EFFICIENT INFORMATION ACCESS IN DATA-INTENSIVE SENSOR NETWORKS

by

Divyasheel Sharma

B.E., Thapar University, 2000

M.S., Clarkson University, 2003

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Divyasheel Sharma

It was defended on

October 14th 2010

and approved by

Vladimir I. Zadorozhny, School of Information Sciences

Marek J. Druzdzel, School of Information Sciences

Prashant Krishnamurthy, School of Information Sciences

Kirk Pruhs, Departmanet of Computer Science

Suman Nath, Microsoft Research

Dissertation Director: Vladimir I. Zadorozhny, School of Information Sciences

EFFICIENT INFORMATION ACCESS IN DATA-INTENSIVE SENSOR NETWORKS

Divyasheel Sharma, PhD

University of Pittsburgh, 2010

Recent advances in wireless communications and microelectronics have enabled wide deployment of smart sensor networks. Such networks naturally apply to a broad range of applications that involve system monitoring and information tracking (e.g., fine-grained weather/environmental monitoring, structural health monitoring, urban-scale traffic or parking monitoring, gunshot detection, monitoring volcanic eruptions, measuring rate of melting glaciers, forest fire detection, emergency medical care, disaster response, airport security infrastructure, monitoring of children in metropolitan areas, product transition in warehouse networks etc.).

Meanwhile, existing wireless sensor networks (WSNs) perform poorly when the applications have high bandwidth needs for data transmission and stringent delay constraints against the network communication. Such requirements are common for *Data Intensive Sensor Networks (DISNs)* implementing *Mission-Critical Monitoring applications (MCM applications).*We propose to enhance existing wireless network standards with flexible query optimization strategies that take into account network constraints and application-specific data delivery patterns in order to meet high performance requirements of MCM applications.

In this respect, this dissertation has two major contributions: First, we have developed an algebraic framework called *Data Transmission Algebra* (DTA) for collision-aware concurrent data transmissions. Here, we have merged the serialization concept from the databases with the knowledge of wireless network characteristics. We have developed an optimizer that uses the DTA framework, and generates an optimal data transmission schedule with respect to

latency, throughput, and energy usage. We have extended the DTA framework to handle location-based trust and sensor mobility. We improved DTA scalability with *Whirlpool* data delivery mechanism, which takes advantage of partitioning of the network. Second, we propose relaxed optimization strategy and develop an adaptive approach to deliver data in data-intensive wireless sensor networks. In particular, we have shown that local actions at nodes help network to adapt in worse network conditions and perform better. We show that local decisions at the nodes can converge towards desirable global network properties e.g., high packet success ratio for the network. We have also developed a network monitoring tool to assess the state and dynamic convergence of the WSN, and force it towards better performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to acknowledge numerous people who have contributed directly or indirectly towards my thesis. My advisor, Vladimir Zadorozhny, has been an inspiration. He has provided continuous support and mentorship throughout my Ph.D. I cannot thank him enough. Likewise, Marek Druzdzel, Prashant Krishnamurthy and Panos Chrysanthis have been encouraging, extremely supportive and helpful during this Ph.D.

My friends, especially, Sunny Bedi and his family, Ratna Jain and Samvith Srinivas made living in Pittsburgh enjoyable. US was more fun because of Anand Kapur and Aashish Paruthi. I thank them for it. Marek's lab also gave me a number of friends: node (Adam), noob (Mark), snake (Tomek), snowman (Peter Sutovsky), tinus (Martijn) and Andrii. I also had great time with my CMU friends, especially, Anubhav Gupta, Niraj Tolia and Ajay Prasad. I will always cherish great times in Pittsburgh with Sharad Pathak, Amar Pegu, Poonam, Sudhakar Alapati, Shalini-Shringi, Tarun Sonkhya, Rama Damerla, Pushkar Raj Deshmukh, Vishal Dhiman, Surbhi Aggarwal, Pratik Patel, Manas Aggarwal and Dhivyansh Shah. I hope to keep in touch, remain friends and share my life with them.

