It follows that

[2] Unless we accept an interpretation in which a given *SEARCHING* node a has the ability to move some set of other nodes A from the *inactive set*, Γ, to the *reserve set*, Λ, with each of the $\{a\} \cup A$ nodes probabilistically transitioning to *ACTIVE* by the usual manner but with probability scaled by the factor $\frac{1}{|\{a\} \cup A|}$. However, this would require additional control mechanisms that are beyond the scope of ADCP, and assumes the existence of sufficient *INACTIVE* nodes to make up the shortfall.

$x \sim B(y_a, p_\tau)$ under a binomial distribution where the number of eligible *ACTIVE* nodes $y_a = \delta$. $p_\tau = 0$ unless the cell is *correctly populated*.

The network designer must select an appropriate balance between cell population stability, favoured by lower values of p_τ, and equitable burden distribution, favoured by higher values of p_τ. One approach is to define the desired level of population volatility and work backward toward a value of p_τ. This is possible because the probability density function is well defined for binomial distributions. If we wish the average number of nodes moving from Δ to Λ per system epoch to be z, we set the required *expected value* of the distribution as $E(x) = z$. As $E(x) = y_a p_\tau$, and $y_a = \delta$, we can extract $p_\tau = \frac{z}{\delta}$.

Typically, the network designer would wish to define $z < (m - n)p_\omega$, such that the average number of *ACTIVE* nodes electing to undergo the *ACTIVE→SUSPENDED* from a correctly populated cell is lower than the average number of *SEARCHING* nodes in the newly-underpopulated cell able to address the shortfall. This prevents the size of the *active set*, Δ, from deviating too far from the ideal value $\delta = n$ for long durations, while still allowing some level of exchange between the *active set*, Δ, and *reserve set*, Λ, to distribute burden fairly between the set of all candidate nodes in the long term.

8.3.7 ADCP state transitions

Figure 8.9 illustrates ADCP states and state transitions. The associated guard conditions are listed in table 8.5 for clarity as they cannot fit in figure 8.9.

Transitions $\{a, b, c\}$ relate to nodes joining or leaving the sensornet; precondition definitions are application-specific and beyond the scope of ADCP. Nodes may *become available* when present in the network from the start, or when added later, and nodes may *cease to be available* owing to hardware failure or depleted power supply; these actions may be beyond application control. However, nodes may *become available* owing to staggered wakeup as discussed in section 8.3.3.1, or *cease to be available* when attempting to replenish depleted energy supplies [145]. Mobile nodes may also *become available* or *cease to be available* when entering or leaving the sensornet physical region.

Transitions $\{d, e, f, g, h\}$ relate to internal ADCP activity for nodes currently within a sensornet. State transitions utilise cell subpopulation estimation methods defined in section 8.3.5 to find the population error estimate ϵ as perceived by each node, and the probability parameters p_ω, p_ψ, p_χ and p_τ, as defined in section 8.3.6. The random number $r \in [0, 1]$ is generated when a probabilistic decision is required, and compared against

the probability calculated for that specific decision; if the test succeeds, the transition is possible. Any transition for which the guard contains r is a non-deterministic transition. All other transitions for which the guard does not contain r are deterministic.

Label	**Condition**
a	*node becomes available*
b	*node ceases to be available*
c	*node ceases to be available*
d	$r > p_\omega$
e	$\epsilon \geq 0$
f	$\epsilon < 0 \wedge r > p_\psi$
g	$\epsilon < 0$
h	$(\epsilon = 0 \wedge r > p_\tau) \vee (\epsilon > 0 \wedge r > p_\chi)$

Table 8.5: Guard conditions for ADCP Finite State Machine in figure 8.9

Algorithm 5 defines the behaviour of ADCP executing at each node $S_i \in \Sigma$, using the probabilistic state transitions defined in section 8.3.6 and cell subpopulation estimation methods defined in section 8.3.5. Variables not defined within the algorithm itself take the standard meanings used elsewhere in this document.

8.3.8 Cost analysis

ADCP is a lightweight protocol. The only required data storage is for a single integer at each node, used to store the estimate of *active set* cardinality, δ, upon which the probabilistic state transition decisions described in section 8.3.6 are based. It follows that ADCP storage cost is O(1) in node population.

For a node in the *SEARCHING* state, the δ estimate is set to an initial value of 0 and incremented each time another member of the cell broadcasts a *sync pulse* transmission. Under ADCP, only the δ *ACTIVE* nodes broadcast LISP *synchronisation transmissions*. It follows that ADCP algorithmic cost is O(n) in cell population. Other ADCP behaviour, such as the probabilistic state transition decisions, occur at each node exactly once per system epoch and are independent of population size; ADCP is O(1) in network size for these behaviours, but O(1) is subsumed by O(n).

Algorithm 5 : ADCP at node S_i

Require: Target cell population, n
Require: Suspended-Searching probability, p_ω
Require: Correctly populated cell Active-Suspended probability, p_τ
Require: Current ADCP state s_i =*SUSPENDED*

```
1: while monitoring local phase φ_i increasing over time do
2:   if φ_i = φ_max then
3:     if s_i =SUSPENDED then
4:       generate random number r ∈ [0, 1]
5:       if r > p_ω then
6:         s_i ⇐ SEARCHING
7:       end if
8:     else if s_i =SEARCHING then
9:       estimate difference between current and target active population, ϵ
10:      if ϵ ≥ 0 then
11:        s_i ⇐ SUSPENDED
12:      else
13:        calculate Searching-Joining probability, p_ψ
14:        generate random number r ∈ [0, 1]
15:        if r > p_ψ then
16:          s_i ⇐ JOINING
17:        end if
18:      end if
19:    else if s_i =JOINING then
20:      fire own synchronisation event at node S_i
21:      reset φ_i = 0
22:      s_i ⇐ ACTIVE
23:    else if s_i =ACTIVE then
24:      estimate difference between current and target active population, ϵ
25:      if ϵ > 0 then
26:        calculate overpopulated cell Active-Suspended probability, p_χ
27:        generate random number r ∈ [0, 1]
28:        if r > p_χ then
29:          s_i ⇐ SUSPENDED
30:        end if
31:      else if ϵ = 0 then
32:        generate random number r ∈ [0, 1]
33:        if r > p_τ then
34:          s_i ⇐ SUSPENDED
35:        end if
36:      end if
37:    end if
38:  else if s_i =JOINING then
39:    if another node S_j ≠ S_i has just broadcast its synchronisation event then
40:      fire own synchronisation event at node S_i
41:      reset φ_i = 0
42:      s_i ⇐ ACTIVE
43:    end if
44:  end if
45: end while
```

8.3.9 Energy efficiency

The primary objective of ADCP is to manage the size of the *active set*, Δ. This is to ensure that any application-level requirements for cell coverage are satisfied by an appropriate

number of active cell members at any given time; not too many, and not too few. However, a secondary objective is to minimise network energy consumption so as to extend the useful lifetime of the network. We must therefore consider the energy characteristics of a system in which ADCP is active.

For a cell containing l nodes in total, we have cell subpopulations Γ, Λ and Δ such that $\gamma + \lambda + \delta = l$. Within each subpopulation we assume nodes have a similar average rate of energy consumption. We label these energy consumption rates w_Γ, w_Λ and w_Δ for subpopulations Γ, Λ and Δ respectively.

By the definitions given in section 8.3.2 it is reasonable to assume that all members of each subpopulation consume energy at a similar rate, if this rate is defined over a reasonable duration. In the case of sensornets implementing the protocols defined in this chapter a reasonable duration would span several system epochs, in order to smooth out the transient behaviour implicit within each epoch. Nodes in the *inactive set*, Γ, are dormant; provided all are equally dormant, it is trivially true that they will consume energy at a similar rate. Nodes in the *reserve set*, Λ, are largely inactive but occasionally assess whether they are needed in the *active set*, Δ. Provided that all nodes in Λ are equally likely to perform this assessment in a given epoch, which is true under ADCP (see section 8.3.6.2), all will consume energy at a similar rate. Nodes in the *active set*, Δ, may consume energy at different rates at different times within a system epoch if CDAP is implemented (see section 8.2), or at a similar rate at all times if CDAP is not implemented. In either case, the average rate of energy consumption is equal for all nodes if measured across an entire epoch, as CDAP allocates equal duty periods within the epoch.

It follows that we can estimate the energy consumption rate of a network cell at a given time t, if we know the size of the subpopulations at t and the average rate of energy consumption of a node within a subpopulation. The total rate of energy consumption across an entire network cell, w_{total}, is given in equation 8.1.

$$w_{total} = \gamma_t w_\gamma + \lambda_t w_\lambda + \delta_t w_\delta \tag{8.1}$$

w_{total} gives the average rate of energy consumption for a stable system, where the rate of energy conversion is measured in terms of some unit of time. To obtain a measure of total energy consumption we simply multiply w_{total} by the number of time units required.

If we know the total number of nodes, l, the target cell population, n, and the number of nodes which are eligible for *ACTIVE* duty, m, we can obtain the average rate of energy

consumption for a network cell in the steady state rather than at some specific time t. From section 8.3.2 we know that $\delta = n$, $\lambda = m - n$, and $\gamma = l - m$.

$$w_{total} = (l - m)w_{\gamma} + (m - n)w_{\lambda} + nw_{\delta} \tag{8.2}$$

Consider the energy profile of nodes in the *reserve set*, Λ. From section 8.3.6.2 we know that the proportion of nodes in Λ which are in the *SEARCHING* state is on average equal to p_{ω} over the duration of a system epoch. Assume that all energy-consuming activity is regulated by CDAP and ADCP, and owing to the energy characteristics of wireless communications subsystems we can ignore other energy consumption effects. It follows that *SEARCHING* and *JOINING* nodes in Λ have similar energy consumption to *ACTIVE* nodes in Δ, and similarly that *SUSPENDED* nodes in Λ have similar energy consumption to *INACTIVE* nodes in Γ. If these assumptions hold for a given sensornet we can obtain equation 8.3.

$$w_{total} = (l - m + (1 - p_{\omega})(m - n))w_{\gamma} + (n + p_{\omega}(m - n))w_{\delta} \tag{8.3}$$

8.3.10 Measuring effectiveness

We define the metrics $R_1 - R_3$ to measure the effectiveness of ADCP. As ADCP has multiple objectives, it follows that we require multiple metrics with which to measure success against these objectives as runtime t increases in the interval $[0, \infty)$. The metrics are sampled at a set of evenly spaced points in system runtime t during a measurement period $[t_{start}, t_{stop}]$.

Firstly, the protocol aims to manage the size δ of the *active set*, Δ, population. R_1 simply measures the value of δ at some time of interest; ideally, $\delta = n$. This measures the effectiveness of ADCP at managing the active cell subpopulation.

Secondly, the protocol aims to reduce energy consumption to the minimum required to support the required level of service. R_2 measures the proportion of total network runtime during which nodes are in the *SUSPENDED* state, and hence are not required to participate in intra- or inter-cellular communications. It follows that, in the *SUSPENDED* state, energy-hungry wireless communications modules can be switched off to conserve energy. This measures the effectiveness of ADCP at enabling energy efficiency improvements.

Thirdly, the protocol aims to share the duty burden evenly within the cell population. R_3 measures the standard deviation of the proportion of total runtime nodes spend in the

active set, Δ. Nodes in the *inactive set*, Γ, are not candidates for active duty and do not contribute to this metric. This measures the effectiveness of ADCP at sharing the duty burden between the candidates for active duty.

As with the CDAP metrics (see section 8.2.2) the R_1 values are defined across the entire network cell, and are not meaningful at the level of an individual node. As with LISP metrics (see section 7.2.5) the R_2-R_3 values are defined across the entire network cell, but are found by taking measurements at individual nodes and aggregating across the entire cell population.

R_1: The size δ of the *active set*, Δ, population at some time t. Note this is the *actual* value of δ available to an omniscient observer, rather than the *estimate* used by cell members in the probability calculations given in section 8.3.6. Measured in *nodes*. Defined in the range $[0, l]$ where l is the total number of nodes in a network cell. The ideal value of $R_1 = n$.

R_2: The proportion of total network runtime during which nodes are in the *SUSPENDED* state, taken as the average across all l nodes, during which nodes are permitted to switch off energy-hungry wireless communications modules. Unitless. Defined in the range $[0, 1]$. The ideal value of $R_2 = 1$.

R_3: The standard deviation of proportion of time for which nodes are assigned to the *active set*, Δ, across all non-*INACTIVE* nodes in the set $(\Lambda \cup \Delta)$. Unitless. Defined in the range $[0, 1]$. The ideal value of $R_3 = 0$.

8.3.11 Experimental results

We implemented ADCP in a modelled unicellular sensornet. A series of experiments was performed to establish the effectiveness of ADCP in terms of the metrics defined in section 8.3.10.

8.3.11.1 Population management: Cold start

The behaviour of ADCP was examined in a *cold start* scenario. Initially all nodes are in the *reserve set*, Λ, such that $R_1 = 0$, corresponding to the conditions expected when a new sensornet is deployed into the environment. As time progresses, we expect that $R_1 \to n$ as nodes independently choose to join the *active set*, Δ, and the network cell approaches its required operating active population.

We set total cell population higher than target active population such that $l > n$ to illustrate correct behaviour where cells have surplus nodes. All nodes begin in the *SUSPENDED* state as per figure 8.9. In the initial state $R_1 = 0$, corresponding to an *underpopulated cell* as defined in section 8.3.6.2. It follows that the relevant ADCP parameters are the *suspended-searching probability*, p_ω, defined in section 8.3.3.2, and the *node activation coefficient*, π_ψ, defined in section 8.3.6.2. However, we must also specify values for the *voluntary suspension probability*, p_ω, as the cell will eventually be correctly populated as defined in section 8.3.6.3, and for the *node suspension coefficient*, π_χ, as the cell may temporarily have too many active nodes as the population settles.

We first consider the development of active cell population, R_1, as a function of time, t, as measured in system epochs of length e time units. We set $l = 20$ and $n = 10$ such that the cell contains more nodes than are required to be active, to illustrate the fact that active cell population does not simply increase without bound as a function of time.

We set the *node activation coefficient* $\pi_\psi = 1$, and the *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set the *voluntary suspension probability* $p_\tau = 0$ as we do not wish to obscure the population stabilisation effect with the fair duty allocation effect which will be considered in section 8.3.11.5.

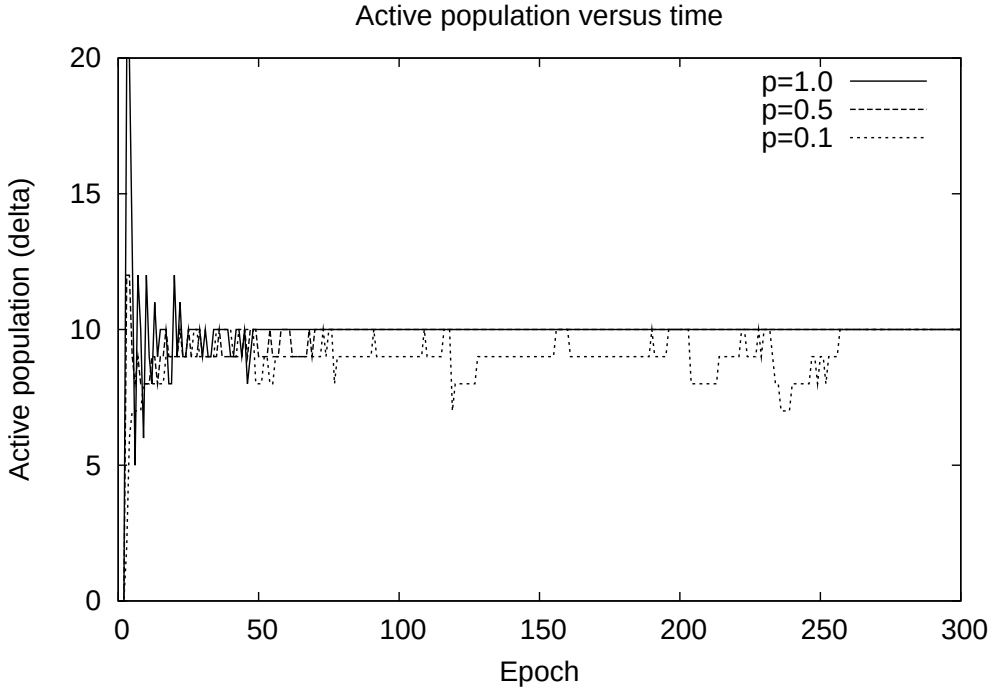


Figure 8.10: Development of R_1 over time for values of p_ω

Figure 8.10 illustrates the relationships between R_1 and time for different values of p_ω. The *suspended-searching probability* p_ω varies in the range $[0, 1]$, with one value of p_ω per trace. We see that for all $p_\omega > 0$ eventually $R_1 = n$, with higher values of p_ω reaching this condition more quickly. This illustrates that it is desirable to set p_ω as high as possible while still conforming to cell energy usage requirements as considered in section 8.3.6.2.

The initial state of the cell is that each node has a random initial local phase, so in addition to the probabilistic action of ADCP there is another source of instability at the start as the LISP mechanism evenly spaces the per-node synchronisation events throughout the system epochs. Each node which implements ADCP estimates the number of active neighbouring nodes sharing its cell by counting, within a given period of equal length to one system epoch, the number of synchronisation events fired by these neighbouring nodes. Note that although this method is used in this experiment, other methods to estimate population sizes are discussed in section 8.3.5.

If these neighbours' event firing times remained constant, relative to the start and finish times of this node's counting period, then each neighbour would be counted exactly once and hence an accurate estimation of the active population would be acquired. However, LISP induces individual nodes to move the timing of their synchronisation events. It follows that, within a given counting period, a node may observe a given neighbour firing its synchronisation event 0 times if the next event event is pushed back beyond the end of the counting period, or 2 times if the occurrence of the next instance is pulled forward to within the counting period. As *SUSPENDED* nodes do not generate synchronisation events, LISP must restabilise the cell every time ADCP transitions a node between the *active set*, Δ, and the *reserve set*, Λ.

As higher values of p_ω are expected to permit the cell population to stabilise at the required value more quickly, it is useful to consider the relationship between p_ω and the critical time t_c required for the cell to reach and maintain the condition $R_1 = n$. t_c is measured in *epochs*, which are independent of the underlying time unit in which the system epoch length is defined. Note that t_c may occur some time after the first occurrence of $R_1 = n$ if the cell takes some time to stablise, during which the active population size varies and is at times higher or lower than required.

Figure 8.11 plots p_ω against t_c to illustrate the relationship between *suspended-searching probability* and cell stabilisation time. Examining the smoothed trend we see that, in general, higher values of p_ω yield lower values of t_c, indicating that setting p_ω is desirable in

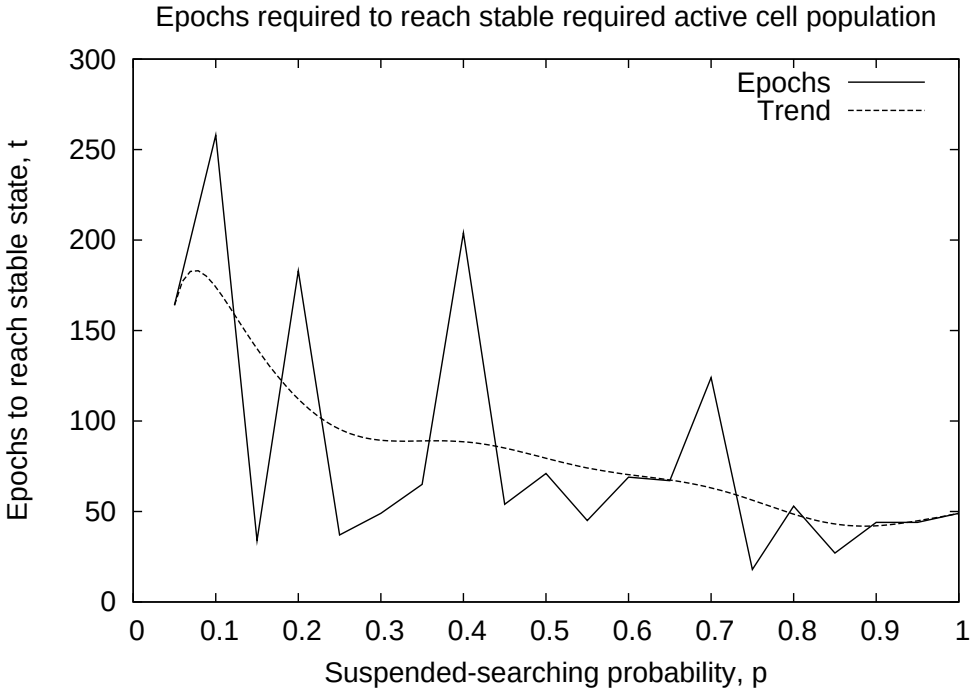


Figure 8.11: Epochs required to reach $R_1 = n$ stable state versus p_ω

systems for which rapid stabilisation is desirable. However, examining the unsmoothed raw datapoints we see that the relationship between p_ω and t_c is not simple. Where t_c appears particularly large for a given p_ω, it is usually the case that $R_1 \approx n$ such that it cycles between n and either $R_1 = n - 1$ or $R_1 = n + 1$ for numerous epochs before finally settling on the correct stable value of $R_1 = n$.

This is a consequence of complex interactions between independently acting nodes. The number of active nodes is sampled periodically, and is well-defined but unknown between these sampling points, but can change unpredictably owing to the probabilistic nature of the decision process which is not coordinated between nodes (see section 8.3.6). The probabilistic decision of any given node indirectly influences the decision of every other node for the remaining runtime of the network. This leads to a situation in which behaviour is deterministic but hard to predict, this property deriving partially from LISP on which the active neighbour estimation function is based.

As the number of nodes per cell grows, the influence of any individual decision is minimised, because the number of candidate nodes making a given decision approaches the expected value of the probability distribution function. However, when the active cell population is close to the desired population, and the sensornet cell does not contain a large number of nodes, the influence of an individual decision may be significant in the

short term.

Another factor to consider in assessing the performance of ADCP is its response to variations in total cell population, l. The value of l may vary across network cells in the initial state of the sensornet owing to a potentially unpredictable or inconsistent deployment method and operational environment. Furthermore, as nodes fail or are added to the sensornet, the value of l for any given cell may rise or fall, and ADCP should continue to work effectively. ADCP should maintain $R_1 = n$ if $l \geq n$, and should maintain $R_1 = l$ where $l < n$ and hence the prior condition cannot be achieved. The latter goal is defined such that, if necessarily $R_1 < n$, then we want R_1 to approach n as closely as possible.

A further set of experiments was implemented, similar to those described above for variable p_ω and fixed l, in which p_ω was fixed and l was variable. We set $p_\omega = 0.5$ as this is the central value of the defined range $[0, 1]$, and varied l in the interval $[1, 100]$ to span an order of magnitude greater, and an order of magnitude lesser, than the critical value of $l = n$. Figure 8.12 illustrates the time, measured in *epochs*, required to reach the stable target population as defined above for each value of l shown on a logarithmic scale. The dashed vertical line at $l = 10$ indicates the transition between the $R_1 = l$ stable condition for $l < n$, and the $R_1 = n$ stable condition for $l \geq n$.

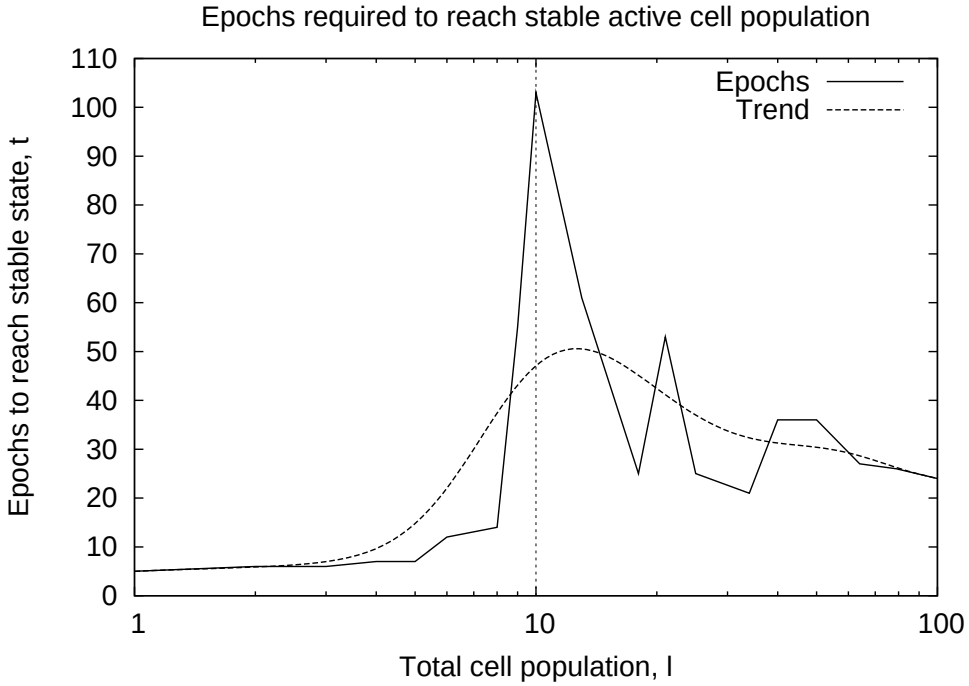


Figure 8.12: Epochs required to reach $R_1 = n$ stable state versus l

Firstly, consider the plot for $l < 10$ in figure 8.12. Observe that relatively few epochs

must pass until the $R_1 = l$ stable condition for $l < n$ condition is reached. This is because the cell is always *underpopulated*, as defined in section 8.3.6.2, such that any node entering the *SEARCHING* state is permitted to undergo the *SEARCHING→JOINING→ACTIVE* transition sequence; the cell is never *overpopulated* or *correctly populated* as defined in sections 8.3.6.1 and 8.3.6.3 respectively. As $l \to n$ a greater number of epochs are generally required as more nodes must probabilistically decide to join the *active set*, Δ.

Secondly, consider the plot for $l \geq 10$ in figure 8.12. Observe that the number of epochs which must pass until the $R_1 = n$ stable condition for $l \geq n$ is reached is generally higher than for the $l < n$ case, primarily because the cell may be categorised as any of *overpopulated*, *underpopulated* or *correctly populated*, as defined in sections 8.3.6.1, 8.3.6.2 and 8.3.6.3 respectively. It follows that the required behaviour is more complex, as the ADCP algorithm cannot simply allow any *SEARCHING* node to undergo the *SEARCHING→JOINING→ACTIVE* transition sequence; unless restricted, R_1 would grow until reaching the $R_1 = l$ upper bound and overshoot the $R_1 = n$ target.

For $l \geq 10$, observe that the plot labelled "Epochs" is somewhat jagged, indicating the relationship between l and t_c is not simple. It follows that it is non-trivial to predict t_c for a given l. This is largely a consequence of the initial stabilisation and calibration period of the underlying LISP algorithm as discussed in chapter 7. The ADCP mechanism also forms a feedback loop; a given node influences its neighbours, which in turn exert influence on the given node in the future. A characteristic typical of feedback-driven systems is that their behaviour can be entirely deterministic and yet difficult to predict [77], and the relationship between system size and stabilisation time can be nonlinear.

ADCP features probabilistic state transitions. This removes the requirement for heavyweight coordination between nodes, but also implies that any given probabilistic transition may or may not occur over any given period. Owing to the nature of ADCP, the active population generally gets close to the target value quickly, but then may require a considerable (and difficult to predict) number of iterations to settle on the final value.

However, the plot labelled "Trend" indicates that the general trend is for t_c to decline with increasing l, such that larger total cell populations, l, generally require less time to establish the desired active population, n. This is explained simply by larger total cell populations having a greater number of suitable candidate nodes which are available to join the *active set*, Δ, but are currently in the *reserve set*, Λ. As $l \to \infty$ the number of candidate nodes in Λ which independently elect to undergo the *SEARCHING→JOINING*

transition becomes closer to the expected value for the relevant probability distribution function discussed in section 8.3.6. ADCP tends to function more effectively in this regard as l becomes larger, and hence the problem size becomes larger, which is particularly useful for very large sensornets in which protocol scalability is of particular concern.

8.3.11.2 Population management: Replacing a failed node

Section 8.3.11.1 considers the startup period for a sensornet, in which cells and their nodes transition from the initial deployment state to a functioning state in which all cells have the correct active population. This transitional period necessarily implies reaction to instability as the correct working conditions are established. We now consider scenarios in which a sensornet cell has undergone this initial transitional period, is currently in a correctly adjusted and functioning state, and then must react to a change in the cell population.

This experiment considers a single cell of l nodes, of which $n < l$ should be in the *active set*, Δ, at any given time. ADCP is implemented to manage the active cell population. As the experiment begins, the active population size is equal to the target stable population size such that $R_1 = n$, and the underlying LISP mechanism is in its stable equilibrium state (see section 7.2.3). The cell is stable at the outset, and remains so until after a delay of t_b after the experiment begins, at which point a single node is selected at random and transitions from the *ACTIVE* state to *INACTIVE*. We then observe the value of R_1 as a function of time until ADCP replaces the removed node, and the $R_1 = n$ condition is later regained at some time t_c.

The cell under consideration has total population $l = 20$ and target active population $n = 10$, leaving 10 further nodes as spares such that there is a non-empty reserve pool from which to replenish the active population when a node is removed. We set the *node activation coefficient* $\pi_\psi = 1$, and the *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set the *voluntary suspension probability* $p_\tau = 0$ as we do not wish to obscure the population stabilisation effect with the fair duty allocation effect which will be considered in section 8.3.11.5. We set the *suspended searching probability* $p_\omega = 0.5$ as this is the central value of the defined range $[0, 1]$.

Figure 8.13 illustrates the action of ADCP replacing a failed node. The cell is stable until epoch $t_a = 50$ at which point an *ACTIVE* node is removed from the cell. Members

of the *reserve set*, Λ, which happen to be in the *SEARCHING* state at this point and in subsequent epochs implicitly detect the missing node as they estimate $\delta = 9$ rather than the expected $\delta = 10$. During the subsequent 49 epochs the cell active population size R_1 varies between 9 and 10 as ADCP replaces the missing node; this variation is explained by inaccuracies in cell active population estimation (discussed in sections 8.3.5 and 8.3.11.1) taken as inputs to the ADCP probabilistic decision processes (discussed in section 8.3.6). Finally, the cell has reattained a stable active population of $R_1 = n = 10$ at time $t_c = 50 + 49 = 99$ epochs from the experiment start.

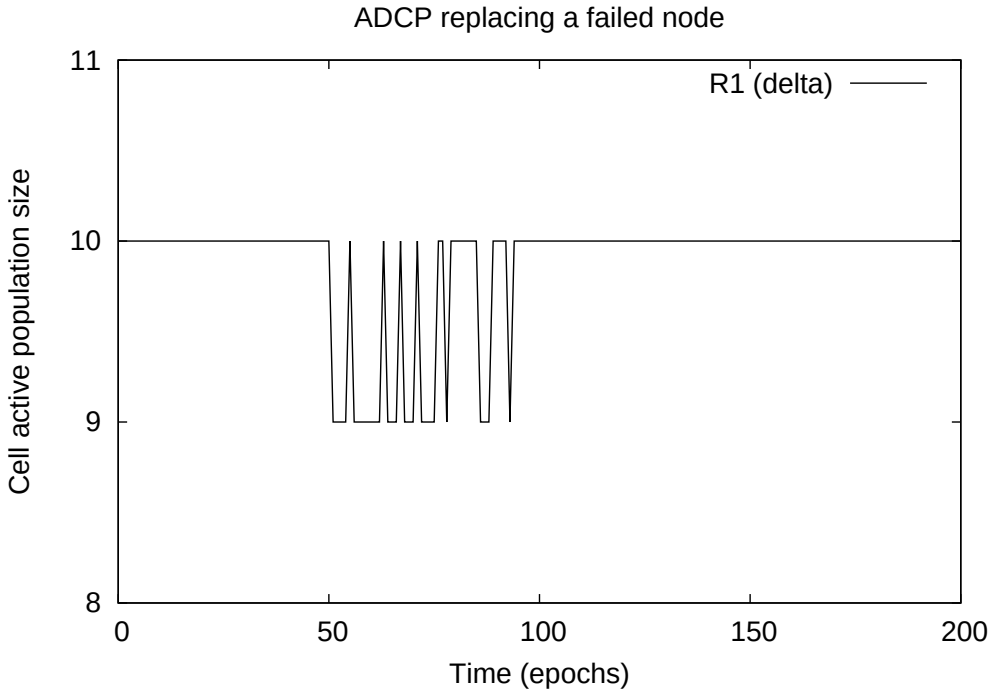


Figure 8.13: Progress of R_1 as a function of time as ADCP replaces a failed node

8.3.11.3 Population management: Removing a surplus node

In section 8.3.11.2 we consider the automatic replacement of a failed node. In this section we consider a similar scenario, in which an extra node is added to a cell in which the $R_1 = n$ condition already holds. This added node begins in the *ACTIVE* state to model scenarios in which, for example, nodes are added after sensornet deployment in response to a perceived shortfall to be remedied quickly while maintaining a low p_ω value, or in which some non-*ACTIVE* node malfunctions and incorrectly assumes the *ACTIVE* state.

If this extra node were in a non-*ACTIVE* state it would not affect the stable state; indeed, if it were in the *SUSPENDED* state it would implicitly be added to the set of

spares available to replace failed nodes. However, if the extra node is in the *ACTIVE* state, then $R_1 = n + 1$ and thus ADCP must transition one node from the *active set*, Δ, to the *reserve set*, Λ. The identity of the transitioned node is not significant; it may, or may not, be the new node recently added to the cell.

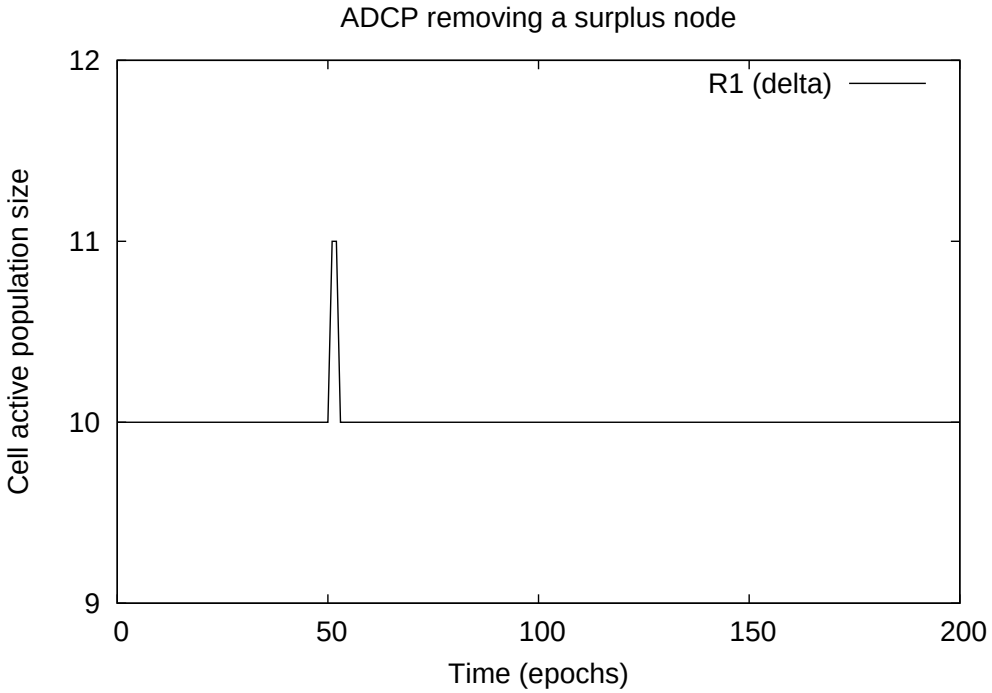


Figure 8.14: Progress of R_1 as a function of time as ADCP removes a surplus node

Figure 8.14 illustrates the action of ADCP removing a surplus node. The cell is stable until epoch $t_a = 50$ at which point an *ACTIVE* node is added to the cell. Members of the *active set*, Δ, implicitly detect the surplus node as they estimate $\delta = 11$ rather than the expected $\delta = 10$. During the subsequent 2 epochs, members of Δ apply the probabilistic decision process described in section 8.3.6, independently deciding whether to transition to the *reserve set*, Λ. Finally, the cell has reattained a stable active population of $\delta = n = 10$ at time $t_c = 50 + 2 = 52$ epochs from the experiment start.

t_c is shorter for removal of a surplus node than replacement of a failed node (discussed in section 8.3.11.2) as members of Δ are constantly estimating cell population sizes, and are able to react immediately to a perceived surplus. By contrast, members of Λ only estimate population sizes when in the *SEARCHING* state, and their reaction is non-immediate; they must follow the *SEARCHING*→*JOINING*→*ACTIVE* transition sequence, spanning a duration of $1-2$ epochs, during which other nodes may independently undergo transition.

8.3.11.4 Proportion of time spent by nodes in ADCP states

The primary purpose of ADCP is to manage the population of the *active set*, Δ, such that the size of the active population is the correct value of $\delta = n$; this allows other protocols and application software which assumes a certain cell population size to function correctly. A secondary purpose, which follows from the primary, is to reduce energy consumption by having those nodes in the *reserve set*, Λ, only periodically monitor the cell to determine whether or not to transition to the *ACTIVE* state; when not monitoring the cell for radio activity, a node can switch its radio module into an low-power state to conserve energy.

In section 8.3.11.1 we see that higher values of the *suspended-searching probability*, p_ω, tend to allow cells to attain the desired $\delta = n$ condition more quickly. However, increasing p_ω induces nodes in Λ to spend more time in the *SEARCHING* state, and hence reduces the opportunity to spend time with radio modules switched into low-power states for energy conservation. It is clear that some tradeoff must be found to balance the competing demands of responsiveness and efficiency. For example, the sensornet designer may select the highest value of p_ω allowed by the energy budget.

In this section we consider metric R_2, the proportion of total network runtime in which nodes are in the *SUSPENDED* state, as defined in section 8.3.10. While in the *SUSPENDED* state, nodes are not required to participate in intra- or inter-cellular communications activity, and hence are permitted to switch off energy-hungry wireless communications modules. R_2 therefore measures the extent to which energy saving is possible as a consequence of ADCP placing currently unused nodes into a long-term reserve pool. The cell under experimental consideration has target active population $n = 10$ and total population $l = 100$, such that we expect the *reserve set*, Λ, to contain 90 nodes. All nodes begin in the *SUSPENDED* state corresponding to a *cold start* scenario as explored in section 8.3.11.1.

We set the *node activation coefficient* $\pi_\psi = 1$, and the *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set the *voluntary suspension probability* $p_\tau = 0$ as we do not wish to obscure the population stabilisation effect with the fair duty allocation effect which will be considered in section 8.3.11.5. We set the *suspended searching probability* $p_\omega = 0.1$ as this is within the defined range $[0, 1]$ and allows nodes in Λ to spend the majority of time in the *SUSPENDED* state, yielding significant opportunity for energy conservation.

Figure 8.15 displays the average proportion of cumulative runtime spent by nodes

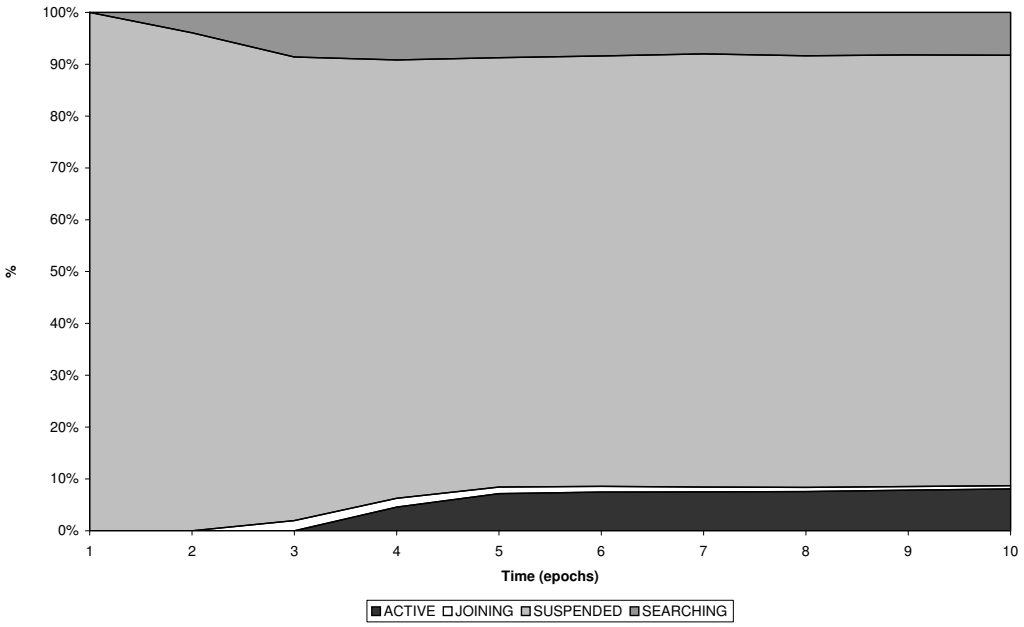


Figure 8.15: Average percentage of total time in ADCP states for first 10 epochs

during the first 10 system epochs as an area plot. The areas of this plot correspond to the *ACTIVE*, *JOINING*, *SUSPENDED*, and *SEARCHING* states respectively in ascending vertical order. Within a few epochs from the initial state we observe that the relative proportion of cumulative time spent in each state approaches a stable value. Figure 8.16 shows the average proportion of cumulative runtime of the same system, but for the first 100 system epochs. This shows that, in the longer term, the proportions remain steady as ADCP maintains subpopulations of stable size for each of the the ADCP states.

Considering the relative proportions of allocated time shown in figures 8.15 and 8.16 qualitatively, the *SUSPENDED* state dominates the allocated time, as measured by R_2. This is as expected and is the desired outcome; for the majority of total system runtime, nodes are permitted to take the *SUSPENDED* state in which energy-hungry wireless communications modules can be switched off.

The time allocated to the *JOINING* state is initially small, becomes smaller as the cell population stabilises and hence fewer nodes must undergo the *SUSPENDED*→*JOINING*→*ACTIVE* transition sequence, and quickly becomes insignificant. The time allocated to *ACTIVE* and *SEARCHING* states is significant, though much smaller than that allocated to *SUSPENDED*, showing that a proportion of node time is always allocated to *ACTIVE* so that communications activity can occur, and a proportion is always allocated to *SEARCHING* to maintain the size of the *active set*.

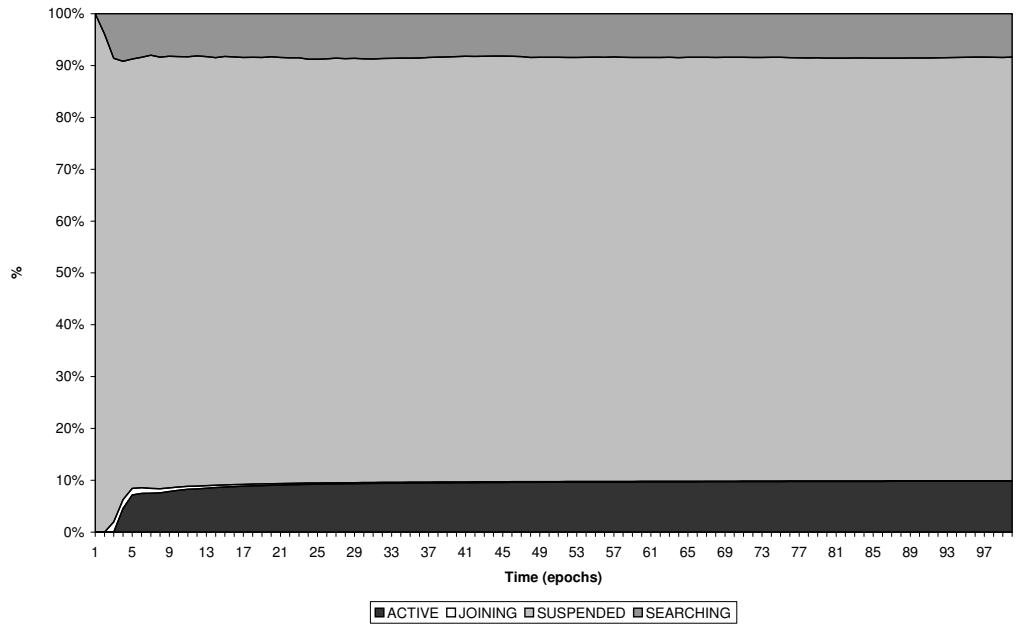


Figure 8.16: Average percentage of total time in ADCP states for first 100 epochs

State	Experiment	Predicted	Error (absolute)
ACTIVE	0.0996	0.1000	0.0004
JOINING	0.0001	0.0001	0.0000
SUSPENDED	0.8179	0.8100	0.0079
SEARCHING	0.0824	0.0900	0.0076

Table 8.6: Average proportion of total time in ADCP states for first 500 epochs

Now consider the numerical results for the proportions of allocated time for ADCP states from the same experiment, continued to 500 epochs to allow further stabilisation, and compare the values predicted by the formulæ of section 8.3.6. Values are given in table 8.6 to 4 decimal places. It can be seen that the proportion of time spent by nodes in the *SUSPENDED* state, R_2, is very close to the predicted value. The wireless communication module for a node consumes energy at average rate x when switched on, and at lower rate y when switched off, under ADCP the average energy consumption rate is given by $x(1-R_2)+yR_2 = x-(x+y)R_2 < x$. If R_2 is significant, which is true for this experiment as $R_2 = 0.8179$, and $x-y$ is significant, which depends on the specification of the wireless communications modules, the consequent reduction in average energy consumption rate is also significant.

8.3.11.5 Fairness and equality

In section 8.3.11.4 a secondary purpose of ADCP is to enable reductions in energy consumption by restricting the proportion of time nodes in the *reserve set*, Λ, spend in the

SEARCHING state; the *SUSPENDED* state is favoured as it allows nodes to switch off energy-hungry wireless communications modules. A tertiary purpose of ADCP is to share the energy burden of network participation equally and fairly throughout the cell population. The *active set*, Δ, is a subset of the total cell population. Rather than assign a fixed subset of the cell population to Δ, we want nodes to transition between Δ and Λ as described in section 8.3.6.3 such that all nodes spend an equal amount of time in Δ in the long term. Metric R_3 measures the standard deviation across the proportion of time nodes spend *ACTIVE*; lower values of R_3 indicate a more equal burden sharing policy.

Some similarity exists between this goal and that of CDAP, as described in section 8.2, but there are significant differences as different problems are addressed. Firstly, CDAP and ADCP operate over different timescales, as discussed in section 7.1.1. Secondly, and more importantly, CDAP constructs a cyclical and predictable schedule with transitions occurring at carefully controlled times, to enable node spatial density management and intra-cellular packet routing. By contrast, the long-term duty cycling implemented under ADCP does not coordinate the time of state transitions between sets of two or more nodes, and the pattern of duty periods is unpredictable for individual nodes (though statistically predictable at the cell level).

This experiment considers a cell of total population $l = 20$ and target active population $n = 10$, such that there is scope for nodes to evenly divide time between Δ and Λ. Nodes are ideally *ACTIVE* for 50% of network runtime such that $R_3 = 0$. We set the *node activation coefficient* $\pi_\psi = 1$, and the *node suspension coefficient*, $\pi_\chi = 1$, as we do not wish to artificially reduce the rate at which nodes enter or leave the *active set*, Δ. We set the *suspended searching probability*, $p_\omega = 0.5$, as this is the central value of the defined range $[0, 1]$. The voluntary suspension decision is taken independently at each *ACTIVE* node by the probabilistic method described in section 8.3.6.2. There is no restriction on newly suspended members of Γ being immediately reselected for transition back to Δ. The *voluntary suspension probability*, p_τ, is varied in $[0, 1]$ and the resulting value of R_3 measured; we focus on values of p_τ closer to 0 to prevent excessive churn.

Consider the value of R_3 as a function of time, t, where $p_\tau > 0$. Shortly after the network begins operation the $\delta = n$ goal is attained as n of l nodes are selected as *ACTIVE*, and the remaining $l - n$ are not. At this stage, the n selected nodes will have been *ACTIVE* for a large proportion of running time, and the $l - n$ other nodes will have been *ACTIVE* for only a small proportion; the standard deviation (R_3) of these *ACTIVE*

proportions will be relatively large, of the same order of magnitude as the mean.

However, if $p_\tau > 0$, as $t \to \infty$ nodes will occasionally be shuffled between Δ and Λ. All l nodes are eventually selected as *ACTIVE* for approximately equal proportions of total time, so R_3 eventually becomes small. Larger values of p_τ increase the shuffling rate such that R_3 declines more quickly.