My family: my father, mother and brother, always supported me. It is only with their love and encouragement that I could finish these studies. Without my father's guidance, this dissertation would not exist. *Excelsior*, he would tell me to remember, always. It was his vision for me to go for higher studies. Unfortunately, he is not here any more to see the fruit of his guidance, support and love. My heartfelt gratitude is to him. This thesis is dedicated to him, with love.

1.0 INTRODUCTION

1.1 MOTIVATION

Recent advances in wireless communications and microelectronics have enabled wide deployment of smart sensor networks. Such networks naturally apply to a broad range of applications that involve system monitoring and information tracking (e.g., airport security infrastructure, monitoring of children in metropolitan areas, product transition in warehouse networks, fine-grained weather/environmental measurements, etc.). Meanwhile, existing wireless sensor networks (WSNs) perform poorly when the applications have high bandwidth needs for data transmission and stringent delay constraints against the network communication. Such requirements are common for *Data Intensive Sensor Networks (DISNs)* implementing *Mission-Critical Monitoring applications (MCM applications)*. As an example of MCM application consider the task of Structural Health Monitoring (SHM) [36], [15] concerned with monitoring the integrity of civil and military structures in order to reduce ownership costs, improve operational lifetime, and protect human life. Another example is a large team of cooperative mobile robots that can be considered to be a wireless network composed of a number of mobile nodes, most of which are power-constrained. Such mobile robots can be deployed in conjunction with stationary sensor nodes to acquire and process data for surveillance and tracking, environmental monitoring for highly sensitive areas, or execute search and rescue operations. Because a failure of the monitoring system is often catastrophic, the associated costs are very high.

The MCM applications have stringent requirements for efficient mechanisms for querying sensor data and delivering the query result. The amount of data collected from all relevant sensors may be quite large and will require very high data transmission rates to satisfy time

constraints. It implies, in particular, in WSNs, that excessive packet collisions can lead to packet losses and retransmissions resulting in significant energy costs and latency. It was reported in [15] that the successful packet delivery ratio in 802.15.4 networks can drop from 95% to 55% as the load increases from 1 packet/sec to 10 packets/sec. Meanwhile, it is common for sensors in the SHM system to generate 6-8 packets/sec of vibration data. The residence time for a packet in a medium-scale multi-hop WSN could be tens of seconds, which is unacceptable for critical monitoring applications. In addition, considerable network variations and limitations on sensor node resources like battery power imply that excessive transmissions in response to monitoring queries can lead to premature network death. At the same time, the requirement of MCM applications must be met despite the random variations in the network characteristics.

A major reason for above problems occurring is that general purpose WSNs with a layered design of the network protocol stack assumed that each layer operates *independently of the individual network constraints and applications* [24]. It results in poor performance for wireless networks especially when the applications have high bandwidth needs and stringent delay constraints. This calls for novel approaches in designing *special purpose* WSNs that support application-driven data interrogation patterns and optimization across multiple network layers. Our major contribution of this research consists in developing methods and techniques for efficient utilization of *special-purpose* WSNs for the task of mission-critical monitoring. We believe that WSNs explicitly designed for the critical monitoring applications can gain significant performance improvements. We propose to enhance existing wireless network standards with flexible query optimization strategies that take into account network constraints and application-specific data delivery patterns in order to meet high performance requirements of MCM applications. We make two major contributions here: First, we have developed an algebraic framework called *Data Transmission Algebra* (DTA) for collision-aware concurrent data transmissions. DTA is based on an algebraic query optimization that utilizes information about how the lower network layers operate while processing critical monitoring queries. As such, this research fuses techniques from different areas of databases and networking. It is important to note that our technique can utilize existing wireless network standards without introducing extra control and processing overheads or

disruption of network protocol hierarchy. In order to prove high utility of our approach, we apply it under real MCM application loads and requirements. They include *non-intrusive* Structural Health Monitoring, - a procedure where the natural dynamics of structure are observed for changes that indicate damage or instability [54], [63], [22]. We have extended the DTA framework to handle location-based trust and for networks with mobile nodes. We improved DTA scalability with *Whirlpool* data delivery mechanism, which takes advantage of partitioning of the network. Second, we propose relaxed optimization strategy and develop an adaptive, decentralized approach to deliver data in data-intensive wireless sensor networks. In particular, we have shown that local actions at nodes help network to adapt in worse network conditions and perform better. We show that local decisions at the nodes can converge towards desirable global network properties e.g., high packet success ratio for the network. We have also developed a network monitoring tool to assess the state and dynamic convergence of a WSN, and force it towards better performance.