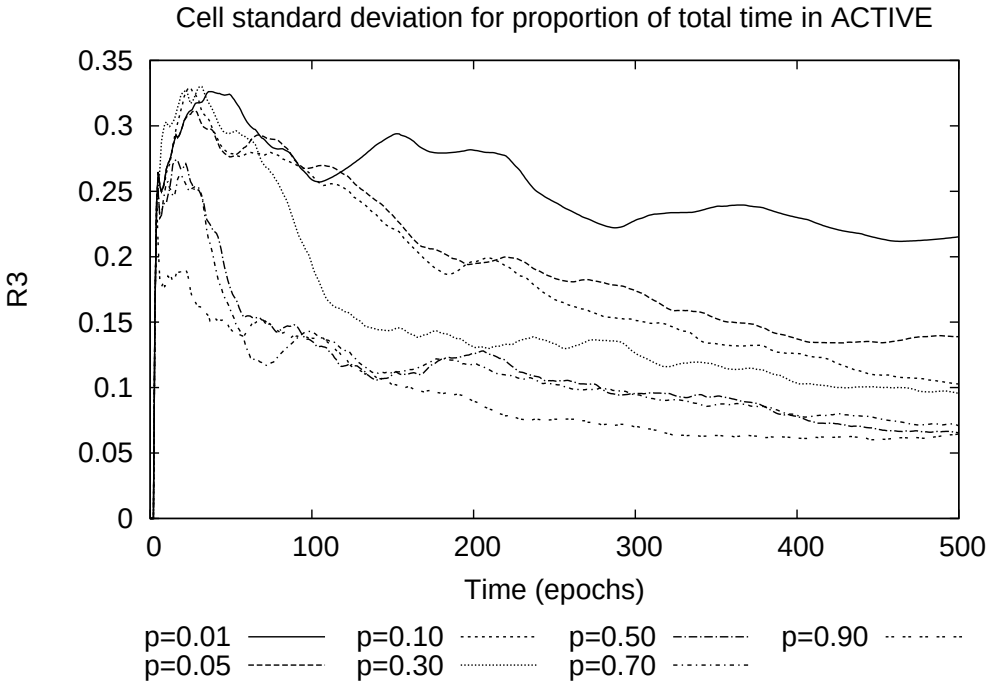


Figure 8.17: Standard deviation of proportion of time in *ACTIVE* versus time

Figure 8.17 illustrates the decline of R_3 as a function of time, t, measured in epochs, for differing values of p_τ in an otherwise unchanged cell. As R_3 declines, this represents an increasingly fair distribution of burden between all l nodes in the cell. It can be seen that higher values of p_τ lead to a sharper decline in R_3 and hence a faster equalisation of burden across the cell population. However, network designers must consider also the impact of excessive node churn when ADCP replaces voluntarily suspended nodes by the method discussed in section 8.3.11.2. For long-running networks it may be desirable to select low p_τ values; this minimises churn, and an approximately equal sharing of burden will eventually be reached during the lengthy network lifetime.

Figure 8.18 illustrates R_3 for differing values of p_τ after a fixed duration from network startup of t epochs, each of 10 seconds. As $t \to \infty$ ADCP manages cell populations such that $R_3 \to 0$, but some sensornets may have finite lifetime so the $R_3 \approx 0$ condition is never reached. Sampling R_3 at time t for a sensornet of expected lifetime t measures the overall

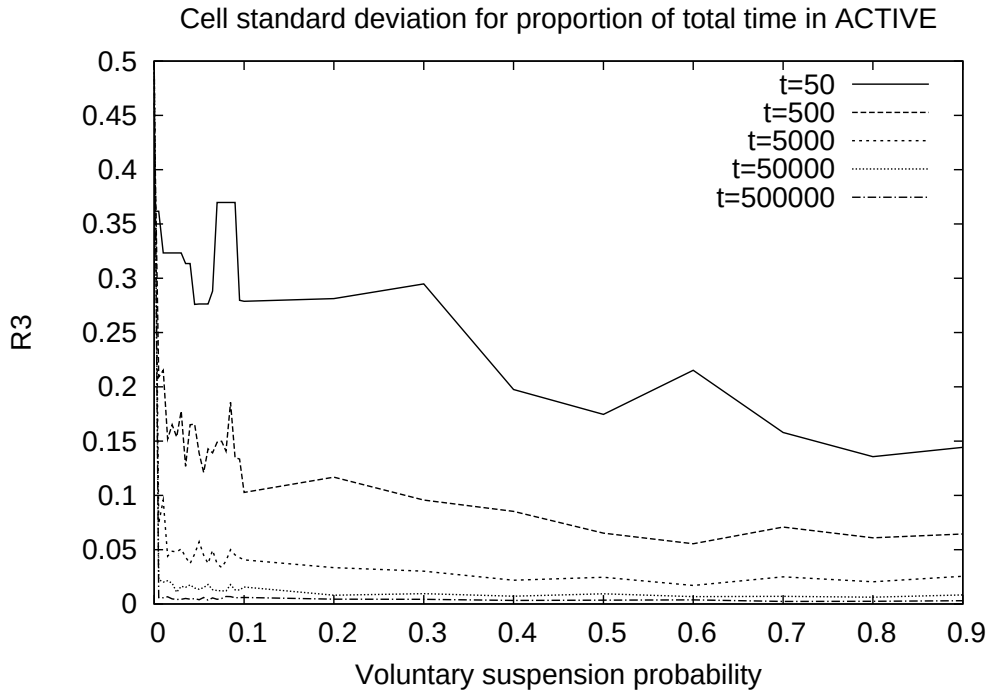


Figure 8.18: Standard deviation of proportion of time in $ACTIVE$ over different timescales

success of ADCP at fairly distributing burden in this situation.

Values of t were selected spanning 5 orders of magnitude from $t = 5 \times 10^2$ to $t = 5 \times 10^6$ epochs. As $t \to \infty$ the selected value of voluntarily suspension probability, p_τ, becomes less important; there is less variance between values of R_3 associated with differing p_τ, and all such values are closer to the ideal value of $R_3 = 0$. This is expected behaviour.

The longer the network runs, the greater the occurrence of any given node being randomly selected to be swapped between Δ and Λ, and hence the smaller the standard deviation between nodes for the proportion of time spent in Δ as measured by R_3; p_τ controls the rate at which nodes transition between Δ and Λ, not the identity of the nodes undergoing this transition. Over long periods the rate becomes irrelevant, provided it is non-zero, as all nodes will eventually spend roughly the same amount of time in Δ; the value of p_τ merely influences the rate at which equality is approached, rather than influencing the eventual result. It follows that selecting an appropriate p_τ value is of greater importance in networks of shorter lifetime, for which a greater value of p_τ may be favoured to obtain the desired level of fairness within that lifetime.

8.4 Summary

The *Cyclic Duty Allocation Protocol* (CDAP) was defined in section 8.2 and extended to perform dynamic state management of mote subsystems in section 8.2.3. The protocol manages a unicellular sensornet such that exactly one mote is available for communication duties at any given time, and that the corresponding energy cost is shared equally by all participating motes. Whereas LISP provides a primitive with which any number of contention and redundancy management mechanisms could be constructed, CDAP provides a specific mechanism with readily understandable properties.

Theoretical estimates of protocol performance and energy efficiency are defined in sections 8.2.1 and 8.2.3, against which empirical measurements are compared in section 8.2.5. It is shown that the protocol achieves its aims, constructing a near-optimal duty schedule with significantly improved energy efficiency.

It would be interesting to assess the efficacy of CDAP in other low-power wireless networking environments, as it is independent of any specific hardware platform, application software or network configuration. Additional energy efficiency improvements are possible if the assumption that every cell must always have at least one active node can be relaxed safely, for example where the sensornet designer can define periods during which subsections of the sensornet are not required.

The *Active Duty Control Protocol* (ADCP) was defined in section 8.3. The protocol manages nodes within a unicellular sensornet such that at any time there are a specific number of nodes actively engaged in network behaviour, with the remainder of the cell population held in a reserve pool. Nodes are moved between the active pool and reserve pool as and when this is necessary to restore the active population to the target population. Whereas CDAP manages network infrastructure in the medium term to meet application requirements, ADCP ensures CDAP has the necessary infrastructure resources to fulfil these requirements in the long term.

Probabilistic methods are employed to achieve rapid convergence on the target population size with low overheads in computation, storage and energy. Methods for calculating ideal probabilities for nondeterministic state transitions are provided in section 8.3.6. Empirical evaluation in section 8.3.11 demonstrates that the ADCP mechanism is efficient and reliable in maintaining an active cell population of the correct size while sharing the duty burden between the entire cell population.

Chapter 9

Intercellular synchronisation

Chapters 7 and 8 describe coordination protocols with which to coordinate activity within a fully-connected sensornet. Although it is reasonable in small sensornets to assume that every node can communicate with every other node, this assumption begins to break down as the node population increases. In larger sensornets there may exist one or more nodes which are not within communication range of one or more other nodes. Asymmetrical node-node links may exist owing to the characteristics of the environment [102]. Contention for the shared wireless medium increases as more nodes simultaneously need to transmit data. The cost and settling time of distributed protocols may increase with node population (see sections 7.2.10.2, 7.2.10.3, 8.3.11.1 and 9.2.16).

Although ADCP (see section 8.3) can maintain a small working set from a large reserve pool of nodes, this does not solve the problem where the distributed application actually requires that a large number of nodes be active simultaneously, or requires a physical distribution of nodes such that the network is not fully connected. A more fundamental approach is required. One strategy is to apply the *divide-and-conquer* approach. If we cannot address the whole problem in one step, then we progressively divide the problem into smaller subproblems. Eventually we arrive at subproblems that are sufficiently small that we can address these directly.

We apply this reductionist approach by dividing the sensornet into a number of cells. This division occurs only logically; no physical changes are implemented or necessary. We can implement the protocols such as those described in sections 7.2, 8.2 and 8.3 *within cells* without modification, provided that we have some mechanism to coordinate activity *between cells*. The *Dynamic Cellular Arbitration Protocol* (DCAP) implements the required intercellular coordination.

9.1 Cellular sensornets

Before defining any intercellular coordination protocol, we must first consider the logical cellular structure and how this relates to the physical composition of the sensornet. This is necessary as it exerts a strong influence on the interactions that are possible within and between sensornet cells.

9.1.1 Physical distribution of nodes

The physical region inhabited by the sensornet is divided into a number of cells, as described in section 2.2.7. This division of physical space defines the physical boundaries for the region occupied by each network cell. We assume that the physical boundaries of cells are not allowed to overlap, and that each node is located within exactly one cell at any given time. It is possible, however, for broadcasts in one cell to spill out into a neighbouring cell unless physical barriers exist around the cell perimeter. In this chapter we are not concerned with the physical cellular structure; we need only know which cells are *adjacent*; by this we mean the set of potentially interacting pairs of cells.

In most cases *logical adjacency* is implied by *physical adjacency* owing to the characteristics of wireless broadcast communications. It is possible for more complex sensornets to include one or more direct wired links between distance nodes, effectively giving *logical adjacency* to a pair of cells which do not share *physical adjacency*. However, we do not consider this situation explicitly as it has no bearing on our analysis.

We are also not concerned with the physical distribution of nodes within cells. Depending on the design of the network, and the method of node deployment, this physical distribution may be uniform or nonuniform. In section 9.2.2 we discuss the assumption that each cell should contain the same number of actively participating nodes, or at least some integral multiple of a shared base population.

9.1.2 Allocating nodes to cells

At any given time a cell contains zero or more nodes. Each node knows to which cell it belongs at any given time, but does not need to know the other members, if any, of this cell. It is possible for a cell to contain zero nodes, and indeed this is likely in large sensornets; there may exist regions of the total physical deployment space which are not interesting to the network operator, or regions may experience very high rates of node

failure, or regions may simply receive no nodes owing to unpredictable and uneven deployment methods. Nevertheless, any such empty cells are perfectly valid cells, unless the network designer specifically defines otherwise. It is possible, for example, that a mobile node could temporarily populate a previously empty cell. Each empty cell represents a network void.

Each node resides in exactly one cell. However, it is possible for the communication range of a node to cover a physical region extending beyond its own cell. This is essential for intercellular communication; it is possible only if there exists at least one pair of nodes, split between the cells, within mutual communication range. Nodes can modulate the broadcast power of their wireless communications modules to exert influence on the physical region within which pairwise exchange is feasible [157], but this is beyond the scope of this thesis and is not required for the methods discussed herein.

9.1.3 Logical structure within cells

We assume that all nodes within a cell are equal peers, such that all cell members can support an equal share of the processing and storage burden for the distributed application. This implies either a homogeneous cell population, or a heterogeneous population in which all members have at least the resources required to support an equal share of the application workload. The failure of a single node may reduce the capability of the cell, but should not cause the entire cell to fail.

This is in contrast to sensornet designs featuring one highly-resourced dedicated *clusterhead* and numerous lower-resourced subordinates per cluster [52]. Typically, the *clusterhead* node has much greater resources such as energy, processing power, storage capacity, network bandwidth and wireless broadcast power; the other *non-clusterhead* nodes are subservient to the *clusterhead* node. In some designs all network traffic within a cell, and between cells, is routed through the clusterhead. In other designs, intracellular traffic can pass directly between cell members, but intercellular traffic must pass through the clusterhead.

Clusterhead failure renders the entire cluster or cell inoperable and invisible to the remainder of the network. It may be difficult to arrange for the physical region of each cell to contain exactly one central highly-resourced clusterhead node if deployment is not fully predictable and controllable. In heterogeneous cells, subordinate nodes may not be suitable for promotion to clusterhead as they do not have sufficient resources to perform

the role. It follows that homogeneous cells without clusterheads are more resilient to unpredictable node hardware failures.

We assume that there is full connectivity between all nodes in a cell. If this is not possible for a given cellular configuration, perhaps as a consequence of physical obstacles or voids, one possible solution is to reconfigure the network with smaller cells until such time as full connectivity is reestablished. We also assume that for each pair of adjacent cells there exists a pair of nodes, one per cell, which is able to communicate. This is essential for adjacent cells to exchange application and network management data.

A weaker form of complete connectivity is established if the graph of cells is connected, but without the requirement that each cell is connected to all neighbouring cells. Although there is no reason in principle why the methods discussed in chapter 9 would not work in a network of this class, we do not give further consideration to this situation. It is also obvious that in a disjoint network of cells, in which one or more pairs of cells are not connected either directly or indirectly, it is not possible to guarantee that any network protocol can coordinate activity between all cells.

9.1.4 Hierarchical structures of cells

Having established a logical cellular structure based on physical regions, we consider how this structure is used. It is obvious that the network designer may find any number of applications for this structure, depending on the requirements of the distributed application. However, a more general observation is that it allows network protocols to be implemented and to make decisions at the level of the network cells, rather than that of the individual node.

Consider a network routing protocol of the type considered in chapters 3 and 4. Packets must be routed within the network via one or more *hops* between pairs of network entities. Usually these entities are physical network nodes. However, in section 9.1.3 we observe that within a network cell all nodes have a similar view of the physical region and wireless communications medium; any cell member is equally capable of handling a given packet exchange.

Individual sensornet nodes generally do not have globally unique identifiers [216]. Applications use geographical location to label sensor data and endpoints of data flows. Given that nodes sharing a cell are physically close, we can route packets between *cells* rather than *nodes* [34]. As network management instructions, raw sensor data and processed

information flow through the network they can be directed between cells by any nodes which happen to be assigned to these cells. The decision as to which of the cell members is responsible for handling a given packet at a given time can be managed by a duty allocation protocol such as CDAP (see section 8.2), but this is beyond the scope of DCAP.

Unless each cell is a *singleton cell* containing exactly one node, the logical network is smaller than the physical network. It follows that protocols which become less reliable as the network size increases will benefit from addressing the network at the level of the logical cell rather than the physical node. As fewer physical nodes are involved, it is possible to achieve corresponding decreases in energy consumption and network congestion among other factors.

(a) Routing between individual nodes

(b) Routing between cells

Figure 9.1: Routing between individual nodes and between cells in cellular sensornets

For example, figure 9.1 illustrates a sensornet of 324 nodes arranged into an 18-by-18 square grid, overlaid by a hexagonal grid of 22 cells. The application must send a packet from the black node at the lower left corner to the black node at the upper right corner. Grey nodes are intermediary nodes along the multi-hop delivery path, indicated by the sequence of arrows, selected by some geography-aware routing protocol which selects next-hop nodes to approximate a straight line between source and destination.

In figure 9.1(a) the cellular structure is inactive and any node may handle the packet, whereas in figure 9.1(b) the cellular structure is active and exactly one node per cell is available to handle the packet at any time. If packets are routed between cells rather than between individual nodes, a much shorter path of fewer node-node hops is achieved.

This confers significant advantages for the application; the end-to-end delivery latency is reduced, there are fewer opportunities for packet corruption or loss, and fewer nodes need expend energy in wireless broadcast and reception.

9.2 Intercellular timing coordination

Correct behaviour of distributed applications may rely on synchronisation in the time domain. For example, if a data packet is to be exchanged between two cells, it is important that both the sending node and the receiving node are active simultaneously and do not become inactive during the exchange. The *Dynamic Cellular Accord Protocol* (DCAP) provides a mechanism to coordinate synchronisation behaviour across a cellular network in which each cell executes an instance of the *Lightweight Improved Synchronisation Primitive* (LISP), as defined in section 7.2. This implicitly coordinates, across all cells, the time-sensitive behaviour of any other protocols or application based on LISP-supplied synchronisation.

9.2.1 DCAP: Dynamic Cellular Accord Protocol

DCAP works in tandem with LISP, as defined in section 7.2. Whereas LISP achieves *desynchronisation* within a network cell, DCAP achieves *synchronisation* between network cells. *Synchrony* can be considered the complement of *desynchrony*. In this section we discuss how DCAP utilises the LISP *sync pulse* transmissions to align the relative phase of equivalent nodes located in adjacent cells to achieve the desired *synchronised equilibrium* condition.

9.2.2 Cell populations

Unlike the approach by Lucarelli and Wang [190], which implements *synchronisation* within an arbitrary partially connected graph of nodes, DCAP works in conjunction with the natural hierarchical structure of a cellular or clustered sensornet. Greater priority is given to local coordination within cells, allowing effective collaborative working to continue within a cell regardless of temporary disruption within neighbouring cells. Exploiting this hierarchy also reduces the effective size of the problem, as there are generally fewer non-empty cells than nodes, enabling the desired level of coordination to be reached more quickly.

We assume that each constituent node of the network belongs to exactly one non-overlapping cell at any given time, and that the node is aware of the identity of its cell. It is not necessary for each cell to be uniquely identified throughout the network; it is sufficient for each cell to have a different identifier to each of its neighbouring adjacent cells, but it is not necessary to distinguish between specific neighbouring cells. For example, this condition might be achieved by some map-colouring algorithm which assigns to each cell an identifier selected from a small set of candidates which are reused multiple times across the entire network. The method underlying any such assignment mechanism is not significant to DCAP, and being beyond the scope of this protocol is not considered further.

We assume that each cell may have zero or more directly adjacent cells, such that broadcasts originating in a given cell are receivable by at least one node in each adjacent cell. Ideally, all nodes in a given cell can successfully exchange messages with all nodes in all adjacent cells, but this is not necessary. It is possible for a node to transition from one cell to another, provided that the communication assumptions stated above still hold under the new node-cell assignment.

The DCAP protocol functions by aligning equivalent LISP *synchronisation events*, originating in adjacent cells, in the time domain. To reach a stable solution it is necessary that the *synchronisation events* of a given cell can be overlaid on those originating in other cells such that, in the target stable *synchronised equilibrium* state, the time difference between a synchronisation event and its equivalents in the adjacent cells is sufficiently small as to be effectively zero within a small timing error margin. This equilibrium state is examined further in section 9.2.6.

If cells are allowed to contain any arbitrary non-uniform number of nodes, it is possible that no configuration exists in which each synchronisation event originating in a given cell can be paired with an equivalent in each adjacent cell with negligible time difference. We must therefore regulate the active population of non-empty cells. Empty cells have no impact on the functioning of the protocol and can be safely ignored. Non-active nodes, located in non-empty cells, which do not participate in LISP-style synchronisation can also be safely ignored.

The simplest policy is for each cell to contain the same number of active nodes, such that each synchronisation event originating in a given cell is paired with exactly one

synchronisation event in each adjacent cell in a simple one-to-one mapping.[1] This might be achieved with a protocol such as ADCP, defined in section 8.3, though any mechanism which achieves the same results would be equally acceptable.

9.2.3 Discriminating intra- and extra-cellular synchronisation pulses

DCAP utilises the synchronisation pulse transmissions implied by running a distinct instance of LISP within every cell. As it does not induce any transmissions of its own it is very lightweight, does not cause additional contention for the wireless medium, and does not consume additional energy. However, it cannot function adequately if nodes cannot determine whether a given observed synchronisation pulse transmission originates from within or from without the cell. Note that this is the *only* identification information that is required. It is not necessary for a node observing a sync pulse to identify the transmitting node, or the cell in which that node resides.

As DCAP does not need to extract much information from the observed sync pulse transmission, only whether it originates from within the same cell, it is possible to achieve this without significant overhead. We assume that identifiers are assigned to each cell such that no adjacent pair of cells is assigned the same identifier. This identifier could be globally unique for the cell, perhaps derived from its geographical location, but this is not necessary. Using fewer identifiers allows a shorter encoding of identity data in each sync pulse transmission. Allocating identifiers to cells is beyond the scope of DCAP but is addressed elsewhere in the literature [96].

Finding the minimal number of unique identifiers required is depends on the spatial configuration of the cells, and is an instance of the *map colouring problem* [323]. By the Four Colour Theorem [14] any planar map requires only 4 unique cell identifiers; maps covering spherical or cylindrical surfaces are equivalent to planar surfaces. The Heawood

[1] Although this is the simplest policy, it is not necessary for each cell to contain the same number of active nodes. It is sufficient for each cell to be populated with some cell-specific integral multiple of a shared integral constant. Under this condition, the *frequency* at which synchronisation events occur in any given cell is a *harmonic* of a shared *fundamental frequency*. If each cell contains an exact integral multiple of c nodes, and each cell implements the LISP protocol with epoch e time units, the fundamental frequency is given by $f = {}^{c}/_{e}$. It follows that a stable configuration can be reached in which synchronisation events occurring at the fundamental frequency are paired between cells, though the identity of the transmitting nodes whose events are paired may change from epoch to epoch. However, in the discussion that follows we will assume that each cell contains the same number of active nodes.

Conjecture [122], proved by Ringel and Youngs [247], gives the minimal number of unique identifiers from the Euler number of most closed surfaces. For example, 7 unique identifiers are necessary for maps covering toroidal surfaces [323]. The only exceptions for which the Heawood Conjecture is not applicable are the Möbius strip and the Klein bottle, both shown by Franklin [95] to require 6 unique identifiers.

An ideal scheme would have each cell transmit on an identifier-specific frequency and have each cell listen to the frequencies associated with all of its adjacent neighbour cells. This would allow the minimal sync pulse transmission to contain no actual information; the synchronisation event time is implicitly conveyed by the transmission time, and the cell identity is implicitly conveyed by the transmission frequency. This would avoid the possibility of clashes between synchronisation transmissions of different cells, or confusion as to the transmitting cell identity.

However, this is not practical for sensornet motes which have only one transceiver and are therefore only able to listen on one frequency at any given time. An alternate solution is to encode the cell identifier into each sync pulse transmission, perhaps implemented as the smallest possible packet under the appropriate network stack with a payload of nothing more than the identifier. As per section 7.2.6 we label the time required for these minimal packets to traverse the network stack as κ.

As a given node observes a sync pulse transmission, it decides whether that transmission originates from within or without the cell. If the former, the information is passed to the LISP protocol defined in section 7.2. If the latter, the information is passed to the DCAP protocol described in this chapter. This identification and channelling is not strictly necessary for the correct functioning of DCAP, as the correct behaviour would be observed provided that nodes do not act on their own sync pulse transmissions, but it is necessary for the correct functioning of LISP.

9.2.4 Cellular network building blocks

Consider a multicellular network consisting of a set Γ of cells $C_1 \cdots C_n$. We assume any given cell is adjacent to one or more other cells, but it is not necessarily true that all cells are adjacent to all other cells. The set A contains adjacent cell pairs (C_x, C_y) where C_x and C_y are two *logically adjacent* cells as defined in section 9.1.3. The (C_x, C_y) tuple is commutatively equal to (C_y, C_x) as all sensornet nodes are equipped with transceivers and bidirectional message exchange between cells is possible. The reflexive (C_x, C_x) tuple is

disallowed as it is meaningless for a cell to be adjacent to itself.

It is necessary for nodes in a given cell to determine which observed synchronisation transmissions originate from an adjacent cell, as per section 9.2.3, but it is not necessary for nodes to determine whence these extracellular transmissions originate. The set D_x contains all cells that are logically adjacent to a given cell C_x. The membership of D_x is defined by taking the subset of tuples from A in which C_x features as exactly one entity, and taking the other entity from all such tuples. The value $d_x = |D_x|$ gives the number, though not the identity, of cells adjacent to C_x. The maximum number of cells which can be adjacent to a given cell is a.