1.2 OBJECTIVE

We believe that the results from this research will be essential in the deployment of robust special-purpose WSNs for MCM applications. The practical question that will be answered is:

"What optimization strategies should be developed to provide the most efficient utilization of wireless sensor networks in critical monitoring environments?"

This research combines both theoretical investigation and experimental evaluation. Specifically, through this thesis, we will provide answers for the following fundamental questions:

1. *How does the wireless network behavior affect the data access and query processing algorithms for critical monitoring applications?*
2. *How can wireless query execution benefit from the synergy of application-driven query optimization and lower network protocol layers?*
3. *How can the proposed optimization strategies be tuned to meet specific performance requirements of mission-critical monitoring applications?*

4. *How can we develop more scalable approaches that adapt to the varying conditions of the wireless network?*

1.3 ORGANIZATION OF THE DISSERTATION

The rest of this dissertation is organized as follows. In the next chapter, we introduce background, related work and give an overview of the proposed research. In Chapter 3, we present our basic algebraic optimization framework and its theory. In Chapter 4, we consider practical implementation strategies for our algebraic framework, how we can integrate this framework to efficient network interrogation strategies and justify our claims with experimental analysis. In Chapter 5, we discuss trust-based routing, which extends over our algebraic framework. In Chapter 6, we develop relaxed optimization strategies for efficient data delivery that adapt to the uncertain network conditions. We conclude with a discussion of our research.

2.0 BACKGROUND AND RELATED WORK

This research investigates application/query performance in data-intensive wireless sensor networks (DISNs), although it is expected that the results are of general applicability to resource constrained wireless environments.

2.1 INTRODUCTION TO COMMUNICATION IN SENSOR NETWORKS

Consider a wireless sensor network deployed in order to monitor structural integrity. An example query over this network could request vibration data over a certain period of time. Answering this query would result in a tree-like data delivery pattern (Figure 1). The transmissions between sensors are *ad hoc* dependent on the query and require the use of a medium access control (MAC) layer to handle transmissions on the same medium and a routing algorithm that enables the nodes to select the right neighbor to transmit data.

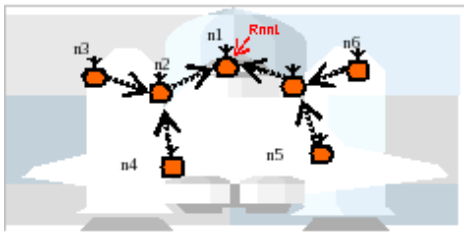

Figure 1: An example of a query tree.

Popular wireless MAC layer technologies are the IEEE 802.11 standard for wireless local

area networks [7] and the IEEE 802.15 standards for wireless personal area networks [6]. IEEE 802.11 has several flavors, of which the 802.11b (maximum data rate of 11 Mbps) and 802.11g (maximum data rate of 54 Mbps) operate at 2.4 GHz and 802.11a (also 54 Mbps) operates at around 5 GHz [67]. 802.11 devices typically use a transmission power of 250 mW (24 dBm) (although other values, both lower and higher, are possible). For low power and low data rate sensor networks, the 802.15.4 standard appears to be suitable. The data rates with 802.15.4 are 20, 40 or 250 kbps in the 868, 915 or 2400 MHz bands respectively. The transmission power with 802.15.4 is also expected to be very low (1 – 2 mW or 0 – 3.6 dBm). The Zigbee industry standard [5] is developing network and application layer protocols to operate over 802.15.4.

The operation of the MAC layer is slightly different depending on the technology. One issue, which is common for all MAC layer protocols, is proper handling of packet collisions. If we assume that all sensor nodes use the same frequency band for transmission, two transmissions that overlap will get corrupted (collide) if the sensor nodes involved in transmission or reception are in the same *collision domain*. More formally, any two communicating nodes *ni* and *nj* specify a collision domain CD(*ni,nj*) defined as the union of the transmission ranges of *ni* and *nj*. Consider two nodes *n1* and *n2* that wish to communicate (Figure 2). In Figure 2, nodes *n1*, *n2*, *n3*, and *n4* are in the same collision domain. This implies that when *n1* and *n2* are communicating, *n3* and *n4* cannot participate in any communications. Moreover, even though node *n5* is outside the collision domain, if it sends a packet to *n3* at the same time that *n1* is sending a packet to *n2*; these two transmissions will collide at *n3*. The information in both packets will be lost. Similarly, *n4* and *n6* cannot communicate when *n1* and *n2* are communicating.

We can use Figure 2 to illustrate how collisions are handled in a typical wireless network such as IEEE 802.11 using carrier sense multiple-access with collision avoidance (CSCMA-CA) [7], [17]. In general, before starting a transmission, nodes must sense the channel for a predetermined amount of time (waiting time). If the channel is busy, the nodes wait for the predetermined amount of time after the channel becomes free. In addition, nodes back-off for a random time to avoid the possibility that two or more nodes transmit at the same time after the waiting period. For this entire period, the node must sense the channel and this

consumes energy. Each packet also needs to be acknowledged by the receiver since wireless channels are unreliable.

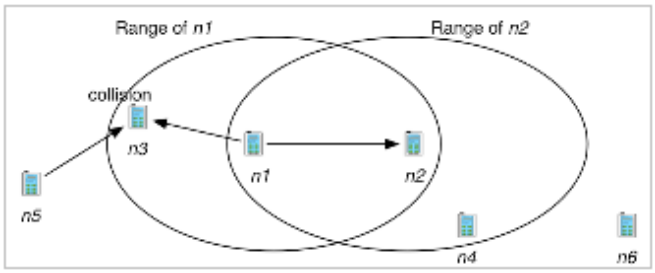

Figure 2: Collision domain of two communicating nodes.

If only basic carrier-sensing is employed, it can lead to the *hidden terminal* problem [17]. Consider Figure 2 again. Node *n4* is outside the range of *n1*. When *n1* is transmitting a packet to *n2*, *n4* cannot sense the energy on the air and may start a transmission, either to *n2* or *n6*. This transmission will collide with the transmission from *n1* at node *n2*. The node *n1* is hidden from *n4* and vice versa. In 802.11, an additional mechanism exists to avoid the hidden terminal problem [7], [78]. In order for a sensor node *n1* to communicate with sensor node *n2*, *n1* needs to send first a request for transmission packet (*Rtx*) to *n2*, so that all other nodes in its transmission range (*n3* in Figure 2) become aware of the communication and remain silent until *n1* ends the transmission. Sensor *n2* replies to *n1* with a confirmation packet (*Ctx*), so that the nodes in its transmission range (*n4* in Figure 2) also become aware of the communication and avoid any transmission until the end of the current transmission. The use of the request and clear messages adds to the overhead of the system. So, in 802.15.4, messages such as Rtx and Ctx are not employed. Consequently, the number of collisions can increase significantly if the load on the network increases. Collisions can be avoided in 802.15.4 to some extent by using *contention free periods* [6]. In 802.15.4, devices can be fully functional or have reduced functionality. A fully functional device can participate in multi-hop routing and can also act as a coordinator for a subset of nodes creating its own local network. In this case, the coordinator employs a beacon mode that uses a superframe structure to indicate a *Contention Access Period* (CAP) that uses CSMA and an optional

contention free period (CFP) that specifies *Guaranteed Time Slots* (GTSs). The coordinator allocates the GTS to some sensor nodes. GTS, however, can only be used for indirect data transmission where sensor nodes first send data frames to coordinator.

2.2 RELATED RESEARCH

This research fuses the areas of database and networking. This section discusses major projects and works in both these areas of query processing in wireless sensor networks.