Extending the LISP definitions in section 7.2.2, each cell $C_x \in \Gamma$ contains a set Σ_x of n active nodes $S_{x1} \cdots S_{xn}$. A separate instance of LISP operates in each cell. Each node $S_{xy} \in \Sigma_x$ executes a single instance of a periodic event V_{xy} exactly once per system epoch with period e. This occurs when the local timer of node S_{xy} reaches the condition $\phi_{xy} = \phi_{max}$, at which point a synchronisation pulse is broadcast and the local timer is reset to $\phi_{xy} = 0$. From a global viewpoint this occurs within some system epoch at time ψ_{xy}, although nodes have no concept of a global clock. The LISP mechanism defined in section 7.2 forces these periodic events V_{xy} to be equidistantly spaced in the time domain, each separated by a delay of $\frac{e}{n}$ time units or equivalently $\frac{\phi_{max}}{n}$ phase units.

9.2.5 Intercellular phase error

Each cell contains the same number of active nodes, n, each transmitting exactly one synchronisation pulse per system epoch of length e. As per the definition of LISP given in section 7.2, in each cell these are spaced evenly in time with an interpulse delay of $\frac{e}{n}$ time units, or equivalently $\frac{\phi_{max}}{n}$ phase units. As stated in section 9.2.3 only the synchronisation pulse transmissions originating from extracellular nodes, in addition to a node's own transmissions, are visible to the DCAP algorithm. A node can trivially measure the delay between any two observed sync pulse transmissions using its local timer.

Figure 9.2 illustrates the sequence of sync pulse transmissions in two adjacent cells over a duration of around two system epochs. Each cell has reached the *desynchronised equilibrium* state for its internal LISP instance, as defined in section 7.2.3. Both cells exhibit a similar sequence of periodic transmissions, equal in frequency but unequal in phase. Each synchronisation pulse transmission, and hence the node responsible for its transmission, is paired with its equivalent in the other cell. Note that this pairing is merely

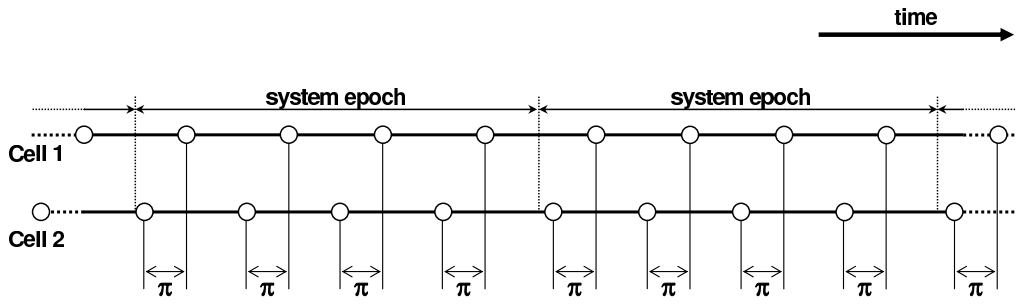


Figure 9.2: Intercellular phase error

from the viewpoint of DCAP, and does not imply any messages are passed between paired nodes, or indeed that any paired node is aware of the identity of its partner. Also note that any pairing is transitory; the identity of the nodes paired between a pair of cells may vary from epoch to epoch.

For each node α in *Cell 1* the paired node is selected as the node β in *Cell 2* for which the interpulse delay between the *Cell 1* and *Cell 2* pulses is minimal. As the sync pulses in both cells are equivalent in all aspects except phase, the interpulse delay is identical for all paired nodes. This shared delay is labelled as the *intercellular phase error*, π. DCAP aims to minimise π by shifting sync pulse transmission times in a similar, though different, manner to that employed by LISP in section 7.2.4.

Provided each cell is fully desynchronised, the interpulse delay between each node-node pair across the cellular boundary is identical and equal to π. It follows that measurement data from only one such pairing is sufficient for calculating the appropriate response, so there is no need for nodes within a cell to share measurement data. It also follows that if it is not possible for all nodes in *Cell 1* to be paired with a node in *Cell 2*, for example where there is not full connectivity between all nodes of the two adjacent cells, provided that one such pairing exists it will exert influence indirectly on all cell members. This allows DCAP to function effectively under non-ideal conditions, though of course it is advantageous for as many nodes to be paired between cells as possible.

9.2.6 Synchronised equilibrium

Section 7.2.3 discusses the equilibrium states for *synchronised equilibrium* and *desynchronised equilibrium*. To recap, n synchronisation events occur within each epoch of length e. Under *desynchronisation* these events are mutually repellent; in the *desynchronised equilibrium* state they are distributed evenly in the time domain, occurring at intervals

of $\frac{e}{n}$. Under *synchronisation* these events are mutually attractive; in the *synchronised equilibrium* state all n events occur simultaneously in each epoch, with a delay of e time units between each cluster of simultaneous events. It follows that $\pi = 0$ for a cell pair when the *synchronised equilibrium* state has been obtained.

Sections 9.2.7 to 9.2.10 discuss the application of *synchronisation* to the problem of coordinating activity between cells of a multicellular network. We define π_{xy} as the *intercellular phase error* between cells C_x and C_y, as measured from C_x, at some time during the active lifetime of a multicellular network. It is obvious $\pi_{xy} = -\pi_{yx}$ as they measure the same magnitude of phase error, but the ordering of equivalent synchronisation events between the cells is reversed. Although we utilise the signed measurements in section 9.2.9 to calculate the phase adjustments which nodes must undertake, the goal is to reduce the absolute value of these measurements toward zero. When the absolute value is approximately equal to zero, within some acceptable margin defined by inherent timing inaccuracy, the sign of a phase error measurement is irrelevant.

9.2.7 Synchronisation between adjacent cells

Figure 9.3 illustrates system convergence on the synchronised equilibrium condition for a network consisting of 5 cells, all mutually adjacent, and all containing 5 active nodes. Each cell is depicted by one of the concentric circles; note that this is merely to illustrate the relationship between equivalent nodes in cells, and the physical regions occupied by cells do not actually overlap. Each circle represents the progress of time within a single system epoch, where the angle from the x-axis represents the progress of time from 0 to ϕ_{max} phase units. Blobs positioned around each of the concentric circles represent the firing of a synchronisation event (see section 7.2.2) by one of the constituent nodes of the cell represented by the circle.

The leftmost element of figure 9.3 illustrates the initial condition for DCAP in which each cell is in the *desynchronised equilibrium* state (see section 7.2.3). There exists a non-zero *intercellular phase error* between pairs of cells (see section 9.2.5) such that $|\pi_{xy}| > 0$ between all pairs of cells in Γ.

The middle element of figure 9.3 illustrates the system of cells Γ after some time has passed. Within each set of *equivalent* synchronisation pulse transmissions, these have been driven closer in the time domain, but there still exists the condition $|\pi_{xy}| > 0$ between all pairs of cells in Γ.

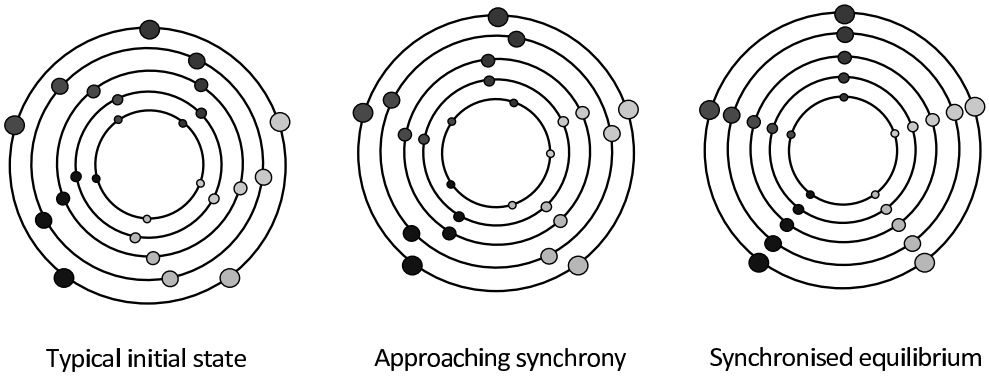


Figure 9.3: Progressive adjacent cell phase alignment

The rightmost element of figure 9.3 illustrates the synchronised equilibrium condition of the system of cells Γ after further time has passed. Within each set of *equivalent* synchronisation pulse transmissions, these have been driven sufficiently close in the time domain that the condition $|\pi_{xy}| \approx 0$ is maintained between all pairs of cells in Γ within the measurement error implied by timing inaccuracies such as clock granularity and the length κ of each transmission.

Although figure 9.3 illustrates a system in which all i cells $C_1 \cdots C_i \in \Gamma$ are mutually adjacent, such that all possible pairings are present as tuples in the adjacency set D, this is not necessary for the system to reach the synchronised equilibrium condition under DCAP. Provided the graph of pulse-coupled oscillators is connected [201], which is true if the graph of cells is connected, the system is guaranteed to converge on the synchronised equilibrium state [1] to an arbitrary level of precision, although this may take more epoch cycles than the fully connected case.

9.2.8 Timing measurements for equivalent events in adjacent cells

Consider a specific cell C_α with adjacent cells given by the set D_α. Each node $S_{\alpha y}$ observes both intracellular sync pulse transmissions, used by LISP, and extracellular sync pulse transmissions originating from all d_α adjacent cells $C_i \in D_\alpha$. Each node $S_{\alpha y}$ measures the time at which extracellular nodes, anonymous to $S_{\alpha y}$, in cells $C_\beta, C_\gamma, \cdots \in D_\alpha$ broadcast sync pulses, with the difference ρ measured in phase units in the interval $[-\frac{\phi_{max}}{2}, +\frac{\phi_{max}}{2}]$. The phase difference between the sync pulse transmission of $S_{\alpha y}$ and some node $S_{\beta z}$ where $z \in [1, n]$ is given by $\rho_{\alpha y \beta z}$. It is obvious that $\rho_{\alpha y \beta z} = -\rho_{\beta z \alpha y}$ as the two events $V_{\alpha y}$ and $V_{\beta z}$ are separated by a fixed distance in the time domain and have a fixed ordering, but

opposite sign depending on whether the relative measurement is taken from the viewpoint of node $S_{\alpha y}$ or from node $S_{\beta z}$.

Within the duration of a system epoch, a given node $S_{\alpha y}$ will have collected at most nd_α measurements of ρ, as there are d_α adjacent cells and each of these contains n active nodes. The set of all ρ values measured by node $S_{\alpha y}$ in epoch j is given by $R_{\alpha yj}$. The node can discard all ρ measurements at the end of each local epoch, which would place an upper bound on DCAP storage costs of na per node. This cost grows linearly in active cell population, n, and maximum adjacent cell count, a, which is a desirable property in the resource-constrained environment of sensornets.

However, for a node in a cell C_α with d_α adjacent cells, it is not necessary to store all nd_α measurements of ρ. Each node $S_{\alpha y}$ will coordinate its synchronisation event time with those of the *equivalent* nodes in adjacent cells in D_α; the measurements for all *non-equivalent* nodes in adjacent cells can be ignored. This reduces the DCAP storage overhead to d_α for a specific cell C_α, with an upper bound of a storage units in the worst case. The set of ρ values in $R_{\alpha yj}$ which originate at equivalent nodes in adjacent cells and are not discarded is given by $P_{\alpha yj}$ such that $|P_{\alpha yj}| = d_\alpha \leq a$.

The issue of discarding all ρ measurements except those d_α measurements from *equivalent* nodes in adjacent cells must now be addressed. As sync pulse transmissions are anonymous, and there is no coordination of LISP instances between cells, there is no explicit binding between equivalent nodes between two cells; the property of equivalence is an artefact of two or more nodes independently setting the firing of their LISP synchronisation events to occur at a similar time.

Recall from section 9.2.4 that the delay between consecutive synchronisation events within a cell is $\frac{e}{n}$ time units or equivalently $\frac{\phi_{max}}{n}$ phase units, when the LISP instance in that cell has reached the *desynchronised equilibrium* condition defined in section 7.2.3. We therefore have node $S_{\alpha y}$ consider all extracellular synchronisation pulse transmissions occurring within $\pm\frac{\phi_{max}}{2n}$ phase units of its own synchronisation event to originate at *equivalent* nodes in adjacent cells. This is similar to the definition of *active duty* periods assigned by CDAP as defined in section 8.2, and it follows that DCAP is automatically compatible with CDAP without any specific action on the part of the network designer. Nodes implicitly reject non-equivalent extracellular synchronisation pulse transmissions by simply ignoring any which do not occur within these $\pm\frac{\phi_{max}}{2n}$ phase unit boundaries.

If an adjacent cell should produce more than one synchronisation event within the

timing bounds outlined above, perhaps as an artefact of the adjacent cell being temporarily unstable and not yet having attained the desynchronised equilibrium state, node $S_{\alpha y}$ has no mechanism with which to reject one or more of the resulting supernumerary synchronisation pulse transmissions. In the short term this will prevent DCAP reaching its synchronised equilibrium condition. This is not problematic in the long term, however, as the unstable cell will quickly become stable in subsequent epochs (see section 7.2.4) at which point this problematic condition will disappear.

9.2.9 Calculating influence of adjacent cells

Section 9.2.8 defines the activity undertaken by each node to obtain the set $P_{\alpha yj}$ of ρ values obtained by node S_y in cell C_α in epoch j, which correspond to the d_α equivalent synchronisation events in cells immediately adjacent to C_α. Recall from section 7.2 that the LISP variants amend the local phase ϕ_i of each participating node S_i when the *successor phase neighbour* event is observed by S_i at $\phi_{i\gamma}$. From section 9.2.8 we know that, by this point, all *equivalent* synchronisation events in adjacent cells will have completed. It would be possible for nodes to adjust their local phase immediately upon observing an equivalent extracellular synchronisation transmission, but this would adversely affect the correct functioning of LISP; instead, the local phase change induced by DCAP is implemented simultaneously with that induced by LISP.

In section 7.2.4 we define that LISP calculates the value θ_i for each node S_i as the midpoint of the *predecessor* and *successor* phase neighbour synchronisation events. We now calculate ι_i as the midpoint of the *equivalent* synchronisation events in adjacent cells for node S_i, where i is shorthand for the unique identification of node S_y in cell C_α in epoch j. We can simply define $\iota_i = \text{avg}(P_{\alpha yj})$ as each node measures its ρ values relative to the firing time of its own synchronisation event, as specified in section 9.2.8.

LISP uses the feedback parameter $f_\alpha \in (0, 1]$ to specify the proportion of perceived intracellular phase error to be fed back into the system at each epoch, in order to manage the tradeoff between responsiveness and stability. DCAP uses the feedback parameter $f_\beta \in (0, 1]$ for a similar purpose. When the DCAP-driven phase adjustment is applied to a given node each node moves the timing of its own synchronisation event closer to that of the equivalent synchronisation events in adjacent nodes. Kuramoto [1] proved that such systems are guaranteed to converge on the desired synchronised equilibrium state to an arbitrary specified level of precision within finite time.

Algorithm 6 defines the DCAP ι_i calculation and local phase adjustment at each node $S_i \in \Sigma$. We assume that each node implements LISP, as defined in section 7.2. Variables not defined within the algorithm itself take the standard meanings used elsewhere in this document.

Algorithm 6 : DCAP at node S_i (node S_y in cell C_α in epoch j)

Require: Buffer of size a, $P_{\alpha yj} = \varnothing$
Require: Intercellular feedback parameter, f_β
Require: Cell population size, n
1: **while** monitoring local phase ϕ_i increasing over time **do**
2: **if** $\phi_i \geq -\frac{\phi_{max}}{2n} \wedge \phi_i \leq +\frac{\phi_{max}}{2n}$ **then**
3: **if** extracellular synchronisation pulse is heard **then**
4: $P'_{\alpha yj} \Leftarrow P_{\alpha yj} \cup \{\phi_i\}$
5: **end if**
6: **end if**
7: **if** $\phi_i = \phi_{max}$ **then**
8: $\iota_i \Leftarrow \text{avg}(P_{\alpha yj})$
9: $\phi'_i \Leftarrow \phi_i + f_\beta \iota_i$
10: $P'_{\alpha yj} \Leftarrow \varnothing$
11: **end if**
12: **end while**

Algorithm 6 describes the DCAP adjustment of node local phase ϕ_i in step 9. Similarly, LISP adjusts ϕ_i in step 7 of algorithm 1 for the original LISP variant defined in section 7.2.4, and equivalently in step 11 of algorithm 2 for the improved LISP variants defined in section 7.2.8. We can unify the DCAP and LISP effects by incorporating the DCAP influence into the LISP local phase adjustment calculation. In each case we substitute equation 9.2 for equation 9.1 in LISP, calculate average intercellular phase error ι_i as defined in algorithm 6, omitting step 9 in algorithm 6. This is functionally equivalent to running LISP and DCAP separately; we unify for ease of analysis.

$$\Delta\phi_i \Leftarrow -f_\alpha \theta_i \tag{9.1}$$

$$\Delta\phi_i \Leftarrow -f_\alpha \theta_i + f_\beta \iota_i \tag{9.2}$$

The net effect is that both LISP and DCAP exert influence on participating nodes simultaneously, exploiting similar coordination strategies to achieve similar but orthogonal goals. The net influence is similar to that of two partial derivatives on some measured quantity. In the following section 9.2.10 we consider the interaction of these influences, and the resulting consequences for selection of appropriate values of f_α and f_β.

9.2.10 Interplay between intracellular and intercellular coordination

LISP attempts to drive each cell into the *desynchronised equilibrium* state, as defined in section 7.2.3, whereas DCAP attempts to drive the set of all cells into the *synchronised equilibrium* state. These goals are not contradictory, as the latter acts on sets of *equivalent* nodes rather than the set of all nodes. However, there exists the possibility that the LISP mechanism may suggest that a given node should move its internal phase measurement in one direction, but the DCAP mechanism may suggest that the internal phase measurement should be moved in the other direction. We now consider the resolution of these potentially conflicting influences.

Recall from section 7.2.4 that LISP takes a feedback proportion parameter f_α, and from section 7.2.4 that DCAP takes a feedback proportion parameter f_β. Each f value determines the proportion of the appropriate *phase error* which is fed back from epoch to epoch at each node. We arrange for $f_\alpha \gg f_\beta$, for example such that f_β is an order of magnitude smaller than f_α. It follows that LISP exerts greater influence per unit time than DCAP. As LISP drives nodes within a cell toward desynchronised equilibrium, DCAP exerts insignificant influence.

However, as cells approach this desynchronised equilibrium state, the amount of phase change induced by LISP in each subsequent epoch becomes smaller. Although the absolute influence exerted by DCAP does not change, the relative magnitude of the DCAP-induced phase change grows in comparison to the LISP-induced phase change. It follows that LISP achieves intracellular desynchronised equilibrium relatively quickly, and DCAP achieves intercellular synchronised equilibrium relatively slowly. In each of the intra- and inter-cellular cases, however, as the system approaches the appropriate equilibrium state the rate of change per epoch decreases, such that when equilibrium is reached no timing changes are induced by either mechanism.

Equation 9.2 defines the change $\Delta\phi_i$ induced as node S_i fires its synchronisation event V_{ij} in epoch j. $\Delta\phi_i$ is implemented at node S_i as an instantaneous and discrete jump in ϕ_i. From the viewpoint of all other nodes, this is equivalent to a gradual change introduced continuously at rate $\frac{d\phi_i}{dt}$ until the S_i synchronisation event V_{ij+1} in the next epoch, labelled $j+1$. This is acceptable as instances of LISP and DCAP running at each node do not interact other than through these synchronisation events.

We label the elapsed time between V_{ij} and V_{ij+1} as δt_i where t is a measure of time passing for the complete system. Differentiating equation 9.2 with respect to t we obtain

equation 9.3 which describes $\frac{d\phi_i}{dt}$ for node S_i. $t_i \to e$ as the cell stabilises.

$$\frac{d\phi_i}{dt} = \frac{-f_\alpha \theta_i + f_\beta \iota_i}{\delta t_i} \tag{9.3}$$

As differentiation is a linear operation, we can trivially rewrite equation 9.3 as equation 9.4 and consider the θ_i and ι_i components separately.

$$\frac{d\phi_i}{dt} = -\frac{f_\alpha \theta_i}{\delta t_i} + \frac{f_\beta \iota_i}{\delta t_i} \tag{9.4}$$

If $f_\alpha \gg f_\beta$, it is obvious that $|-f_\alpha \theta_i| \gg |f_\beta \iota_i|$ unless $\iota_i \gg \theta_i$. The influence of DCAP on ϕ_i is less significant than that of LISP until LISP approaches its stable equilibrium, as defined in section 7.2.3, at which point θ_i becomes small. At this point the influence of DCAP becomes significant. As all nodes sharing the cell with S_i have a similar view of neighbouring cells, all will make similar DCAP-induced adjustments in the ι_i component of ϕ_i, and hence θ_i will remain small as ϕ_i is varied by DCAP.

9.2.11 Preventing transmission clashes in adjacent cells

If communication is implemented by broadcasts in a shared medium, it is possible for two or more nodes to have packets ready for transmission simultaneously. MAC protocols are responsible for preventing simultaneous broadcast and the resulting packet loss or corruption from interference [284], although the *hidden terminal* problem [98] may render it impossible to guarantee this does not happen. The specific MAC protocol selected is not significant to DCAP. The mechanisms employed by MAC protocols to avoid clashes are beyond the scope of this thesis, but are widely discussed in the literature [76].

As we use the broadcast times of synchronisation pulse transmissions as a proxy for the firing time of the underlying synchronisation event occurring at the broadcasting node, however, we must give some consideration to the timing of these broadcasts. LISP requires that the sync pulse packets are of minimal length κ (see section 7.2.6) which minimises the opportunity for clashes or delays. Furthermore, LISP specifically seeks to spread these sync pulse transmissions as far apart in time as is possible, as per section 7.2.3. CDAP goes further by segregating sync pulse broadcasts and all other network traffic, allocating short *SYNC* periods for the former during which other network traffic is disallowed (see section 8.2.1.1). As exactly one node will broadcast exactly one sync pulse transmission within each *SYNC* period this removes the opportunity for clashes.

Avoiding transmission clashes is more complicated in a multicellular network. The DCAP protocol requires that the equivalent synchronisation pulses in each cell be brought as close together as possible in the time domain, as discussed in section 9.2.6. It is obvious that as DCAP approaches synchronised equilibrium the sync pulse transmissions originating at the equivalent nodes in a set of adjacent cells will get closer and closer, such that the MAC protocol must intervene to prevent overlapping broadcasts. If a node is transmitting, the others must wait until the wireless medium becomes free before another may begin.

It follows that DCAP cannot guarantee to reduce the *intercellular phase error* between cells below a threshold value of $a\kappa$, where κ is the length of a sync pulse transmission and a is the maximum number of adjacent cells. At most $a+1$ nodes, one from a given cell and one from each of the a adjacent cells, must transmit. The first node claims the wireless medium and completes its broadcast. Other nodes must wait for the wireless medium to become available, the waiting process being managed by the MAC protocol. Eventually all $a+1$ transmissions complete, the final node having waited for a other transmissions to complete in at least $a\kappa$ time units. Any overhead or inefficiency of a non-perfect MAC protocol will increase this time. However, as κ is orders of magnitude smaller than e, it follows that $a\kappa$ remains small in comparison with e.

If the order in which cells are selected to make these broadcasts remains unchanged between system epochs, a small constant phase difference of at most $\frac{a\kappa}{\phi_{max}}$ will exist between pairs of cells. However, if the order is not unchanged, this phase difference will vary from epoch to epoch, inducing a small amount of jitter with the same upper bound on magnitude. In either case, if CDAP window management is used (see section 8.2.3.2) then the lower bound on synchronisation window length should be increased from 2κ to $(a+2)\kappa$. This allows sufficient time for the $a+1$ equivalent nodes in a given cell and its a adjacent neighbouring cells to complete their sequence of sync pulse transmissions, plus an additional period of time κ to allow dormant nodes to rejoin cells under ADCP as per section 8.3.3.

9.2.12 Measuring effectiveness

We define the metric U_1 to measure the effectiveness of DCAP. As DCAP has the single goal of synchronising sets of equivalent synchronisation pulse transmissions between adjacent cells we need only the single metric U_1 to measure the extent to which the system

of cells Γ conforms at any given time. In section 9.2.13 we take measurements against a number of different controlled factors to evaluate several aspects of system behaviour.

U_1: The average magnitude of intercellular phase error π_{xy} as measured between all adjacent pairs of cells (C_x, C_y) in D. This measure is obtained by first measuring ρ_{xpyq} for all nodes $S_p \in C_x$ and all nodes $S_q \in C_y$, which are identical within measurement error, and then taking the average of these ρ values as π_{xy} to minimise the influence of measurement error as measured values are equally likely to be larger or smaller than the true values. We then find the magnitude of each π_{xy}, and take the average of all such $|\pi|$ values as U_1 for the system of all cells in Γ. Measured in *phase units*. Defined in the range $[0, \frac{\phi_{max}}{n}]$ for cells of n active nodes. The ideal value of $U_1 = 0$.

9.2.13 Experimental results

DCAP was implemented in a modelled multicellular sensornet. A series of experiments was performed to establish the effectiveness of DCAP as defined in section 9.2.12.