Several techniques have been proposed to alleviate the problem of limited resources at the network level such as energy-efficient routing, clustering and transmission scheduling [29], [72], [30], [13]. Sensor networks can be viewed as a highly distributed database [4], [35], [70] that requires novel resource-constrained and network-aware query execution strategies. Sensor database research has also investigated special query strategies, such as sampling [47], prediction [23], approximation [16], and in-network aggregation [35], [12], [48], [62]. Since a tree query topology is susceptible to node and transmission failures, which are common in sensor networks [48], several alternative approaches for data delivery in sensor networks have been proposed recently. Data dissemination schemes like SPIN [31] using flooding technique, interest gradient based Directed Diffusion [21]; clustering based LEACH [30] have been proposed in literature. Synopsis Diffusion [52] proposes a multi-path routing scheme which is more robust than tree topology based TAG [48] to avoid double-counting in sensor readings. As discussed in the previous section, one of the major limiting factors in data delivery is packet collisions in the network. Wave scheduling [66] minimizes packet collisions by carefully scheduling the sensor nodes so that each node can stay idle for most of the time, turning on its radio only at scheduled intervals during which it can receive or send a message. It results in energy savings at the expense of increased message latency and at present, it does not study irregular wave schedules. Another way of eliminating collisions is to create an orthogonal transmission mechanism whereby a central authority, such as a base station allocates specific time slots for nodes to transmit based on reservation or polling that will be similar to time division multiple access (TDMA). This however requires

a centralized mechanism that could be fairly complex to implement, consume significant overhead for signaling and be difficult to implement in a multi-hop scenario. This issue is properly addressed in our framework. Moreover, in the case of TDMA, each node is assigned a time slot in a frame by a central authority and it can transmit only in that time slot in each frame. If it has nothing to send, that time slot is wasted since other nodes cannot access this slot. In contrast to TDMA, our approach assumes that a predefined schedule is sent to a node and it is up to the node to decide how to behave within a set of constraint specified by the schedule. It is not mandatory for every node to follow a schedule, so the scheduling helps where it can, otherwise it utilizes MAC. Finally, our scheduling can be implemented with different degrees of centralization/distribution.

Distributed TDMA scheme, for example, LMAC [32] was implemented as a distributed time slot scheduling algorithm for collision-free communications. Chatterjea et al introduced AI-LMAC [14] that uses captured local data about traffic patterns to modify operations accordingly. The protocol is an extension of the LMAC and adapts to the application requirements. Another distributed TDMA scheduling procedure was proposed in [10]. Its design goal is to permit a mobile to move and then reallocate itself a time slot without involving the entire network. Ali et al. [9] proposed distributed and adaptive TDMA algorithms for multi-hop mobile networks. One concern with this design is that dynamic topology changes may lead to frequent exchanges of control packets that could consume bandwidth and energy resources.

The above distributed TDMA scheduling schemes have considerable control message overhead for building data delivery schedules. Below, we provide a brief overview of other contention and scheduling based functionalities [41] related studies, since, our first contribution in this thesis falls into the scheduling-based category. S-MAC [71] is a contention-based energy saving protocol implemented in the MAC layer of sensor network. It allows neighboring nodes to sleep for long periods and then to wake up, both in a synchronized fashion to avoid possible wastage of idle listening, collisions and retransmissions. Thus, the neighbors conserve energy when the other node is transmitting. However, S-MAC provides no on-demand interaction with the receiver in case if there is a need to communicate between sender and receiver. It uses a static sleep interval, regardless of dynamic wireless environ-

ment and traffic load. Schurgers et al [61] proposed STEM, to save energy that implements a two-radio architecture that lets the data channel sleep until communication is desired. Based on the assistance from the monitoring channel, collisions and retransmission are alleviated. However, in STEM a busy tone has to wakeup a node's entire neighborhood since the intended receiver's identity is not included on the monitoring channel. Vaidya et al [65] tried to improve STEM by introducing a rate estimation scheme on top of it. This work selectively wakes up the data channel of nodes, which have previously involved in communication via RE using an optimal wakeup interval to minimize the energy consumption. Both STEM and rate-estimation take the two-radio architecture for granted. They ignore the complexity of adopting the second radio on a tiny sensor that may result in difficult antenna receivers design and additional energy consumption on the second radio. Sivalingam et al. [64] proposed an EC-MAC (Energy Conserving-Medium Access Control) protocol. Using this protocol a central controller is responsible for reservation and scheduling strategies. The transmission is organized by the controller into frames, and the time slots of the frame serve various purposes. In consequence, collisions are reduced and retransmissions are avoided. Liu et al. [42] proposed Enhanced 802.11. Here, a contention-free schedule is derived from overheard ATIM frames, sent by neighbors, under the assumption of a fully connected network. Each node must receive all transmitted ATIM frames within the network to ensure the complete schedule. The ATIM frame includes information about the number of packets to be transmitted by each node, so every node can calculate the transmission schedule. If the ATIM and acknowledge packets are exchanged, the contention-free schedule can allow the nodes finishing their transmissions early in the data window and then switching to low-power mode in the middle of the data window. This advantage allows the nodes with short transmissions, which is especially common in a high-speed network to stay in low-power longer. With the contention-free schedule, the contention process for ATIM is still required but is eliminated in data transmissions that are normally much larger than an ATIM window size. EC-MAC and Enhanced 802.11 schemes can only operate in an environment where every sensor hears each other while proposed DTA is suitable for fully connected network and multi-hop network also.

Data delivery in wireless networks is probabilistic since the connectivity is the likelihood

of a successful communication [11]. Neighboring nodes have connectivity based on interference, congestion, and other sources of losses. Depleting energy resources and hardware characteristics lead to unreliable estimation of sensors ranges. Thus, routing algorithms for sensor networks should take these factors in consideration and be evaluated along with the lower level estimation mechanisms as well as application level factors under realistic application traffic loads.

A wireless sensor network for Structural Health Monitoring is described in [15] which employs a multi-hop network that continuously collects data and transmits them to a base station. In this chapter, challenges such as synchronization and timely and reliable data delivery are discussed. End-to-end and per hop error recovery, wavelet compression and time-stamping for synchronization are used to address these challenges. This work also identifies some potential problems with low data rate sensor networks and high data rate 802.11 like networks as described below.

There are considerable deficiencies in applying existing wireless sensor networks to the task of mission critical monitoring. An examination of the factors that affect both energy consumption and response time in sensor queries reveals that (a) data transmission *collisions* represent a major source of time and energy waste in wireless communications; (b) unnecessary amounts of *active time* for the sensors, due to lack of synchronization among data transmissions, is another major source of wasted time and energy in wireless sensor networks [15]. IEEE 802.15.4 was designed for quite general-purpose wireless networks providing an acceptable average performance for a wide class of applications. The lack of mechanisms to overcome the hidden terminal problem can result in excessive collisions. The data rates of 802.15.4 are quite low and can increase the latency of data delivery, which can be a big disadvantage for the task of critical monitoring.

Recent study [15] has shown that common MAC protocols can achieve 100% data delivery reliability with a packet rate up to 1 packet/sec per node. The number of collisions can increase significantly if the load on the network increases. Consider a Structural Health Monitoring system as described in [15]. Sensors in such systems can generate up to 20 kilo-samples/s on four channels for a total of 80 kilo-samples/s. Each sample is 16 bits long resulting in data being generated by each sensor at a rate of 1280 kbps (160 bytes/s).

The maximum physical layer packet size in 802.15.4 is 127 bytes of which 16-32 bytes are part of the MAC/PHY headers. Assuming 80 bytes/packet at the PHY layer, two packets are generated every second by each sensor node. Occasionally, packet sizes can be smaller resulting in higher packet rates and increased possibility of collisions with 802.15.4. As reported in [15] the average residence time for 1 packet in a multi-hop network of 10 sensors could be up to 142 secs. When this rate increases to 2 packets/sec per sensor, the network collapses.

The above factors result in considerable under-utilization of the sensor networks exploiting common data delivery techniques. Roughly speaking, only 120 bytes/sec of the network bandwidth is utilized out of the available 3750 bytes/sec. Meanwhile, a typical SHM application generates 200-600 bytes/sec of row data, which introduces an obvious performance bottleneck in existing WSNs for the critical monitoring task.

Figure 3 summarizes the characteristics of Data Intensive Sensor Networks. As the data load increases we observe considerable performance degradation of the key performance parameters of wireless sensornets. Packet success ratio drops due to frequent collisions and retransmissions. The data glut results in the increased time delay and overall energy consumption. After certain load threshold the performance characteristics of traditional WSNs become unacceptable.