9.2.14 Value of U_1 as a function of time

In this section we consider the convergence of U_1 toward its limiting value of 0 as a function of time measured in system epochs of length e time units. A network of 30 cells, each containing 10 nodes, was constructed with the cell boundaries taking the shape of a hexagonal planar tiling. In an infinite hexagonal planar tiling each hexagonal cell is surrounded by 6 neighbouring cells, but in a finite example the cells around the perimeter may have 2, 3 or 4 neighbouring cells depending on position. Real-world sensornets may take any spatial configuration as required to fit the physical environment, but in this section we are interested only in assessing the behaviour of DCAP. To ensure that each cell has all 6 neighbours we join the north and south edges, folding the plane into a cylinder, then join the east and west edges, folding the cylinder into a torus. This removes the issue of nonuniformity of cell adjacency degree, and hence avoids the influence of edge effects in the resulting experimental measurements.

We set the LISP feedback parameter $f_\alpha = 0.9$ and epoch length $e = 10s$, as section 7.2.10 indicates these values are effective for cells similar to those considered here, using LISP variant B described in section 7.2.8 to give improved performance in attaining desynchronised equilibrium within the cell population as discussed in section 7.2.3.

Section 9.2.10 states that the value of the DCAP feedback parameter f_β should generally be significantly smaller than f_α to allow correct interoperation, so f_β was set in the range $(0, 0.1]$. Larger values $f_\beta > 0.1$ are permitted but not considered here.

Figure 9.4 shows the measured value of U_1 as simulated time progressed from $t = 0$ epochs for $f_\beta = \{0.0005, 0.001, 0.01\}$. It can be seen that each f_β value results in U_1 converging on the ideal value $U_1 = 0$ over time, with higher values of f_β yielding a faster decline in U_1. This indicates that, over time, the system of cells approaches the desired *synchronised equilibrium* condition discussed in section 9.2.6; the paired equivalent periodic synchronisation events *between* two adjacent cells become progressively closer in the time domain, while synchronisation events *within* a given cells remain evenly spaced. Smaller values than $f_\beta = 0.005$ also yield this desired behaviour, but more slowly, and hence are omitted from figure 9.4 for clarity.

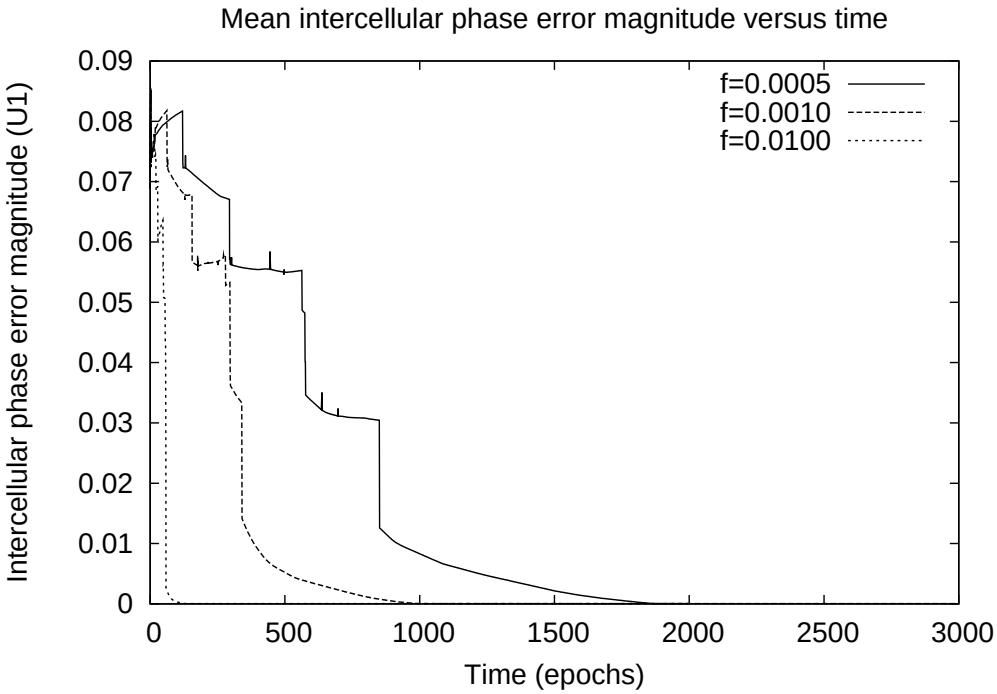


Figure 9.4: Convergence of U_1 as a function of time (standard y-axis)

Figure 9.5 displays the same experimental data but with a logarithmic y-axis.[2] The

[2]Note that the traces do not actually reach zero. This is simply an artefact of the finite time resolution of the simulation; if the calculations are performed with infinite precision then U_1 approaches 0 asymptotically. However, real sensornet hardware is also subject to similar timing artefacts as a consequence of continuous real-world time being quantised by mote timers such as CPU clocks. It follows that real-world implementations of any synchronisation mechanism is subject to artefacts of this type.

sharp decline observed in U_1, even under this logarithmic scale, indicates that DCAP is highly efficient at managing the firing times of sets of equivalent synchronisation events between adjacent cells. It can also be seen that higher values of f_β tend to bring about the desired synchronised equilibrium condition more quickly.

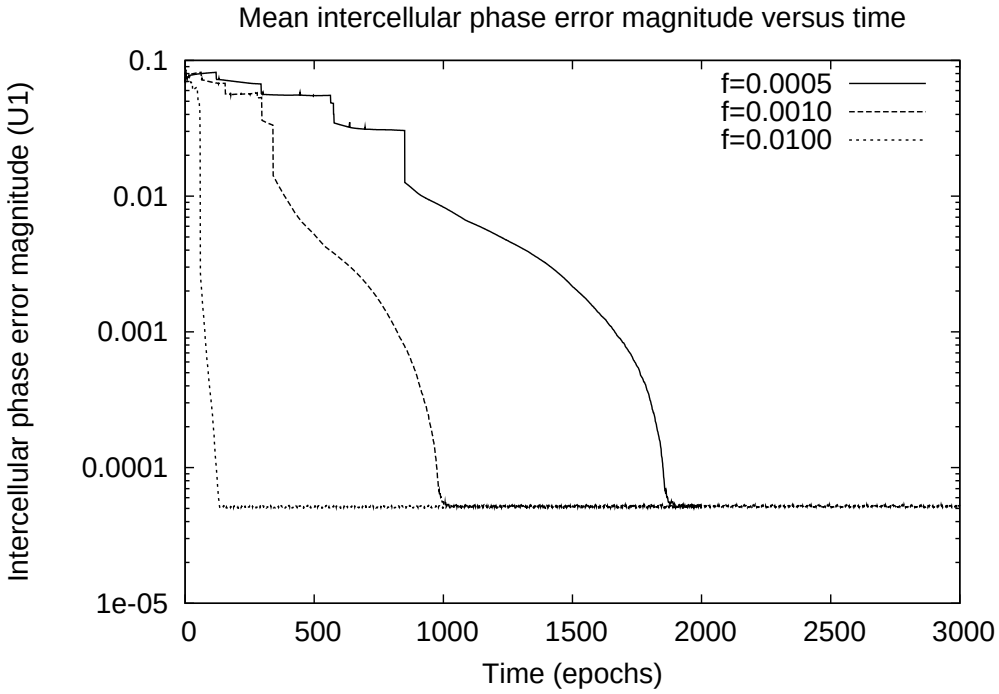


Figure 9.5: Convergence of U_1 as a function of time (logarithmic y-axis)

Sensornet designers may wish to select higher values of f_β to allow the network to approach the desired stable state more quickly, but taking care to prevent $f_\beta \approx f_\alpha$ which may otherwise prevent LISP from functioning effectively. Any such f_β value leads to an equivalent stable state so the selection of a specific value is a matter of balancing efficiency and stability, though if correct behaviour of distributed applications is dependent on adjacent cells being coordinated it may be beneficial to favour the former over the latter.

9.2.15 Time to synchrony for varying f_β

In this section we consider the time t required for a system to reach the synchronised equilibrium condition, discussed in section 9.2.6, as a function of varying *feedback parameter*, f_β. We employ the same cellular network considered in section 9.2.14, again considering DCAP behaviour for $f_\beta \in (0, 0.1]$. $U_1 \to 0$ asymptotically as $t \to \infty$ so we must define a threshold value for U_1 at which point the network of cells is considered sufficiently converged, and compare the value of t at which this condition is reached for each value of f_β

of interest.

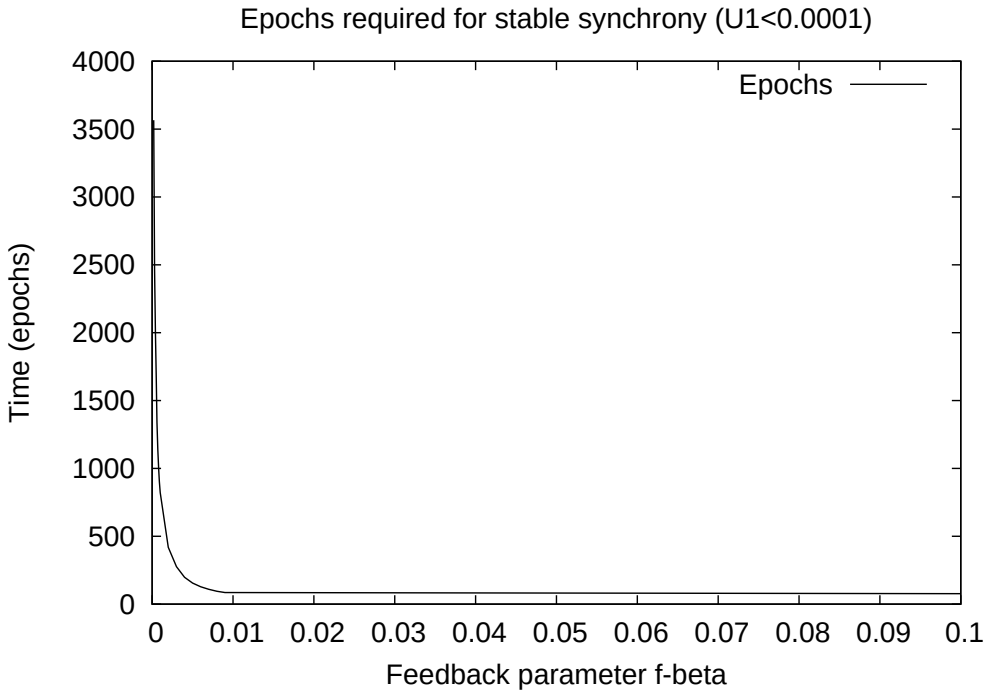


Figure 9.6: Epochs to stable synchronous equilibrium versus f_β

We define the threshold condition as $U_1 < 10^{-4}$ phase units because, when converted to 10^{-3} seconds for epochs of $10s$, this is similar to the $\kappa = 0.01s$ value employed in section 7.2.6 to define the limit of convergence for LISP in this network system. We define the system as converged at the point U_1 becomes smaller than 10^{-4} units and does not subsequently become larger. Figure 9.6 illustrates the number of epochs, on the y-axis, which must pass before convergence is reached for varying f_β, given on the x-axis.

In figure 9.6 we see that the number of epochs required is relatively large for small f_β, but declines rapidly as f_β increases; after $f_\beta \approx 0.01$ little further improvement is observable. This echoes the result given in section 9.2.14. Figure 9.7 presents the same results as a $\log - \log$ plot to enable further analysis of the relationship between f_β and t. Approximately linear relationships are observed between $\log f_\beta$ and $\log t$, with the response curve being divided into two segments of different but always negative gradient around $f_\beta \approx 0.01$. This matches the observation from figure 9.6 that comparatively small decrease in t is observed for f_β beyond this point.

If network response is reasonably insensitive to a wide range of f_β values, sensornet designers can select any value from this interval and achieve broadly similar results. It follows that sensornet designers may not need to allocate large amounts of effort into

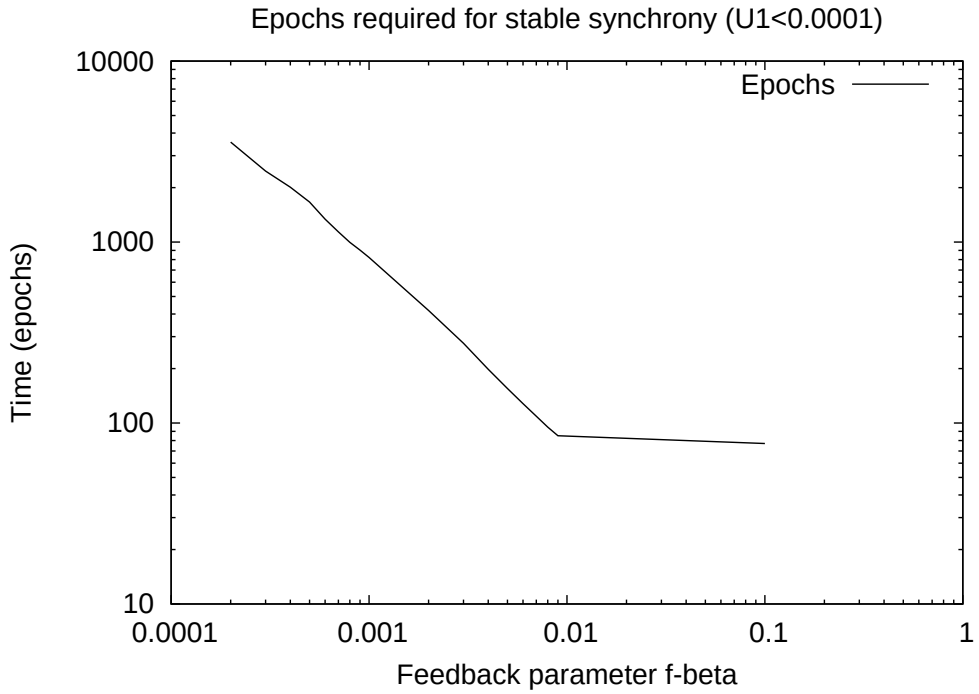


Figure 9.7: Epochs to stable synchronous equilibrium versus f_β (log-log plot)

tuning DCAP to find suitable near-optimal values, enabling an efficient design process. The near-linear relationship is also useful for predicting the number of epochs required to reach synchronous equilibrium states for an arbitrary f_β during the design process.

9.2.16 Time to synchrony for varying a

In this section we consider the time t required for a system to reach the synchronised equilibrium condition, discussed in section 9.2.6, as a function of varying *cell adjacency degree*, a, in the range $a \in [0, b]$. For an arbitrary network b could be any value $b \in \mathbb{N}^*$, depending on the interaction between the spatial configuration and communication characteristics. We set $b = 9$, encompassing the hexagonal cell configuration common to many planar cellular wireless networks [299] in which $a = 6$, but also allowing for other more exotic configurations which might arise in wired sensornets.

We reuse the cellular network considered in section 9.2.14. However, rather than define cell adjacency by position in a toroidal hexagonal grid, pairs of cells are randomly selected to be mutually adjacent, such that each cell is adjacent to a other cells. We measure time t to reach the stable synchronous equilibrium condition as discussed in section 9.2.15, but set $f_\beta = 0.01$ and vary a. We ignore the cases of $a = 0$ as no intercellular interaction is possible, and of $a = 1$ as this leads to a disconnected network in which paired cells can

interact with their partners but no other intercellular interactions are possible. For $a = 2$ the network must take the form of a ring to be fully connected. For each $a = 3$ we take the configuration of $a = 2$ and add randomly selected cell pair adjacencies, and so forth until we reach $a = b$.

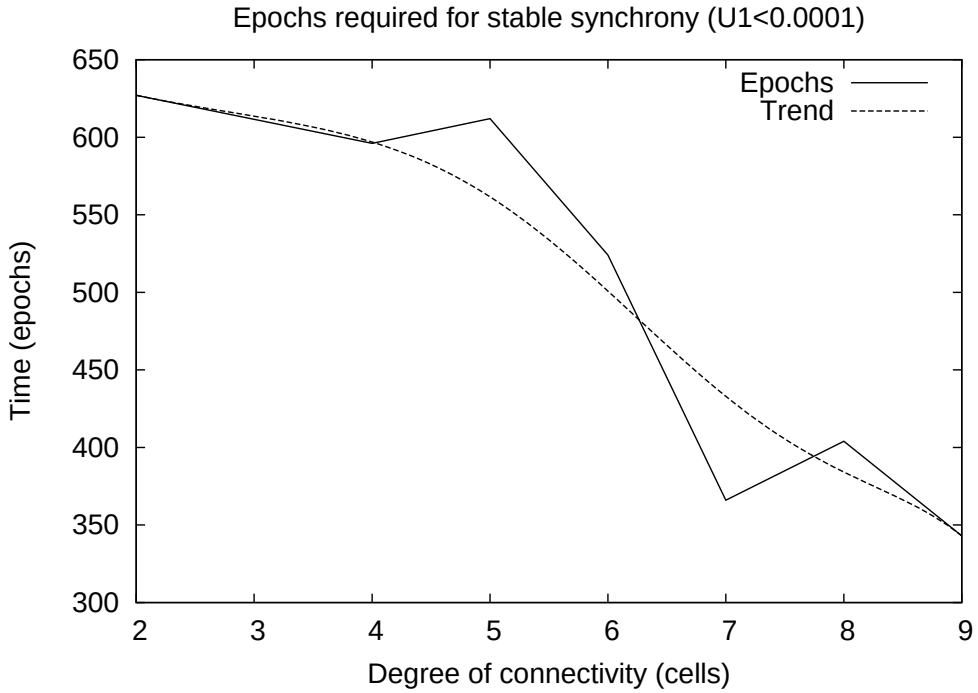


Figure 9.8: Epochs to stable synchronous equilibrium versus degree of connectivity, a

Figure 9.8 illustrates the relationship between t and a. In general, t tends to decline with increasing a, although the relationship is not smooth. This is explained by the mechanism used employed by DCAP to calculate intercellular phase error; the greater the number of cells that are adjacent to a given cell, the greater the proportion of the complete network which is visible to nodes in that cell.

More specifically, a greater number of adjacent cells implies a greater number of measured π values (see section 9.2.5) on which to calculate the necessary local phase adjustment as per section 9.2.9. Every cell influences, and is influenced by, every other cell in a fully connected network of cells. Where two cells are directly adjacent the mutual influence is stronger than the mutual influence of two non-adjacent cells connected only indirectly through one or more other cells. It follows that increasing the value of a increases the extent to which cells can directly influence other cells, and hence reduces the time required for the DCAP mechanism to achieve the synchronised equilibrium state.

9.3 Summary

The *Dynamic Cellular Accord Protocol* (DCAP) was defined in chapter 9. Whereas LISP provides *intracellular* coordination, DCAP provides *intercellular* coordination. If we have a sensornet composed of cells, each of which is managed by LISP, we find that although each observes very similar absolute measures of behaviour, there nevertheless exists the potential for clashing behaviour between cells. This potential for clashing behaviour clearly extends to anything based on, or coordinated by, the LISP protocol. More specifically, this includes the unmodified CDAP and ADCP protocols.

DCAP addresses this problem by aligning the relative *cell phase* of neighbouring cells. LISP and DCAP work in tandem; the former operates *within* cells, and the latter operates *between* cells. Whereas LISP applies the *desynchrony* principle to push apart synchronisation events within a cell, DCAP applies the *synchrony* principle to draw together equivalent sets of synchronisation events distributed between neighbouring cells. When comparing two neighbouring cells, the event firing time of each pair of equivalent synchronisation events is pushed ever closer and closer as time progresses.

Over time, each set of equivalent synchronisation events across all cells is pushed closer and closer, reaching an arbitrary specified level of equality in finite time within a given error margin implied by the length of synchronisation packets. This intercellular coordination allows networking protocols to operate at the level of the cell, rather than the individual node. This effectively defines a virtual network of cells, constructed from a physical network of nodes. As this virtual network is smaller than the physical network, it follows that networking protocols and applications which function better in smaller than larger networks can be more effective than is possible under a non-hierarchical network structure.

Chapter 10

Evaluation and conclusions

This chapter evaluates the novel work presented in chapters 3 to 9 against the research hypothesis defined in chapter 1, with discussion of the implications of the findings. Attention is given to possible directions for future extension work.

10.1 Evaluation

This section restates the hypothesis defined in section 1.4. We consider whether the results presented elsewhere in this thesis support or refute this hypothesis.

10.1.1 Restatement of hypothesis

The hypothesis defines the nature of the novel research contributions discussed in this thesis. This was stated in section 1.4 as:

> *Properties of sensornet behaviour can be measured and quantified such that objective evaluation and comparison among the set of candidate configurations is feasible. Performance improvements in measurable attributes of sensornet behaviour can be obtained through appropriate design decisions in network protocol selection and logical configuration.*

Each individual research element presented in chapters 3 to 9 is of value in itself, but is of greater value as a component of a larger problem. The hypothesis frames the larger problem into which each thread runs and contributes part of the full solution. To evaluate the hypothesis, we first consider whether each subsection of the hypothesis is supported by evidence. Having established this is true for all subsections, we can evaluate whether these fragments collectively support the hypothesis considered as a single entity.

10.1.2 Chapter 3

Chapter 3 develops a measurable model of sensornet systems. The model is constructed by providing definitions for a set of controllable factors, a set of measurable attributes, and a set of invariant properties. Each measurable attribute encapsulates the extent to which a given sensornet instance observes a given notion of desirable behaviour. Each controllable factor, derived from the definitions of protocols implemented in the simulated network and specified within a defined range, exerts influence on the behaviour of the network and hence the observed values of the measurable attributes. Typical failure modes of network protocol can be examined with respect to these attributes. This chapter provides evidence mainly for the *"Properties of sensornet behaviour can be measured and quantified such that objective evaluation and comparison among the set of candidate configurations is feasible."* part of the hypothesis.

Quantification of attributes of desirable behaviour requires not only a definition of these attributes, but also a means of measurement. Simulation methods allow measurement experiments that are completely controllable and repeatable, and define a simulated environment in which every aspect of every simulated entity can be examined over time. A simulation-based sensornet evaluation framework was defined and implemented with which quantified measures of these attributes can be obtained at sufficiently low cost to render feasible the acquisition of enough data points to reason about the measured system.

It was demonstrated that sensornet behaviour can be quantified in a number of measurable attributes associated with desirable network performance. The relationship between controlled factors and the induced measurable response can therefore be analysed quantitatively. Obtaining compromise solutions that give good responses in all measurable attributes is a multi-objective optimisation problem for which it is not appropriate to consider individual factors or individual responses in isolation. Chapters 4 to 6 address this issue.

10.1.3 Chapters 4 to 6

Chapters 4 to 6 evaluate the extent to which the specific tuning of a protocol influences the observed performance attributes, and presents solutions to the protocol tuning problem based on principled and evolutionary search techniques. These chapters provide evidence mainly for the *"Performance improvements in measurable attributes of sensornet behaviour can be obtained through appropriate design decisions in network protocol selec-*

tion and logical configuration." part of the hypothesis, focusing on *protocol* selection and configuration.

We require a mechanism to identify an appropriate tuning of an appropriate protocol in a specific given context. Ideally the network behaviour induced by this selection would offer optimal or near-optimal behaviour as measured in the solution quality attributes. Chapters 4 and 5 present two fundamentally different search-based approaches to solving this multi-objective optimisation problem; the first based on a principled Design Of Experiments approach, and the second based on an Evolutionary Algorithm approach.

Both search-based approaches were shown to be capable of obtaining an appropriate near-optimal configuration of a protocol in a networking context, with a quantitative measure of the disparity between this solution and the hypothetical perfect and worst-case solutions. A quantitative comparison was made between the solutions obtained by each method as a function of computation resources invested, to enable network designers to decide when an appropriate protocol selection and configuration result has been reached and further investment of effort cannot be justified.

The robustness of protocol tunings to variation in deployment context was examined in chapter 6. The protocol tuning solutions obtained by the two search-based methods offered acceptable robustness to deployment environment change, but some measured attributes tended to decline as the deployment environment diverged from the training environment. Network behaviour tended to become worse as networks became larger, or when the physical density of network nodes diverged in either direction from an ideal value.

The search-based approaches described in chapters 4 to 6 are highly effective in extracting the best possible behaviour for a given protocol in a given network. To achieve further gains in measurable attributes associated with desirable network behaviour, it is necessary to look beyond protocol tuning.

10.1.4 Chapters 7 to 9

Chapter 7 to 9 consider lightweight and fault-tolerant protocols for synchronised node state management to maximise the useful lifetime of networks composed of unreliable nodes in a harsh environment. These chapters provides evidence mainly for the *"Performance improvements in measurable attributes of sensornet behaviour can be obtained through appropriate design decisions in network protocol selection and logical configuration."* part of the hypothesis, focusing on *network* selection and configuration.

Whereas a network protocol cannot in itself physically add or remove network nodes, given a network composed of a set of nodes it is possible to manage which subset of these nodes actively participates in the network at any given time. This defines a logical network, a subset of the physical network, which possesses the desired properties.