2.3 OVERVIEW OF THE DISSERTATION

This dissertation studies the problem of efficient query processing in resource constrained critical wireless sensor environments. Our objective is to develop efficient information access mechanisms for wireless sensor networks under data-intensive wireless network constraints and specific application requirements. We propose a database-driven approach to optimize data transmissions in sensor networks for the task of critical monitoring. Using the optimizer, the potential problems with wireless sensor networks can be significantly alleviated. We also propose decentralized optimization techniques that provide adaptive data delivery based on

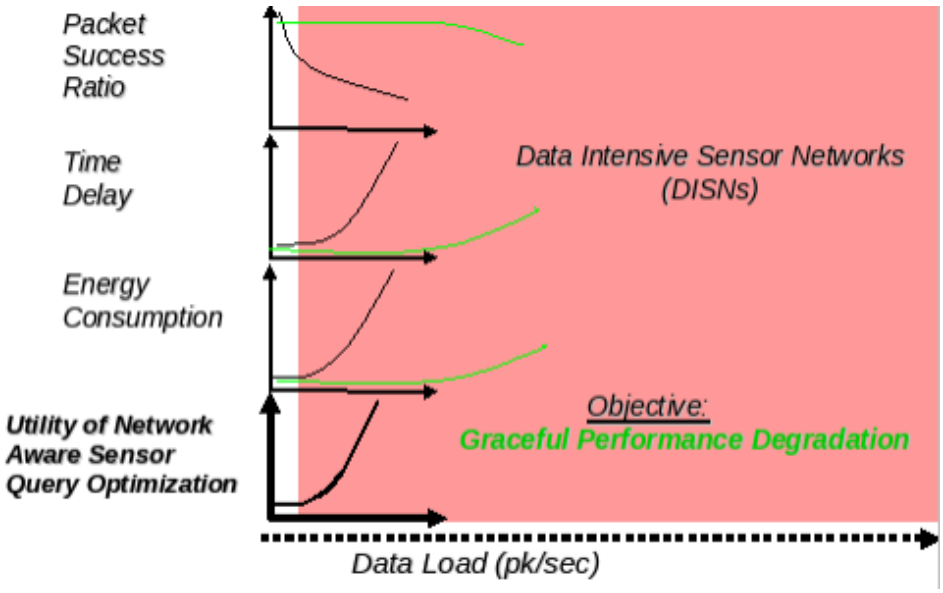

Figure 3: Utility of the Proposed Optimization Approach.

local knowledge that is readily available to a sensor node. This thesis has proceeded in the following stages:

1. Development of a basic cross-layer wireless query optimization framework (DTA).
2. Theoretical study and justification of the DTA framework.
3. Exploring practical implementation strategies for DTA framework.
4. Explored limitations of DTA framework and developed a more scalable and decentralized optimization scheme.

3.0 BASIC CROSS-LAYER OPTIMIZATION FRAMEWORK

In this chapter, we first discuss our basic cross-layer optimization framework. Then, we describe how we can integrate this framework with existing wireless infrastructure. We also describe the theoretical foundations of our framework and analyze soundness and completeness of our approach.

3.1 DATABASE-LIKE QUERY OPTIMIZATION FOR EFFICIENT TRANSMISSION SCHEDULING

Our basic approach consists of integrating the low level wireless network protocols with database-driven algebraic query optimization. A query optimizer should take into consideration the current network topology, the applications' coverage requirements, collision domains of the wireless nodes, etc. in order to generate efficient *query schedules*. Thus, in order to generate the query schedules, we propose to utilize an algebraic framework - *Data Transmission Algebra (DTA)* – that captures the low-level network features along with their constraints and requirements. In this chapter, we introduce DTA that allows the optimizer to generate *collision-aware* query schedules. In chapter 4, we improve DTA's scalability with Whirlpool, and in chapter 5, we extend the DTA framework to handle trust-aware data delivery.