An integrated suite of lightweight protocols was defined in chapters 7 to 9 for distributed sleep state management of nodes in a sensornet. A mechanism is provided for synchronisation of time-sensitive activity across a cellular sensornet. The logical configuration of the network is dynamically managed by having nodes enter a low power sleep state when not required to participate in the logical network, being effectively removed from the sensornet. Those nodes which do participate in the logical network at some given time are organised into a cellular structure, with a controlled active population in each cell, to provide an even spatial density of nodes throughout the deployment region.

Consequently, the logical network formed and managed by these protocols has the properties identified as necessary by the work in chapters 4 to 6. Reducing the active network size to the minimum which supports the sensornet application can dramatically improve network performance while reducing energy consumption, with the remaining inactive nodes available as spares to replace failing nodes.

The main obvious limitation of a protocol-driven logical network construction approach is that logical nodes can be situated only where physical nodes exist, which may be suboptimal within a given arbitrary physical sensornet. Furthermore, without application-specific knowledge of likely or plausible dataflows it is not possible to optimise the logical sensornet further, such that network entities can remain inactive in low power states for periods during which they are unlikely to be required. To achieve these further goals, further research of the type proposed in section 10.2 might be considered.

10.1.5 Support for the hypothesis considered as a whole

In sections 10.1.2 to 10.1.4 it is shown that this thesis presents evidence in favour of each element of the defined research hypothesis. It remains to consider whether the hypothesis as a whole is supported by the available evidence.

Each element of the hypothesis is supported by evidence presented in one or more of chapters 3 to 9. The flow of work presented in these chapters takes a logical path, consistent with the structure in which the hypothesis elements are composed and without unfilled gaps or unsubstantiated assumptions. All parts of the hypothesis are addressed by

one or more pieces of original research, yielding the associated novel findings, and without interruptions to the logical flow or unsupported claims. It is therefore reasonable to claim that the hypothesis defined in section 1.4 is supported by the work presented in this thesis.

10.2 Future work

The deployment environment of a sensornet may not be known, or may be incompletely specified, during sensornet design. Firstly, it is often difficult to formally specify real-world environments which are notoriously prone to unforeseen developments and interactions. Secondly, sensornets are often intended for deployment into hazardous or unstable environments, such that the deployment context may change significantly from the initial state and continue to change throughout the active life of the network. Thirdly, it may be desirable to construct a reusable sensornet design which can be deployed into a range of environments without modification, rather than to customise and optimise the sensornet design for each deployment instance.

In each of these cases it is clear that the sensornet design cannot be fully optimised in advance, as the deployment context which influences the selection of appropriate design elements and configurations is not known in advance. Nevertheless, the distributed application requirements can often be formulated in advance as a QoS contract specification. What remains is the challenge of ensuring the sensornet can fulfil this contract. One strategy is to measure the robustness of a given configuration to changing conditions, as tackled in chapter 4, and demonstrate that network performance is within acceptable bounds provided that changes to the sensornet and its deployment environment also lie within specified bounds. However, this strategy may yield suboptimal behaviour; a single compromise configuration is selected that offers acceptable performance, though not necessarily the best possible, where performance improvements could be achieved by reconfiguring the network.

A more sophisticated strategy is to adopt a *dynamic measurement and recalibration* approach, in which the configuration of the network changes over time in response to observed network and environmental conditions. A *self-optimising* sensornet would measure its own behaviour, and that of the deployment context, and make appropriate changes to its own configuration in response to these observations. This effectively creates a control feedback loop within a closed system of interacting nodes, such that the sensornet can adapt to unexpected situations and unforeseen conditions within certain limits. It follows

that the three deployment problems raised at the start of this section would be addressed by the implementation of a self-optimising network.

Section 3.5 begins to address per-packet dynamic selection of protocol parameters, and chapter 7 begins to address self-configuring networks which adapt their logical structure against statically specified targets. Although these sections provide some coverage of dynamic optimisation, they do not address the issue of obtaining an appropriate specification of desired behaviour, or the network-wide coordination of dynamically managed behaviour against defined QoS contracts. Interactions between layers of the network stack from the physical layer to the application layer should be taken into consideration if optimal configurations are to be obtained at runtime; unpredictable events in the physical environment affect the lower layers owing to the impact on packet propagation, and affects the higher layers owing to the application response to physical sensor data.

Sensornets can be designed to operate indefinitely by having nodes scavenge energy from their surroundings [145]. However, in any environment there are limits on the amount of energy that is available for practicable extraction, and hence there are limits on the rate at which any given node can harvest energy. If the maximum possible energy harvesting rate is lower than the energy consumption rate of active nodes, it is obvious that indefinite operation is not possible without careful network management. One strategy is to design hardware with lower energy demands [326], but some essential behaviours such as wireless communications are inherently energy-hungry [86]. An alternative approach is closer integration between network management and energy management subsystems. Constructing distributed duty schedules which balance in-network energy production and consumption, and time- and geography-sensitive sensor data production and consumption, has received surprisingly little attention in the literature. These approaches are orthogonal, and maximal benefit may be observed by applying both in unison.

Self-optimising networks must necessarily be self-measuring in order that sufficient information be made available upon which to base reconfiguration decisions. Each network node has a view of traffic flowing through itself, and a partial view of other traffic flowing through the network in its vicinity. Measurement data can be extracted from these observations, aggregated to conserve space, and transformed into a more useful format. This processed observation data might be used locally, or shared with neighbouring nodes, or distributed throughout the network. More sophisticated *machine learning* techniques might be applied to extract a more detailed understanding of packet flows, for example to

extract the periodicity or temporal correlations between packet flows and observed physical phenomena, such that more sophisticated configuration change responses are possible.

Self-optimising networks must measure their own behaviour, and determine whether improvement could be achieved by reconfiguration. The predictive statistical models discussed in section 4.4 could be used to determine an appropriate response to observed conditions, and to predict the likely improvement (if any) achievable by making a given configuration change. For some networks it may be desirable to coordinate this reconfiguration analysis and action throughout the entire system homogeneously, whereas in others it may be desirable to allow heterogeneous configurations in different subnetworks if local conditions differ throughout the network.

A further issue is that of the nature of any network configuration change. One approach is to define a number of discrete preset configurations at design time from which sensornet nodes can select the most appropriate example at runtime. Another approach is to allow configuration parameters to vary continuously, where appropriate. Whereas it may be easier to reason about the former approach, as each configuration selected from the set of presets can be examined in isolation, the latter approach may offer better scope for performance improvement and responsiveness to changing conditions. However, this would be at the expense of increased complexity in the reconfiguration mechanism and it would be harder to reason about the behaviour of the control feedback loop.

10.3 General conclusions and closing remarks

This thesis addressed the problem of *principled tuning and structuring methods for sensornets*, defining a testable hypothesis to encapsulate the essence of the problem. A sequence of research activities was planned and implemented, using both a top-down approach from the hypothesis definition and a bottom-up approach from intermediate results as they became available. These activities and their outputs provided evidence in support of the hypothesis.

A number of products and results, of interest and utility to sensornet designers, were delivered as a consequence of these research activities. These include protocol optimisation processes, and integrated protocol suites, all of which were measured and evaluated to an appropriate level. It was shown that significant improvements in sensornet behaviour can be obtained by careful tuning of the sensornet protocol stack and logical configuration, without modifying the underlying hardware platform or the overlying application software.

Abbreviations

ADCP Active Duty Control Protocol

ANOVA ANalysis Of VAriance

CDAP Cyclic Duty Allocation Protocol

CDMA Code Division Multiple Access

CSMA Carrier Sense Multiple Access

DCAP Dynamic Cellular Arbitration Protocol

DOE Design Of Experiments

EA Evolutionary Algorithm

FD Factorial Design

FFD Full Factorial Design

GPS Global Positioning System

IEEE Institute of Electrical and Electronics Engineers

IGF Implicit Geographic Forwarding

IP Internet Protocol

LIPS Lightweight Integrated Protocol Suite

LISP Lightweight Improved Synchronisation Primitive

MAC Media Access Control

MOEA Multi Objective Evolutionary Algorithm

MTBF Mean Time Between Failures

MTTF Mean Time To Failure

OSI Open Systems Interconnection

PDF Probability Density Function

QoS Quality of Service

RF Radio Frequency

TBG TTL Bounded Gossip

TCP Transmission Control Protocol

TDMA Time Division Multiple Access

TTL Time To Live

References

[1] J. Acebrón, L. Bonilla, C. Pérez Vicente, F. Ritort, and R. Spigler. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, 77(1):137–185, April 2005.

[2] M. Adamou, S. Khanna, I. Lee, I. Shin, and S. Zhou. Fair real-time traffic scheduling over a Wireless LAN. In *Proc. 22nd IEEE Real-Time Systems Symposium*, pages 279–288, London, 2-6 December 2001. IEEE Computer Society, Washington, DC.

[3] R. Adler. A study of locking phenomena in oscillators. *Proceedings of the Institute of Radio Engineers*, 34(6):351–357, June 1946.

[4] J. Ahn and P. Danzig. Speedup vs. simulation granularity. *IEEE/ACM Transactions on Networking*, 4(5):743–757, October 1996.

[5] K. Akkaya and M. Younis. An energy-aware QoS routing protocol for wireless sensor networks. In *Proc. 23rd International Conference on Distributed Computing Systems*, pages 710–715, Providence, RI, 19-22 May 2003. IEEE Computer Society, Los Alamitos, CA.

[6] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2(4):351–367, 2004.

[7] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[8] E. Alba, B. Dorronsoro, F. Luna, A. Nebro, P. Bouvry, and L. Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. *Computer Communications*, 30(4):685–697, February 2007.

[9] M. Ali and K. Langendoen. A case for peer-to-peer network overlays in sensor networks. In *Proc. 1st International Workshop on Wireless Sensor Network Ar-*

chitecture, pages 55–60, Cambridge, MA, 24 April 2007. ACM Press, New York, NY.

[10] M. Aly, A. Gopalan, and A. Youssef. Load-balancing query hotspots for next-generation sensornets. In *Proc. 50th IEEE Global Telecommunications Conference*, pages 775–779, Washington, DC, 26-30 November 2007. IEEE Communications Society, New York, NY.

[11] G. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 483–485, Atlantic City, NJ, 30 April - 2 May 1968. Thomson Book Company, Washington, DC.

[12] G. Anastasi, E. Ancillotti, M. Conti, and A. Passarella. Design and performance evaluation of a transport protocol for ad hoc networks. *The Computer Journal*, 52(2):186–209, March 2009.

[13] D. Anderson. BOINC: A system for public-resource computing and storage. In *Proc. 5th IEEE International Workshop on Grid Computing*, pages 4–10, Pittsburgh, PA, 8 November 2004. IEEE Computer Society, Los Alamitos, CA.

[14] K. Appel, W. Haken, and J. Kock. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[15] W. Archer, P. Levis, and J. Regehr. Interface contracts for TinyOS. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 158–165, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[16] J. Ariaratnam and S. Strogatz. Phase diagram for the Winfree model of coupled nonlinear oscillators. *Physical Review Letters*, 86(19):4278–4281, May 2001.

[17] J. Ash and L. Potter. Robust system multiangulation using subspace methods. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 61–68, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[18] A. Atkinson and A. Donev. *Optimum Experiment Design*. Clarendon Press, Oxford, 1992.

[19] R. Aylward and J. Paradiso. A compact, high-speed, wearable sensor network for biomotion capture and interactive media. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 380–389, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[20] R. Bagrodia and R. Meyer. PARSEC: A parallel simulation environment for complex system. *IEEE Computer*, 31(10):77–85, October 1998.

[21] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala. Improving simulation for network research. Technical Report 99-702, USC Computer Science Dept., March 1999.

[22] A. Baker, S. Dixon, F. Drabble, J. Gibbings, A. Lewkowicz, D. Moffat, and R. Shaw. *The Systematic Experiment*. Cambridge University Press, Cambridge, 1986.

[23] N. Baker. ZigBee and Bluetooth: strengths and weaknesses for industrial applications. *IEEE Computing and Control Engineering Journal*, 16(2):20–25, 2005.

[24] H. Balakrishnan, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48, 2003.

[25] P. Ballarini and A. Miller. Model checking medium access control for sensor networks. In *Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 255–262, Porto Sani, 13-15 October 2006. IEEE Computer Society, Los Alamitos, CA.

[26] D. Barbara. Mobile computing and databases - a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, 1999.

[27] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, 30(7):1655–1695, 2007.

[28] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *Proc. 6th ACM conference on Embedded Network Sensor Systems*, pages 43–56, Raleigh, NC, 5-7 November 2008. ACM Press, New York, NY.

[29] C. Barrett, S. Eidenbenz, L. Kroc, M. Marathe, and J. Smith. Parametric probabilistic routing in sensor networks. *Mobile Networks and Applications*, 10(4):529–544, 2005.

[30] C. Bernardeschi, P. Masci, and H. Pfeifer. Early prototyping of wireless sensor network algorithms in PVS. In *Proc. 27th International Conference on Computer Safety, Reliability, and Security*, pages 346–359, Newcastle-upon-Tyne, 22-25 September 2008. Springer-Verlag, Berlin.

[31] M. Bhardwaj, T. Garnett, and A. Chandrakasan. Upper bounds on the lifetime of sensor networks. In *Proc. 36th IEEE International Conference on Communications*, pages 785–790, Helsinki, 11-14 June 2001. IEEE Communications Society, Piscataway, NJ.

[32] S. Biaz, G. Holland, Y. Ko, and N. Vaidya. Evaluation of protocols for wireless networks. In *Proc. 5th International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 2150–2156, Las Vegas, NV, 28 June - 1 July 1999. CSREA Press.

[33] Bluetooth SIG, Kirkland, WA. *Bluetooth Specification*, 4th edition, December 2009.

[34] A. Bonivento, C. Fischione, L. Necchi, F. Pianegiani, and A. Sangiovanni-Vincentelli. System level design for clustered wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 3(3):202–214, August 2007.

[35] G. Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, Boston, MA, 2007.

[36] M. Botts, G. Percival, C. Reed, and J. Davidson. OGC Sensor Web enablement: Overview and high level architecture. In *GeoSensor Networks*, volume 4540/2008 of *Lecture Notes in Computer Science*, pages 175–190. Springer, Berlin, 2008.

[37] J. Bowen. The ethics of safety-critical systems. *Communications of the ACM*, 43(4):91–97, 2000.

[38] G. Box, J. Hunter, and W. Hunter. *Statistics for Experimenters*. Wiley, Hoboken, NJ, 2nd edition, 2005.

[39] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, Hoboken, NJ, 4th edition, 2008.

[40] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *Computer*, 33(5):59–67, 2000.

[41] A. Burns and G. Baxter. *Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective*, chapter 4. Time bands in systems structure, pages 74–88. Springer, London, April 2006.

[42] A. Burns and A. Wellings. *Real-Time Systems and Programming Languages*. Addison-Wesley, Harlow, UK, 3rd edition, March 2001.

[43] G. Buttazzo. *Hard Real-Time Computing Systems*. Springer, New York, NY, 2nd edition, 2004.

[44] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proc. 23rd Real-Time Systems Symposium*, pages 39–48, Austin, TX, 3-5 December 2002. IEEE Computer Society, Washington, DC.

[45] A. Carzaniga, D. Rosenblum, and A. Wolf. Content-based addressing and routing: A general model and its application. Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, Boulder, CO, January 2000.

[46] A. Carzaniga and A. Wolf. Content-Based Networking: A new communication infrastructure. In *Revised Papers from the NSF Workshop on Developing an Infrastructure for Mobile and Wireless Systems*, pages 59–68, Scottsdale, AZ, 15 October 2002. Springer-Verlag, London.

[47] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In *Proc. 2nd ACM International Workshop on Principles of Mobile Computing*, pages 38–43, Toulouse, 30-31 October 2002. ACM Press, New York, NY.

[48] M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Gună, M. Corrà, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: the Torre Aquila deployment. In *Proc. 8th International Conference on Information Processing in Sensor Networks*, pages 277–288, San Francisco, CA, 13-16 April 2009. IEEE Computer Society, Los Alamitos, CA.

[49] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Computer Communications Review*, 31(2):20–41, 2001.

[50] D. Chakraborty, A. Ashir, T. Suganuma, G. Mansfield Keeni, T. Roy, and N. Shiratori. Self-similar and fractal nature of internet traffic. *International Journal of Network Management*, 14(2):119–129, 2004.

[51] J. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, 2004.

[52] J. Chen, E. Shen, and Y. Sun. The deployment algorithms in wireless sensor net works: A survey. *Information Technology Journal*, 8(3):293–301, 2009.

[53] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks. In *Proc. 5th IEEE Singapore International Conference on Networks*, pages 197–211, Singapore, April 14-17 1997. IEEE Press, New York, NY.

[54] Chipcon Products. CC1000 single chip very low power RF transceiver datasheet, part number SWRS048A rev. A. http://focus.ti.com/lit/ds/symlink/cc1000.pdf. Accessed 25/07/2009.

[55] J. Cho, J. Lee, T. Kwon, and Y. Choi. Directional antenna at sink (DAaS) to prolong network lifetime in wireless sensor networks. In *Proc. 12th European Wireless Conference*, Athens, 2-5 April 2006. VDE Verlag, Berlin.

[56] M. Chrissis, M. Konrad, and S. Shrum. *CMMI: Guidelines for Process Integration and Product Improvement*. Addison Wesley, Boston, MA, 2nd edition, November 2006.

[57] D. Christensen, D. Brandt, U. Schultz, and K. Stoy. Neighbor detection and crosstalk elimination in self-reconfigurable robots. In *Proc. 1st International Conference on Robot Communication and Coordination*, pages 1–8, Athens, 15-17 October 2007. IEEE Press, Piscataway, NY.

[58] X. Chu. Open Sensor Web Architecture: Core services. Master's thesis, University of Melbourne, Australia, December 2005.

[59] C. Coello, G. Lamont, and D. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-objective Problems*. Genetic Algorithms and Evolutionary Computation. Springer, 2nd edition, September 2007.

[60] D. Corne, N. Jerram, J. Knowles, and M. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proc. 3rd Genetic and Evolutionary Computation Conference*, pages 283–290, San Francisco, CA, 7-11 July 2001. Morgan Kaufmann, San Mateo, CA.

[61] D. Corne, J. Knowles, and M. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In *Proc. 6th International Conference on Parallel Problem Solving from Nature*, pages 839–848, Paris, 16-20 September 2000. Springer-Verlag, London.

[62] P. Corsini, P. Masci, and A. Vecchio. Configuration and tuning of sensor network applications through virtual sensors. In *Proc. 4th IEEE International Conference on Pervasive Computing and Communications*, pages 316–320, Pisa, 13-17 March 2006. IEEE Computer Society, Los Alamitos, CA.

[63] J. Cowie, D. Nicol, and A. Ogielski. Modeling 100,000 nodes and beyond: Self-validating design. In *Proc. DARPA/NIST Workshop on Validation of Large Scale Network Simulation Models*, pages 1–4, Reston, VA, 25-26 May 1999. NIST,Gaithersburg, MD.

[64] F. Cristian, H. Aghili, and R. Strong. Clock synchronization in the presence of omission and performance failures, and processor joins. In *Proc. 16th IEEE International Symposium on Fault-Tolerant Computing Systems*, pages 218–223, Vienna, 1-4 July 1986. IEEE Computer Society Press, Washington, DC.

[65] Crossbow Technology. MICA2 wireless measurement system datasheet, part number 6020-0042-08 rev. A. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf. Accessed 09/01/2008.

[66] G. Cugola and M. Migliavacca. A context and content-based routing protocol for mobile sensor networks. In *Proc. 6th European Conference on Wireless Sensor Networks*, pages 69–85, Cork, 11-13 February 2009. Springer-Verlag, Berlin.

[67] D. Culler, P. Dutta, C. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Proc. 10th Workshop on Hot Topics in Networks*, pages 24–29, College Park, MD, 14-15 June 2005. USENIX Association, Berkeley, CA.

[68] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.

[69] J. Dahmann, R. Fujimoto, and R. Weatherly. The Department of Defense High Level Architecture. In *Proc. 29th Winter Simulation Conference*, pages 142–149, Atlanta, GA, 7-10 December 1997. ACM Press, New York, NY.

[70] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 3–12, Tel Aviv, 26-30 March 2000. IEEE Press, New York, NY.

[71] H. Davis and R. Miller. Power management for MICA2 motes. In *Proc. 1st Southern Appalachian Symposium on Programming Languages and Systems*, pages 10–15, Johnson City, TN, 25 October 2005. East Tennessee State University.

[72] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 1995(9):115–148, 1995.

[73] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proc. 6th International Conference on Parallel Problem Solving From Nature*, pages 849–858, London, 18-20 September 2000. Springer-Verlag, Berlin.

[74] J. Degesys, I. Rose, A. Patel, and R. Nagpal. DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks. In *Proc. 6th IEEE International Conference on Information Processing in Sensor Networks*, pages 11–20, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[75] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Data reduction techniques for sensor networks. Technical report, University of Maryland, July 2003.

[76] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, April 2006.

[77] R. Devaney. *A first course in chaotic dynamical systems*. Addison-Wesley, Reading, MA, 1992.

[78] Q. Dong. Maximizing system lifetime in wireless sensor networks. In *Proc. 4th international symposium on Information Processing in Sensor Networks*, pages 13–19, Los Angeles, CA, 25-27 April 2005. ACM Press, New York, NY.

[79] J. Doyle. *Routing TCP/IP*, volume 1. Cisco Press, Indianapolis, IN, 1998.

[80] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal stability based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communication*, 4(1):36–45, February 1997.

[81] P. Dutta and D. Culler. System software techniques for low-power operation in wireless sensor networks. In *Proc. 16th IEEE/ACM International Conference on Computer-Aided Design*, pages 925–932, San Jose, CA, 6-10 November 2005. IEEE Computer Society, Washington, DC.

[82] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A building block approach to sensornet systems. In *Proc. 6th ACM Conference on Embedded Networked Sensor Systems*, pages 267–280, Raleigh, NC, 5-7 November 2008. ACM Press, New York, NY.

[83] C. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica. A modular network layer for sensornets. In *Proc. 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 18–27, Seattle, WA, 6-8 November 2006. USENIX Association, Berkeley, CA.

[84] D. Egan. The emergence of ZigBee in building automation and industrial control. *IEEE Computing and Control Engineering Journal*, 16(2):14–19, 2005.

[85] A. Ercan, A. El Gamal, and L. Guibas. Object tracking in the presence of occlusions via a camera network. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 509–518, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[86] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. 11th IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2033–2036, Salt Lake City, UT, 7-11 May 2001.

[87] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. 5th International Conference on Mobile Computing and Networking*, pages 263–270, Seattle, WA, 17-19 August 1999. IEEE Communications Society, New York, NY.

[88] Euclid and T. Heath. *The Thirteen Books of The Elements*, volume 1. Dover Publications, Mineola, NY, 1956.

[89] C. Farrar, H. Sohn, M. Fugate, and J. Czarnecki. Integrated structural health monitoring. In *Proc. 8th Annual International Symposium on Smart Structures and Materials*, pages 119–136, Newport Beach, CA, 1-5 March 2001. Society of Photo-optical Instrumentation Engineers, Bellingham, WA.

[90] S. Farshchi, P. Nuyujukuian, A. Pesterev, I. Mody, and J. Judy. A TinyOS-enabled MICA2-based wireless neural interface. *IEEE Transactions on Biomedical Engineering*, 53(7):1416–1423, July 2006.

[91] S. Feit. *TCP/IP*. McGraw-Hill Series on Computer Communications. McGraw-Hill, New York, NY, 2nd edition, 1996.

[92] N. Fenton. *Software metrics: a rigorous approach*. Chapman Hall, London, 1991.

[93] A. Fraboulet, G. Chelius, and É. Fleury. Worldsens: Development and prototyping tools for application specific wireless sensor networks. In *Proc. 6th international symposium on Information Processing in Sensor Networks*, pages 176–185, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[94] C. Frank and K. Römer. Algorithms for generic role assignment in wireless sensor networks. In *Proc. 3rd International Conference on Embedded Networked Sensor Systems*, pages 230–242, San Diego, CA, 2-4 November 2005. ACM Press, New York, NY.

[95] P. Franklin. A six colour problem. *Journal of Mathematical Physics*, 13:363–369, 1934.

[96] H. Frey and D. Görgen. Planar graph routing on geographical clusters. *Ad Hoc Networks*, 3(5):560–574, December 2005.

[97] H. Friis. A note on a simple transmission formula. *Proceedings of the Institute of Radio Engineers*, 34(5):254–256, May 1946.

[98] C. Fullmer and J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. In *Proc. 22nd ACM SIGCOMM conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 39–49, Cannes, 16-18 September 1997. ACM Press, New York, NY.

[99] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Computer Communication Review*, 33(1):143–148, 2003.

[100] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mobile Computer Communications Review*, 5(4):11–25, 2001.

[101] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proc. 1st international conference on Embedded Networked Sensor Systems*, pages 89–102, Los Angeles, CA, 5-7 November 2003. ACM Press, New York, NY.

[102] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, Los Angeles, CA, February 2002.

[103] Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proc. 2nd International Workshop on Wireless Sensor Networks and Applications*, pages 108–114, San Diego, CA, 19 September 2003. ACM Press, New York, NY.

[104] A. Garcia-Hernando, J. Martinez-Ortega, J. López-Navarro, A. Prayati, and L. Redondo-López. *Problem Solving for Wireless Sensor Networks*, chapter "Hardware Platforms for WSNs", pages 17–50. Springer, London, December 2008.

[105] D. Gay, P. Levis, and D. Culler. Software design patterns for TinyOS. In *Proc. 9th ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 40–49, Chicago, IL, 15-17 June 2005. ACM Press, New York, NY.

[106] W. Gentzsch. Sun Grid Engine: towards creating a compute power grid. In *Proc. 1st International Symposium on Cluster Computing and the Grid*, pages 35–39, Brisbane, 15-18 May 2001. IEEE Computer Society, Los Alamitos, CA.

[107] O. Gnawali, K. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The Tenet architecture for tiered sensor networks. In *Proc. 4th International Conference on Embedded Networked Sensor Systems*, pages 153–166, Boulder, CA, 1-3 November 2006. ACM Press, New York, NY.

[108] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *Proc. 20th Annual USENIX Technical Conference*, pages 17–26, New Orleans, LA, 16-20 January 1995. USENIX Association, Berkeley, CA.

[109] E. Götürk. Emulating ad hoc networks: Differences from simulations and emulation specific problems. In *New Trends in Computer Networks*, volume 1 of *Advances in Computer Science and Engineering: Reports*. Imperial College Press, London, October 2005.

[110] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical Report 02-771, Computer Science Department, University of Southern California, Los Angeles, CA, September 2002.

[111] M. Grenier and N. Navet. Fine-tuning MAC-level protocols for optimized real-time QoS. *IEEE Transactions on Industrial Informatics*, 4(1):6–15, February 2008.

[112] L. Gu and J. Stankovic. Radio-triggered wake-up capability for sensor networks. In *Proc. 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 27–36, Toronto, 25-28 May 2004. IEEE Computer Society, Washington, DC.

[113] V. Guliashki, H. Toshev, and C. Korsemov. Survey of evolutionary algorithms used in multiobjective optimization. *Problems of Engineering Cybernetics and Robotics*, 60:42–54, 2009.

[114] R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using Kairos. In *Proc. 1st IEEE International Conference on Distributed Computing in Sensor Systems*, volume 3560 of *Lecture Notes in Computer Science*, pages 126–140, Marina del Rey, CA, June 2005. Springer, Berlin.

[115] P. Gunningberg, P. Lundgren, H. Nordstroem, and C. Tschudin. Lessons from experimental MANET research. *Ad Hoc Networking Journal*, 3(2):221–233, 2005.

[116] Z. Haas, J. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE Transactions on Networking*, 14(3):479–491, 2006.

[117] T. Hammel and M. Rich. A higher capability sensor node platform suitable for demanding applications. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 138–147, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[118] M. Handy, M. Haase, and D. Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-heads selection. In *Proc. 4th International Workshop on Mobile and Wireless Communications Network*, pages 368–372, Stockholm, 9-11 September 2002. IEEE Communications Society, Piscataway, NJ.

[119] L. Hatton. *Safer C: Developing software for high-integrity and safety-critical systems*. McGraw-Hill, Maidenhead, 1994.

[120] T. He, B. Blum, Q. Cao, J. Stankovic, S. Son, and T. Abdelzaher. Robust and timely communication over highly dynamic sensor networks. *Real-Time Systems*, 37(3):261–289, 2007.

[121] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proc. 26th IEEE International Conference on Distributed Computing Systems*, pages 46–55, Providence, RI, 19-22 May 2003. IEEE Computer Society, Los Alamitos, CA.

[122] P. Heawood. Map colour theorem. *Quarterly Journal of Mathematics*, 24:332–338, 1890.

[123] C. Hedrick. RFC 1058: Routing Information Protocol. Downloaded from http://www.ietf.org/rfc/rfc1058.txt, 1988. Accessed 28/05/2007.

[124] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proc. SCS Multiconference on Distributed Simulation*, pages 3–11, Phoenix, AZ, January 2001. Society for Computer Simulation, San Diego, CA.

[125] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proc. 18th ACM Symposium on Operating Systems Principles*, pages 146–159, Banff, 21-24 October 2001. ACM Press, New York, NY.

[126] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 3(4):660–670, 2002.

[127] M. Heissenbüttel, M. Braun, M. Wälchli, and T. Bernoulli. Evaluating the limitations of and alternatives in beaconing. *Ad Hoc Networks*, 5(5):558–578, 2007.

[128] T. Henderson, S. Roy, S. Floyd, and G. Riley. ns-3 project goals. In *Proc. 1st International Workshop on NS-2*, pages 13–20, Pisa, October 10 2006. ACM Press, New York, NY.

[129] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 360–369, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[130] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November - December 2002.

[131] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–103, Cambridge, MA, 7-11 March 2000. ACM Press, New York, NY.

[132] C. Hoare. *Communicating Sequential Processes*. Prentice Hall International, Upper Saddle River, NJ, 1985.

[133] X. Hou, D. Tipper, D. Tupho, and J. Kabara. Gossip-based sleep protocol (GSP) for energy efficient routing in wireless ad hoc networks. In *Proc. 5th IEEE Wireless Communications and Networking Conference*, volume 3, pages 1305–1310, New Orleans, LA, 16-20 March March 2003. IEEE Communications Society, New York, NY.

[134] L. Hu and D. Evans. Localization for mobile sensor network. In *Proc. 10th Annual International Conference on Mobile Computing and Networking*, pages 45–57, Philadelphia, PA, 36-30 September 2004. ACM Press, New York, NY.

[135] C. Hua and T. Yum. Asynchronous random sleeping for sensor networks. *ACM Transactions on Sensor Networks*, 3(3):710–714, 2007.

[136] P. Huang, D. Estrin, and J. Heidemann. Enabling large-scale simulations: Selective abstraction approach to the study of multicast protocols. In *Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 241–248, College Park, MD, 24-28 July 1998. IEEE Computer Society, Los Alamitos, CA.

[137] IEEE 802.11 Working Group. *IEEE Std. 802.11-1999, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE Std. 802.11.* IEEE Press, Piscataway, NJ, 1999. Reference number ISO/IEC 8802-11:1999(E).

[138] IEEE 802.15 Working Group. *IEEE Std. 802.15.1-2005, Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).* IEEE Press, Piscataway, NJ, 2005.

[139] IEEE 802.15 Working Group. *IEEE Std. 802.15.4-2006, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* IEEE Press, Piscataway, NJ, 2006.

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

[141] ISO. *ISO 9001: Quality systems - Model for quality assurance in design, development, production, installation and servicing.* International Organization for Standardisation, Geneva, 2nd edition, 1994.

[142] S. Ivanov, A. Herms, and G. Lukas. Validation of the ns-2 wireless model using simulation, emulation, and real network. In *Proc. 4th Workshop on Movile Ad-Hoc Networks*, pages 433–444, Bern, 1-2 March 2007. Verband der Elektrotechnik Elektronik Informationstechnik, Frankfurt.

[143] K. Iwanicki and M. van Steen. Multi-hop cluster hierarchy maintenance in wireless sensor networks: A case for gossip-based protocols. In *Wireless Sensor Networks*, volume 5432/2009 of *Lecture Notes in Computer Science*, pages 102–116. Springer, Berlin, January 2009.

[144] A. Jayasumana, Q. Han, and T. Illangasekare. Virtual Sensor Networks - a resource efficient approach for concurrent applications. In *Proc. 4th International Conference on Information Technology*, pages 111–115, Las Vegas, NV, 2-4 April 2007. IEEE Computer Society, Washington, DC.

[145] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *Proc. 4th IEEE International Conference on Information Processing in Sensor Networks*, pages 65–70, Los Angeles, CA, 25-27 April 2005. IEEE Press, Piscataway, NJ.

[146] X. Jiang, J. Taneja, J. Ortiz, A. Tavakoli, P. Dutta, J. Jeong, D. Culler, P. Levis, and S. Shenker. An architecture for energy management in wireless sensor networks. *ACM SIGBED Review*, 4(3):31–36, 2007.

[147] Z. Jin and R. Gupta. Improved distributed simulation of sensor networks based on sensor node sleep time. In *Proc. 4th IEEE International Conference on Distributed Computing in Sensor Systems*, pages 204–218, Santorini, June 11-14 2008. Springer-Verlag, Berlin.

[148] D. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. 1st Workshop on Mobile Computing Systems and Applications*, pages 158–163, Santa Cruz, CA, 8-9 December 1994. IEEE Computer Society, Washington, DC.

[149] D. Jourdan and O. de Weck. Layout optimization for a wireless sensor network using a multi-objective genetic algorithm. In *Proc. 60th IEEE Semiannual Vehicular Technology Conference*, pages 2466–2470, Milan, 17-19 May 2004. IEEE Vehicular Technology Society, New York, NY.

[150] J. Kahn, R. Katz, and K. Pister. Emerging challenges: mobile networking for Smart Dust. *Journal of Communications and Networks*, 2(3):188–196, September 2000.

[151] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.

[152] H. Kang and J. Wong. A localized multi-hop desynchronization algorithm for wireless sensor networks. In *Proc. 28th IEEE International Conference on Computer Communications*, pages 1–5, Rio de Janeiro, 19-25 April 2009. IEEE Press, New York, NY.

[153] A. Kansal, D. Potter, and M. Srivastava. Performance aware tasking for environmentally powered sensor networks. *ACM SIGMETRICS Performance Evaluation Review*, 32(1):223–234, June 2004.

[154] F. Kargl and E. Schoch. Simulation of MANETs: a qualitative comparison between JiST/SWANS and ns-2. In *Proc. 1st international workshop on System Evaluation for Mobile Platforms*, pages 41–46, San Juan, Puerto Rico, 11 June 2007. ACM Press, New York, NY.

[155] H. Karl and A. Willig. *Protocols And Architectures For Wireless Sensor Networks*. Wiley, Hoboken, NJ, 2005.

[156] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. 6th Annual Conference on Mobile Computing and Networking*, pages 243–254, Boston, MA, 6-11 August 2000. ACM Press, New York, NY.

[157] V. Kawadia and P. Kumar. Power control and clustering in ad hoc networks. In *Proc. 22nd IEEE Conference on Computer Communications*, pages 459–469, San Francisco, CA, 1-3 April 2003. IEEE Computer Society, Washington, DC.

[158] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proc. 2nd Evolutionary Multi-Criterion Optimization Conference*, pages 376–390, Faro, 8-11 April 2003. Springer, Berlin.

[159] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 254–263, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[160] D. Knuth. Two notes on notation. *American Mathematical Monthly*, 99(5):403–422, May 1992.

[161] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *Proc. 21st International Conference on Data Engineering*, pages 131–142, Tokyo, 5-8 April 2005. IEEE Computer Society, Washington, DC.

[162] D. Kotz, C. Newport, and C. Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Dept. of Computer Science, Dartmouth College, July 2003.

[163] B. Krishnamachari, D. Estrin, and S. Wicker. Modeling data-centric routing in wireless sensor networks. In *Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 241–248, New York, NY, 23-27 June 2002. IEEE Press, New York, NY.

[164] P. Krutchen. *The Rational Unified Process: An Introduction.* Addison-Wesley, Boston, MA, 3rd edition, 2003.

[165] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2-3):169–185, 2002.

[166] S. Kumar, T. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. *Wireless Networks*, 14(3):277–294, 2008.

[167] Y. Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In H. Araki, editor, *Proc. International Symposium on Mathematical Problems in Theoretical Physics*, volume 39, pages 420–422, Kyoto, 23-29 January 1975. Springer, Berlin.

[168] J. Kurose, M. Schwartz, and Y. Yemini. *Tutorial: hard real-time systems*, chapter Multiple-access protocols and time-constrained communication, pages 432–459. IEEE Computer Society Press, Los Alamitos, CA, 1989.

[169] P. Kyasanur, R. Choudhury, and I. Gupta. Smart Gossip: An adaptive gossip-based broadcasting service for sensor networks. In *Proc. 3rd IEEE Conference on Mobile Ad-hoc and Sensor Systems*, pages 91–100, Vancouver, 9-12 October 2006. IEEE Computer Society, Los Alamitos, CA.

[170] D. Lacks, M. Chatterjee, and T. Kocak. Design and evaluation of a distributed clustering algorithm for mobile ad hoc networks. *The Computer Journal*, 52(6):656–670, 2009.

[171] K. Lahiri, S. Dey, D. Panigrahi, and A. Raghunathan. Battery-driven system design: A new frontier in low power design. In *Proc. 15th International Conference on VLSI Design*, pages 261–266, Bangalore, 7-11 January 2002. IEEE Computer Society, Washington, DC.

[172] O. Landsiedel, K. Wehrle, B. Titzer, and J. Palsberg. Enabling detailed modeling and analysis of sensor networks. *Praxis der Informationsverarbeitung und Kommunikation*, 28(2):101–106, 2005.

[173] L. Lazos, R. Poovendran, and J. Ritcey. Probabilistic detection of mobile targets in heterogeneous sensor networks. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 519–528, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[174] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 21–30, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[175] J. Lehnert, D. Görgen, H. Frey, and P. Sturm. A scalable workbench for implementing and evaluating distributed applications in mobile ad hoc networks. In *Proc. 12th Western Simulation MultiConference*, pages 31–38, San Diego, CA, 19-21 January 2004. Society for Modeling and Simulation, San Diego, CA.

[176] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[177] P. Levis and D. Culler. Maté: a tiny virtual machine for sensor networks. In *Proc. 10th international conference on Architectural Support for Programming Languages and Operating Systems*, pages 85–95, San Jose, CA, 5-10 March 2002. ACM Press, New York, NY.

[178] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proc. 2nd USENIX/ACM Symposium on Network Systems Design and Implementation*, pages 343–356, Boston, MA, 2-4 May 2005. USENIX Association, Berkeley, CA.

[179] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. 1st ACM Conference on Embedded Networked Sensor Systems*, pages 126–137, Los Angeles, CA, 5-7 November 2003. ACM Press, New York, NY.

[180] H. Li, P. Shenoy, and K. Ramamritham. Scheduling communication in real-time sensor applications. In *Proc. 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 10–18, Toronto, 25-28 May 2004. IEEE Computer Society, Washington, DC.

[181] M. Li and Y. Liu. Underground structure monitoring with wireless sensor networks. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 69–78, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[182] X. Li, H. Kang, and H. Chen. Sensing workload scheduling in hierarchical sensor networks for data fusion applications. In *Proc. 3rd International Conference on Wireless Communications and Mobile Computing*, pages 214–219, Honolulu, HI, 12-16 August 2007. ACM Press, New York, NY.

[183] X. Li, K. Moaveninejad, and O. Frieder. Regional gossip routing for wireless ad hoc networks. *Mobile Networks and Applications*, 10(1-2):61–77, 2005.

[184] C. Liu, K. Wu, and J. Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel Distributed Systems*, 18(7):1010–1023, 2007.

[185] J. Liu. *Real-Time Systems*. Prentice Hall, Upper Saddle River, NJ, 2000.

[186] J. Liu and C. Lin. Energy-efficiency clustering protocol in wireless sensor networks. *Ad Hoc Networks*, 3(3):371–388, 2005.

[187] J. Liu, K. Lin, R. Bettati, D. Hull, and A. Yu. *Paradigms for Dependable Applications*, chapter 3.1. Foundations of Dependable Computing. Kluwer, Norwell, MA, 1st edition, September 1994.

[188] K. Liu, N. Abu-Ghazaleh, and K. Kang. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *Proc. 4th Annual IEEE International Conference on Pervasive Computing and Communications*, pages 42–46, Pisa, 13-17 March 2006. IEEE Computer Society, Washington, DC.

[189] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *Proc. 8th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 55–66, San Jose, CA, 24-27 September 2002. EEE Computer Society, Washington, DC.

[190] D. Lucarelli and I. Wang. Decentralized synchronization protocols with nearest neighbor communication. In *Proc. 2nd International Conference on Embedded Networked Sensor Systems*, pages 62–68, Baltimore, MD, 3-5 November 2004. ACM Press, New York, NY.

[191] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: a secure sensor network communication architecture. In *Proc. 6th International Symposium on Information Processing in Sensor Networks*, pages 479–484, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[192] C. Ma, K. Fang, and D. Lin. On the isomorphism of fractional factorial designs. *Journal of Complexity*, 17(1):86–97, March 2001.

[193] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. 29th ACM SIGMOD International Conference on Management of Data*, pages 491–502, San Diego, CA, 9-12 June 2003. ACM Press, New York, NY.

[194] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[195] E. Markatos. Tracing a large-scale peer to peer system: an hour in the life of Gnutella. In *Proc. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 65–74, Berlin, 22-24 May 2002. IEEE Computer Society, Los Alamitos, CA.

[196] MathWorks, Inc., Natwick, MA. *MATLAB Curve Fitting Toolbox*, 2009.

[197] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[198] J. McQuillan and I. Richer. *Computer Networks and Simulation*, chapter "A new network simulation technique", pages 187–193. North-Holland, New York, 1st edition, 1978.

[199] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949.

[200] D. Mills. Precision synchronization of computer network clocks. *ACM SIGCOMM Computer Communication Review*, 24(2):28–43, 1994.

[201] R. Mirollo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal Of Applied Mathematics*, 50(6):1645–1662, December 1990.

[202] A. Mohan, H. Wei, D. Gay, P. Buonadonna, and A. Mainwaring. End-to-end performance characterization of sensornet multi-hop routing. In *Proc. 2nd IEEE International Conference on Pervasive Services*, pages 27–36, Santorini, 11-14 July 2005. IEEE Computer Society, Los Alamitos, CA.

[203] G. Molina, E. Alba, and E. Talbi. Optimal sensor network layout using multiobjective metaheuristics. *Journal of Universal Computer Science*, 14(15):2549–2565, January 2008.

[204] J. Monks, V. Bharghavan, and W. Hwu. A power controlled multiple access protocol for wireless packet networks. In *Proc. 20th IEEE Conference on Computer Communications*, pages 219–228, Anchorage, AK, 22-26 April 2001. IEEE Computer Society, Los Alamitos, CA.

[205] D. Moodley and I. Simonis. A new architecture for the Sensor Web: The SWAP framework. In *Proc. 5th International Semantic Web Conference*, Athens, GA, 5-9 November 2006.

[206] E. Morenoff and J. McLean. An approach to standardizing computer systems. In *Proc. 22nd National ACM Conference*, pages 527–535, Washington, DC, August 1967. New York, NY.

[207] L. Mounier, L. Samper, and W. Znaidi. Worst-case lifetime computation of a wireless sensor network by model-checking. In *Proc. 4th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pages 1–8, Chania, Crete Island, 22 October 2007. ACM Press, New York, NY.

[208] G. Mulligan. The 6LoWPAN architecture. In *Proc. 4th workshop on Embedded Networked Sensors*, pages 78–82, Cork, 25-26 June 2007. ACM Press, New York, NY.

[209] S. Murthy and J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

[210] D. Musiani, K. Lin, and T. Rosing. Active sensing platform for wireless structural health monitoring. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 390–399, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[211] P. Naik and K. Sivalingam. *Wireless Sensor Networks*, chapter "A survey of MAC protocols for sensor networks", pages 93–107. Springer, New York, NY, May 2004.

[212] V. Naoumov and T. Gross. Simulation of large ad hoc networks. In *Proc. 6th International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 50–57, San Diego, CA, 14-19 September 2003. ACM Press, New York, NY.

[213] V. Naumov, R. Baumann, and T. Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proc. 7th ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 108–119, Florence, 2006. ACM Press, New York, NY.

[214] A. Newell. *Unified theories of cognition*. Harvard University Press, Cambridge, MA, 1990.

[215] R. Newton, G. Morrisett, and M. Welsh. The regiment macroprogramming system. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 489–498, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[216] L. Ni, Y. Zhu, J. Ma, Q. Luo, Y. Liu, S. Cheung, Q. Yang, M. Li, and M. Wu. Semantic Sensor Net: an extensible framework. *International Journal of Ad Hoc and Ubiquitous Computing*, 4(3/4):157–167, 2009.

[217] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. 5th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 151–162, Seattle, WA, 15-19 August 1999. ACM Press, New York, NY.

[218] P. Ölveczky. and S. Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. *Theoretical Computer Science*, 410(2-3):254–280, 2009.

[219] G. Ozsoyoglu and R. Snodgrass. Temporal and real-time databases: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

[220] M. Özsu and P. Valduriez. Distributed and parallel database systems. *ACM Computing Surveys*, 28(1):125–128, 1996.

[221] S. PalChaudhuri, A. Saha, and D. Johnson. Adaptive clock synchronization in sensor networks. In *Proc. 3rd IEEE International Conference on Information Processing in Sensor Networks*, pages 340–348, Palo Alto, CA, 22-23 April 2004. ACM Press, New York, NY.

[222] Y. Pan and X. Lu. Energy-efficient lifetime maximization and sleeping scheduling supporting data fusion and QoS in Multi-Sensornet. *Signal Processing*, 87(12):2949–2964, 2007. Special Section: Information Processing and Data Management in Wireless Sensor Networks.

[223] H. Park, J. Burke, and M. Srivastava. Design and implementation of a wireless sensor network for intelligent light control. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 370–379, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[224] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. 16th IEEE International Conference on Computer Communications*, pages 1405–1413, Las Vegas, NV, 22-25 September 1997. IEEE Computer Society, Los Alamitos, CA.

[225] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. Iyengar. Random asynchronous wakeup protocol for sensor networks. In *Proc. 1st International Conference on Broadband Networks*, pages 710–717, San Jose, CA, 25-29 October 2004. IEEE Computer Society, Los Alamitos, CA.

[226] A. Patel, J. Degesys, and R. Nagpal. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. In *Proc. 1st International Conference on Self-Adaptive and Self-Organizing Systems*, pages 87–96, Boston, MA, 9-11 July 2007. IEEE Computer Society, Washington, DC.

[227] N. Patwari and P. Agrawal. Effects of correlated shadowing: Connectivity, localization, and RF tomography. In *Proc. 7th International Conference on Information Processing in Sensor Networks*, pages 82–93, St. Louis, MO, 22-24 April 2008. IEEE Computer Society, Los Alamitos, CA.

[228] M. Paulk, editor. *The Capability Maturity Model: Guidelines for improving the software process*. Addison-Wesley, Cambridge, MA, 1996.

[229] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *SIGCOMM Computer Communications Review*, 24(4):234–244, 1994.

[230] C. Perkins and E. Royer. Ad-hoc On-demand Distance Vector routing. In *Proc. 2nd IEEE Workshop on Mobile Computer Systems and Applications*, pages 90–100, New Orleans, LA, 25-26 February 1999. IEEE Computer Society, Washington, DC.

[231] F. Perrone, D. Nicol, J. Liu, C. Elliot, and D. Pearson. Simulation modeling of large-scale ad-hoc sensor networks. In *Proc. 1st European Simulation Interoperability Workshop*, pages 12–23, London, 25-27 June 2001. Simulation Interoperability Standards Organization, Orlando, FL.

[232] L. Perrone and D. Nicol. A scalable simulator for TinyOS applications. In *Proc. 36th Winter Simulation Conference*, volume 1, pages 679–687, San Diego, CA, 8-11 December 2002. ACM Press, New York, NY.

[233] J. Ploennigs, P. Buchholz, M. Neugebauer, and K. Kabitzsch. Automated modeling and analysis of CSMA-type access schemes for building automation networks. *IEEE Transactions on Industrial Informatics*, 2(2):103–111, May 2006.

[234] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *Proc. 3rd International Conference on Embedded Networked Sensor Systems*, pages 76–89, San Diego, CA, 2-4 November 2005. ACM Press, New York, NY.

[235] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[236] K. Praditwong and X. Yao. A new multi-objective evolutionary optimisation algorithm: the Two-Archive algorithm. In *Proc. 3rd International Conference on Computational Intelligence and Security*, pages 95–104, Harbin, 15-19 December 2007. Springer-Verlag, Berlin.

[237] R. Pressman and D. Ince. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, New York, NY, 4th edition, 1997.

[238] F. Quintão, F. Nakamura, and G. Mateus. Evolutionary algorithm for the dynamic coverage problem applied to wireless sensor networks design. In *Proc. 7th IEEE Congress on Evolutionary Computation*, pages 1589–1596, Edinburgh, 2-4 September 2005. IEEE Press, New York, NY.

[239] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[240] K. Ramamritham. Real-time databases. *Distributed and Parallel Databases*, 1(2):199–226, 1993.

[241] D. Rao and P. Wilsey. Simulation of ultra-large communication networks. In *Proc. 7th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 112–119, College Park, MD, 24-28 October 1999. IEEE Press, New York, NY.

[242] D. Rao and P. Wilsey. Modeling and simulation of active networks. In *Proc. 34th Annual Simulation Symposium*, pages 177–184, Seattle, WA, 22-26 April 2001. IEEE Computer Society, Los Alamitos, CA.

[243] D. Rao and P. Wilsey. Multi-resolution network simulations using dynamic component substitution. In *Proc. 9th International Symposium in Modeling, Analysis and*

Simulation of Computer and Telecommunication Systems, pages 142–149, Cincinnati, OH, 15-18 August 2001. IEEE Computer Society, Los Alamitos, CA.

[244] S. Ratnasamy, B. Karp, S. Shenker, D.Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8(4):427–442, 2003.

[245] MIT Technology Review. 10 emerging technologies that will change the world, February 2003. Accessed 27/09/09.

[246] G. Riley, R. Fujimoto, and M. Ammar. A generic framework for parallelization of network simulations. In *Proc. 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 128–144, College Park, MD, 24-28 October 1999. IEEE Computer Society, Los Alamitos, CA.

[247] G. Ringel and J. Youngs. Solution of the heawood map-coloring problem. *Proceedings of the National Academy of Sciences*, 60:438–445, 1968.

[248] K. Römer. Programming paradigms and middleware for sensor networks. In *GI/ITG Workshop on Sensor Networks*, pages 49–54, Karlsruhe, February 2004. Gesellschaft für Informatik e.V, Bonn.

[249] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.

[250] M. Rudafshani and S. Datta. Localization in wireless sensor networks. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 51–60, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[251] B. Sakandar and D. Barnes. *Cisco LAN Switching Fundamentals*. Cisco Press Fundamentals Series. Cisco Press, 2nd edition, September 2004.

[252] S. Saroiu, P. Gummadi, and S. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems*, 9(2):170–184, 2003.

[253] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proc. 5th IEEE Wireless Communications and Networking Conference*, pages 1124–1130, New Orleans, LA, 16-20 March 2003. IEEE Communications Society, New York, NY.

[254] J. Schiller, Ac. Liers, and H. Ritter. ScatterWeb: A wireless sensornet platform for research and teaching. *Computer Communications*, 28(13):1545–1551, 2005.

[255] B. Schneier. *Secrets and Lies: Digital security in a networked world.* Wiley, New York, 1st edition, 2000.

[256] A. Schoofs, P. van der Stok, and P. Stanley-Marbell. Portability versus efficiency tradeoffs in MAC implementations for microsensor platforms. *IEEE Embedded Systems Letters*, 1(1):24–27, May 2009.

[257] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *SIGCOMM Computer Communications Review*, 33(1):137–142, 2003.

[258] M. Shepperd and D. Ince. *Derivation and Validation of Software Metrics.* Clarendon Press, Oxford, 1993.

[259] K. Shin, A. Abraham, and S. Han. Self organizing sensor networks using intelligent clustering. In *Proc. 6th International Conference on Computational Science and its Applications*, pages 40–49, Glasgow, 8-11 May 2006. Springer, Berlin.

[260] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proc. 2nd International Conference on Embedded Networked Sensor Systems*, pages 188–200, Baltimore, MD, 3-5 November 2004. ACM Press, New York, NY.

[261] K. Shuaib, M. Boulmalf, F. Sallabi, and A. Lakas. Co-existence of ZigBee and WLAN, a performance study. In *Proc. 5th Wireless Telecommunications Symposium*, pages 1–6, Los Angeles, CA, 27-29 April 2006. IEEE Press, New York, NY.

[262] M. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proc. 5th IEEE Wireless Communications and Networking Conference*, volume 2, pages 1266–1273, New Orleans, LA, 16-20 March 2003. IEEE Communications Society, New York, NY.

[263] I. Simonis. Sensor Webs: A roadmap. In *Proc. 1st Göettinger GI and Remote Sensing Days*, Göttingen, 2004. Institute for Geoinformatics, University of Müenster.

[264] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley. Tracking multiple targets using binary proximity sensors. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 529–538, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[265] S. Singh and C. Raghavendra. PAMAS - power aware multi-access protocol with signalling for ad hoc networks. *SIGCOMM Computer Communications Review*, 28(3):5–26, 1998.

[266] S. So, F. Koushanfar, A. Kosterev, and F. Tittel. LaserSPECks: laser SPECtroscopic trace-gas sensor networks - sensor integration and applications. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 226–235, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[267] A. Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, and H. Zhang. J-Sim: A simulation environment for wireless sensor networks. In *Proc. 38th Annual Symposium on Simulation*, pages 175–187, San Diego, CA, 4-6 April 2005. IEEE Computer Society, Los Alamitos, CA.

[268] A. Sobeih, M. Viswanathan, and J. Hou. Check and Simulate: A case for incorporating model checking in network simulation. In *Proc. 2nd ACM-IEEE International Conference on Formal Methods and Models for Codesign*, pages 27–36, San Diego, CA, 22-25 June 2004. IEEE Computer Society, Los Alamitos, CA.

[269] P. Sousa, P. Carvalho, and V. Freitas. Scheduling time-sensitive IP traffic. In *Proc. 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pages 368–380, Belfast, 7-10 September 2003. Springer-Verlag, Berlin.

[270] M. Spiegel and L. Stephens. *Theory and Problems of Statistics*. McGraw-Hill, New York, NY, 1998.

[271] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Some implications of low power wireless to IP networking. In *Proc. 5th Workshop on Hot Topics in Networks*, pages 31–36, Irvine, CA, 29-30 November 2006. ACM Press, New York, NY.

[272] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proc. 7th International Workshop on Security Protocols*, pages 172–194, Cambridge, 19-21 April 2000. Springer-Verlag, London.

[273] J. Stankovic. Distributed computing. Technical Report UM-CS-1992-011, University of Massachusetts at Amherst, 1992.

[274] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003.

[275] P. Stavroulakis, editor. *Chaos Applications in Telecommunications*. CRC Press, Boca Raton, FL, 1st edition, 2006.

[276] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. PIPENET - a wireless sensor network for pipeline monitoring. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 264–273, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[277] S. Strogatz. From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1-4):1–20, 2000.

[278] S. Strogatz. *Sync: the emerging science of spontaneous order*. Penguin, London, 2003.

[279] B. Sundararaman, U. Buy, and A. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[280] S. Sundresh, W. Kim, and G. Agha. SENS: A sensor, environment and network simulator. In *Proc. 37th Annual Simulation Symposium*, pages 221–230, Arlington, VA, 18-22 April 2004. IEEE Computer Society, Los Alamitos, CA.

[281] I. Symeou and A. Burns. Knowledge-theoretic protocols for wireless sensor networks. In *Proc. 3rd IET UK Embedded Forum*, pages 3–9, Durham, 2-3 April 2007. IET Press, London.

[282] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. 2nd international conference on Embedded Networked Sensor Systems*, pages 214–226, Baltimore, MD, 2004. ACM Press, New York, NY.

[283] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. 1st European Workshop on Sensor Networks*, pages 307–322, Berlin, 19-21 January 2004. Springer-Verlag, Berlin.

[284] A. Tanenbaum. *Computer Networks*. Pearson, Harlow, 3rd edition, March 1996.

[285] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topologies, power laws, and hierarchy. *SIGCOMM Computer Communications Review*, 32(1):76–76, 2002.

[286] M. Taqqu and V. Teverosky. Is network traffic self-similar or multifractal? *Fractals*, 5(1):63–73, 1997.

[287] J. Tate and I. Bate. YASS: A scaleable sensornet simulator for large scale experimentation. In *Proc. 31st Communicating Process Architectures Conference*, pages 411–430, York, 7-10 September 2008. IOS Press, Amsterdam.

[288] J. Tate and I. Bate. Energy efficient duty allocation protocols for wireless sensor networks. In *Proc. 14th IEEE Conference on Engineering of Complex Computer Systems*, pages 58–67, Potsdam, 2-4 June 2009. IEEE Press, New York, NY.

[289] J. Tate and I. Bate. An improved lightweight synchronisation primitive for sensornets. In *Proc. 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 448–457, Macau, 12-15 October 2009. IEEE Computer Society, Los Alamitos, CA.

[290] J. Tate and I. Bate. Sensornet protocol tuning using principled engineering methods. *The Computer Journal*, 2009. Advance Access publication on 19 August 2009 at http://comjnl.oxfordjournals.org/cgi/content/abstract/bxp077.

[291] J. Tate and I. Bate. Tuning complex sensornet systems using principled engineering methods. In *Proc. 16th IEEE International Conference on the Engineering of Computer Based Systems*, pages 275–284, San Francisco, CA, 14-16 April 2009. IEEE Computer Society, Los Alamitos, CA.

[292] J. Tate and I. Bate. Understanding behavioural tradeoffs in large-scale sensornet design. In *Proc. 1st IEEE International Workshop on Quantitative Evaluation of Large-Scale Systems and Technologies*, pages 1085–1091, Bradford, 26-29 May 2009. IEEE Press, New York, NY.

[293] J. Tate and I. Bate. Do sensornet protocol variants yield real benefits? In *Proc. 17th IEEE International Conference on the Engineering of Computer Based Systems (to appear)*, Oxford, 22-26 March 2010. IEEE Computer Society, Los Alamitos, CA.

[294] J. Tate and I. Bate. Maintaining stable node populations in long-lifetime sensornets. In *Proc. 15th IEEE Conference on Engineering of Complex Computer Systems (to appear)*, Oxford, 22-26 March 2010. IEEE Computer Society, Los Alamitos, CA.

[295] J. Tate, I. Bate, and S. Poulding. Tuning protocols to improve the energy efficiency of sensornets. In *Proc. 4th IET UK Embedded Forum*, pages 51–61, Durham, 9-10 September 2008. IET Press, London.

[296] J. Tate, B. Woolford-Lim, I. Bate, and X. Yao. Comparing design of experiments and evolutionary approaches to multi-objective optimisation of sensornet protocols. In *Proc. 11th IEEE Congress on Evolutionary Computation*, pages 1137–1144, Trondheim, 18-21 May 2009. IEEE Press, New York, NY.

[297] A. Tavakoli, P. Dutta, J. Jeong, S. Kim, J. Ortiz, D. Culler, P. Levis, and S. Shenker. A modular sensornet architecture: past, present, and future directions. *ACM SIGBED Review*, 4(3):49–54, 2007.

[298] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[299] H. Tian and H. Shen. An optimal coverage scheme for wireless sensor network. In *Proc. 4th International Conference on Networking*, pages 722–730, Réunion, 17-21 April 2005. Springer-Verlag, Berlin.

[300] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless microsensor network models. *SIGMOBILE Mobile Computing and Communications Review*, 6(2):28–36, 2002.

[301] S. Tilak, V. Kolar, N. Abu-Ghazaleh, and K. Kang. Dynamic localization control for mobile sensor networks. In *Proc. 24th IEEE International Performance, Computing,and Communications Conference*, pages 587–592, Phoenix, AZ, 7-9 April 2005. IEEE Computer Society, Los Alamitos, CA.

[302] B. Titzer, D. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proc. 4th international symposium on Information Processing in*

Sensor Networks, pages 67–72, Los Angeles, CA, 25-27 April 2005. IEEE Press, New York, NY.

[303] C. Toh. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communnications*, 4(2):103–139, 1997.

[304] C. Toh, P. Mahonen, and M. Uusitalo. Standardization efforts and future research issues for wireless sensors and mobile ad hoc networks. *IEICE Transactions on Communications*, 88(9):3500–3507, 2005.

[305] M. Totaro and D. Perkins. Using statistical design of experiments for analyzing mobile ad hoc networks. In *Proc. 8th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 159–168, Montreal, 10-13 October 2005. ACM Press, New York, NY.

[306] R. Tridgell. The CCIR radio-paging code no. 1: A new world standard. In *Proc. 42nd IEEE Vehicular Technology Conference*, volume 32, pages 403–406, San Diego, CA, 23-26 May 1982. IEEE Press, New York, NY.

[307] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Duke University, Durham, NC, April 2000.

[308] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. 5th symposium on Operating Systems Design and Implementation*, pages 271–284, Boston, MA, 9-11 December 2002. ACM Press, New York, NY.

[309] D. Veitch, N. Hohn, and P. Abry. Multifractality in TCP/IP traffic: the case against. *Computer Networks*, 48(3):293–313, 2005.

[310] N. Vlajic and D. Xia. Wireless sensor networks: to cluster or not to cluster? In *Proc. 7th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 268–277, Buffalo-Niagara Falls, NY, 26-29 June 2006. IEEE Computer Society, New York, NY.

[311] T. Voigt, J. Alonso, A. Dunkels, H. Ritter, and J. Schiller. Connecting wireless sensornets with TCP/IP networks. In *Proc. 2nd International Conference on Wired/Wireless Internet Communications*, pages 143–152, Frankfurt, 4-6 February 2004. Springer, Berlin.

[312] D. Wang. An energy-efficient clusterhead assignment scheme for hierarchical wireless sensor networks. *International Journal of Wireless Information Networks*, 15(2):61–71, June 2008.

[313] Q. Wang, X. Liu, J. Hou, and L. Sha. GD-aggregate: A WAN virtual topology building tool for hard real-time and embedded applications. In *Proc. 28th IEEE International Real-Time Systems Symposium*, pages 379–388, Tuscon, AZ, 3-6 December 2007. IEEE Computer Society, Los Alamitos, CA.

[314] X. Wang and A. Apsel. Pulse coupled oscillator synchronization for low power UWB wireless transceivers. In *Proc. 50th IEEE International Midwest Symposium on Circuits and Systems*, pages 1524–1527, Montreal, 5-8 August 2007. IEEE Press, New York, NY.

[315] T. Wark, C. Crossman, W. Hu, Y. Guo, P. Valencia, P. Sikka, P. Corke, C. Lee, J. Henshall, K. Prayaga, J. O'Grady, M. Reed, and A. Fisher. The design and evaluation of a mobile sensor/actuator network for autonomous animal control. In *Proc. 6th international conference on Information Processing in Sensor Networks*, pages 206–215, Cambridge, MA, 25-27 April 2007. ACM Press, New York, NY.

[316] B. Warneke, M. Last, B. Liebowitz, and K. Pister. Smart Dust: Communicating with a cubic-millimeter computer. *IEEE Computer*, 34(1):44–51, 2001.

[317] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proc. 1st International Symposium on Networked Systems Design and Implementation*, pages 29–42, New York, NY, 29-31 March 2004. ACM Press, New York, NY.

[318] C. Wen and W. Sethares. Automatic decentralized clustering for wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2005(5):686–697, 2005.

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

[320] A. Wheeler. Commercial applications of wireless sensor networks using ZigBee. *IEEE Communications Magazine*, 45(3):70–77, 2007.

[321] B. White, J. Lepreau, and S. Guruprasad. Lowering the barrier to wireless and mobile experimentation. *SIGCOMM Computer Communications Review*, 33(1):47–52, 2003.

[322] D. White and S. Poulding. A rigorous evaluation of crossover and mutation in genetic programming. In L. Vanneschi, S. Gustafson, and M. Ebner, editors, *Proc. 12th European Conference on Genetic Programming*, volume 5481 of *LNCS*, pages 220–231, Tuebingen, 15-17 April 2009. Springer, Berlin.

[323] R. Wilson. *Four Colours Suffice*. Allen Lane, London, 2002.

[324] A. Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoretical Biology*, 16:15–42, 1967.

[325] A. Winfree. *The Geometry of Biological Time*. Springer, London, 2nd edition, June 1980.

[326] K. Wong and D. Arvind. Specknets: New challenges for wireless communication protocols. In *Proc. 3rd IEEE International Conference on Information Technology and Applications*, volume 2, pages 728–733, Sydney, 4-7 July 2005. IEEE Computer Society, Washington, DC.

[327] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. 1st international conference on Embedded Networked Sensor Systems*, pages 14–27, Los Angeles, CA, 5-7 November 2003. ACM Press, New York, NY.

[328] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. 1st International Conference on Embedded Networked Sensor Systems*, pages 14–27, Los Angeles, CA, 5-7 November 2003. ACM Press, New York, NY.

[329] W. Wood and D. Martin. *Experimental Method*. Athlone Press, London, 1974.

[330] K. Wu, Y. Gao, F. Li, and Y. Xiao. Lightweight deployment-aware scheduling for wireless sensor networks. *Mobile Networks and Applications*, 10(6):837–852, 2005.

[331] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. 7th international conference on Mobile Computing and Networking*, pages 70–84, Rome, 16-21 July 2001. ACM Press, New York, NY.

[332] S. Yamashita, T. Shimura, K. Aiki, K. Ara, Y. Ogata, I. Shimokawa, T. Tanaka, H. Kuriyama, K. Shimada, and K. Yano. A 15x15 mm, 1 μA, reliable sensor-net module: enabling application-specific nodes. In *Proc. 5th International Conference on Information Processing in Sensor Networks*, pages 383–390, Nashville, TN, 19-21 April 2006. ACM Press, New York, NY.

[333] E. Yang, N. Haridas, A. El-Rayis, A. Erdogan, and T. Arslan. Multiobjective optimal design of MEMS-based reconfigurable and evolvable sensor networks for space applications. In *Proc. 2nd NASA/ESA Conference on Adaptive Hardware and Systems*, pages 27–34, Edinburgh, 5-8 August 2007. IEEE Computer Society, Los Alamitos, CA.

[334] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 878–890, Miami, FL, 13-17 March 2005. IEEE Press, New York, NY.

[335] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. 23rd International Conference on Distributed Computing Systems*, pages 28–37, Providence, RI, 19-22 May 2003. IEEE Computer Society, Los Alamitos, CA.

[336] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[337] R. York. Nonlinear analysis of phase relationships in quasi-optical oscillator arrays. *IEEE Transactions on Microwave Theory and Techniques*, 41(10):1799–1809, October 1993.

[338] G. Yu. Cell-based coverage management and routing protocol for wireless sensor networks. In *Proc. 3rd International Conference on Mobile Technology, Applications, and Systems*, pages 1–7, Yilan, 10-12 September 2008. ACM Press, New York, NY.

[339] D. Yupho and J. Kabara. The effect of physical topology on wireless sensor network lifetime. *Journal of Networks*, 2(5):14–23, September 2007.

[340] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proc. 12th Workshop on Parallel and Distributed*

Simulation, pages 154–161, Banff, 26-29 May 1998. IEEE Computer Society, Los Alamitos, CA.

[341] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Transactions on Sensor Networks*, 2(2):221–262, 2006.

[342] ZigBee Alliance, San Ramon, CA. *ZigBee Specification*, January 2007.

[343] H. Zimmerman. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

[344] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proc. 10th International Conference on Parallel Problem Solving from Nature*, pages 832–842, Birmingham, 18-22 September 2004. Springer-Verlag, London.

[345] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In *Proc. 4th Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, 19-21 September 2001. Wiley, New York.

[346] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[347] M. Zorzi and R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. *IEEE Transactions on Mobile Computing*, 2(4):337–348, 2003.

Appendix A

Protocol definitions

This section presents algorithmic definitions of protocols considered in chapters 3 to 6.

A.1 TTL-Bounded Gossip

Algorithm 7 defines the behaviour of a node which can transmit and receive data packets under the TBG protocol [217]. The protocol does not specify required behaviour for a packet sender so we present no corresponding algorithm for this role.

Algorithm 7 : TTL-Bounded GOSSIP (relay)

Packet π defines destinations set, D
Packet π defines hops TTL, h
Packet π defines delivery deadline, d
Node R maintains set of previously-seen packets, C
Node R maintains packet broadcast queue, Q
GOSSIP rebroadcast probability is $X_6 \in [0, 1]$
Packet π received successfully by candidate relay R at time τ
if $R \in D$ **then**
    Consume packet π at R
else
    if $\pi \in C$ **then**
        Drop packet π at R
    else
        Record π at R such that $C' = C \cup \{\pi\}$
        if $(h = 0) \vee (\tau > d)$ **then**
            Drop packet π
        else
            Generate random number $r \in [0, 1]$
            if $r < X_6$ **then**
                Decrement h such that $h' = h - 1$
                Enqueue packet π for future rebroadcast by node R such that $Q' = Q \cup \{\pi\}$
            else
                Drop packet π at R
            end if
        end if
    end if
end if

A.2 Implicit Geographic Forwarding

Algorithms 8 and 9 respectively define the behaviour of nodes acting as candidate packet relays, and nodes attempting to send packets, under the IGF protocol [120].

Algorithm 8 : Implicit Geographic Forwarding (relay)

Node N_i is in state $IDLE$
Node N_i has CTS threshold angle, X_7
Node N_i has state timeout base, X_8
Node N_i maintains packet broadcast queue, Ψ
while $|\Psi| = 0$ **do**
  Listen for incoming RTS packet, ρ
  if RTS packet ρ is received at time τ_{RTS} **then**
    Extract sender of RTS packet ρ, $S = SENDER(\rho)$
    Extract destination of data packet π defined in RTS packet ρ, $D = DEST(\rho)$
    Calculate angle $\theta = \angle DSN_i$
    if $\theta < X_7$ **then**
      Extract IGF sequence number $\omega = SEQ(\rho)$
      Create CTS packet σ_i containing ω and θ such that $(SEQ(\sigma_i) = \omega) \wedge (ANGLE(\sigma_i) = \theta)$
      Broadcast CTS packet σ_i
      Node N_i switches to state $DATA_WAIT$
      while $\tau < (\tau_{RTS} + 2X_8)$ **do**
        Listen for incoming DATA packet, π
        if π is received, and $(SEQ(\pi) = \omega) \wedge (RELAY(\pi) = N_i)$ **then**
          Create ACK packet α containing ω, such that $SEQ(\alpha) = \omega$
          Broadcast ACK packet α
          if $DEST(\pi) = N_i$ **then**
            Consume packet π at destination node $D = N_i$
          else
            Enqueue packet π for future rebroadcast by node N_i such that $\Psi' = \Psi \cup \{\pi\}$
          end if
          Node S returns to state $IDLE$
          End current iteration of outermost WHILE loop
        else
          Ignore DATA packet π
        end if
      end while
      Node N_i returns to state $IDLE$
      End current iteration of outermost WHILE loop
    else
      Ignore RTS packet ρ
      End current iteration of outermost WHILE loop
    end if
  end if
end while

Algorithm 9 : Implicit Geographic Forwarding (sender)

Node S is in state $IDLE$
Node S has CTS threshold angle, X_7
Node S has state timeout base, X_8
Node S maintains packet broadcast queue, Π
while $|\Pi| > 0$ **do**
  DATA packet π is removed from head of broadcast queue Π at time τ_{START} such that $\Pi' = \Pi \setminus \{\pi\}$
  DATA packet π header defines delivery deadline, d
  DATA packet π header defines destination node, D
  DATA packet π header stores selected relay node, R, initially undefined
  Set node S relay candidate CTS packet set, $\varsigma = \emptyset$
  if $(\tau_{START} > d)$ **then**
    Drop packet π
  else
    Generate unique sequence number, ω
    Assign sequence number ω to packet π header such that $SEQ(\pi') = \omega$
    Broadcast RTS packet ρ_π for DATA packet π
    Node S switches to state CTS_WAIT
    while $\tau < (\tau_{START} + X_8)$ **do**
      Listen for incoming CTS packets, σ, from neighbours $N_i \in N$
      if σ is received, and $(SEQ(\sigma) = \omega) \wedge (ANGLE(\sigma) \leq X_7)$ **then**
        Store CTS packet σ such that $\varsigma' = \varsigma \cup \{\sigma\}$
      else
        Ignore CTS packet σ
      end if
    end while
    if $|\varsigma| = 0$ **then**
      Requeue packet π such that $\Pi' = \Pi \cup \{\pi\}$
      Node S returns to state $IDLE$
      End current iteration of outermost WHILE loop
    else
      Select CTS packet $\sigma_{BEST} \in \varsigma$ with the lowest value obtained by $ANGLE(\sigma_i)$ for all $\sigma_i \in \varsigma$
      Select relay node $N_i = SENDER(\sigma_{BEST})$
      Assign selected relay node N_i to DATA packet π such that $RELAY(\pi') = N_i$
      Broadcast DATA packet π taking time δ
      Node S switches to state ACK_WAIT
      while $\tau < (\tau_{START} + X_8 + \delta + X_8)$ **do**
        Listen for incoming ACK packet, α
        if α is received, and $SEQ(\alpha) = \omega$ **then**
          Node S returns to state $IDLE$
          End current iteration of outermost WHILE loop
        else
          Ignore ACK packet α
        end if
      end while
      Inform application at node S that packet π was broadcast but receipt was not acknowledged
      Node S returns to state $IDLE$
      End current iteration of outermost WHILE loop
    end if
  end if
end while