3.1.1 Basic DTA Framework

The basic DTA consists of a set of operations that take transmissions between wireless sensor nodes as input and produce a schedule of transmissions as their result. The simplest DTA schedule is an *elementary transmission ni~nj* denoting a one-hop transmission from sensor node *ni* to node *nj*. Each transmission *ni~nj* is associated with a collision domain CD(*ni*, *nj*) defined as the union of the transmission ranges of the communicating nodes. A transmission schedule is either an elementary transmission or a composition of elementary transmissions using one of the operations of the DTA. The basic DTA includes three operations that combine two transmission schedules A and B:

1. ***o*(A,B)** - a strict order operation, that is, A must be executed before B;
2. ***c*(A,B)** - a non-strict order operation, that is, either A executes before B, or vice versa;
3. ***a*(A,B)** - an overlap operation, that is, A and B can be executed concurrently.

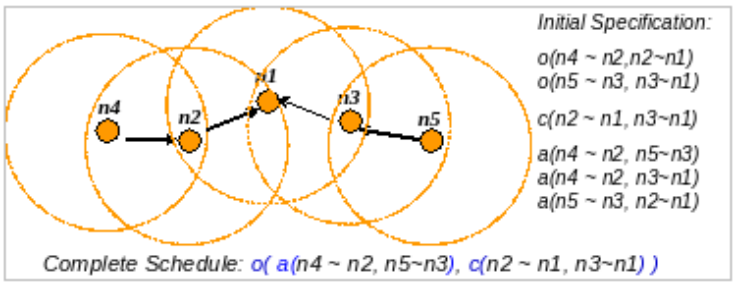

Figure 4: Example of DTA specifications.

For example, consider the query tree in Figure 4 that shows the *initial DTA specification* reflecting basic constraints of the query tree. For instance, operation *o(n4~n2, n2~n1)* specifies that transmission *n2~n1* occurs after *n4~n2* is completed because of the query tree topology. Operation *c(n2~n1, n3~n1)* specifies that there is an order between transmissions *n2~n1* and *n3~n1* since they share the same destination. However, this order is not strict. Operation *a(n4~n2, n5~n3)* specifies that *n4~n2* can be executed concurrently with *n5~n3*, since neither *n3* nor *n5* belongs to CD(*n4,n2*), and neither *n4* nor *n2* are in CD(*n5,n3*). Figure 4 also shows an example of a complete schedule that involves all elementary transmissions of the query tree.

3.1.2 Query Scheduling using DTA

Each operation of the initial specifications defines a simple transmission schedule consisting of two elementary transmissions. The DTA introduces a set of transformation rules that can be used to generate more complex schedules from the initial specification. Applying the DTA rules we generated schedules stage by stage starting from initial schedules with two elementary transmissions (stage 1). Stage 2 generates schedules with 3 transmissions; stage 3 generates schedules with 4 transmissions. Last stage generates complete schedule that includes all elementary transmissions of the query tree. An example of DTA transformation rules is the following.

Rule1: o(A,B), o(C,D), a(A,C) $\rightarrow$ *o(a(A,C), B)*

Rule2: o(a(A,C), B), c(B,D) $\rightarrow$ *o(a(A,C), c(B,D))*

These rules apply towards generating more complex schedules from the initial specification in Figure 4.

Apply Rule1: o(n4~n2,n2~n1), o(n5~n3,n3~n1), a(n4~n2,n5~n3) $\rightarrow$ *o(a(n4~n2n5~n3), n2~n1)*

None of the simple or complex transmission schedules considered so far include all elementary transmissions of the query tree, so we call them partial schedules. Our goal is to generate DTA expressions for complete schedules.

Apply Rule2: o(a(n4~n2n5~n3), n2~n1) $\rightarrow$ *o(a(n4~n2n5~n3), c(n2~n1,n3~n1))*

More DTA transformation rules are given in DTA theory section. Below we introduce a cost model for optimizing data transmissions in order to generate complete and efficient schedules.

Figure 5 shows simple cost estimation expressions for each of the DTA expressions. In this case, the cost corresponds to the execution time associated with a particular schedule. For example, the execution time of elementary transmission *ni~nj* consists of local processing times T_p at nodes n_i and n_j plus the time T_{tx} required for transmitting data from n_i to n_j.

The execution time of strict order of schedules A and B is the sum of execution times of A and B. For overlapping schedules A and B, the execution time would be the maximum of the execution times of A and B. Finally, the execution time of the choice between A and B

Schedule	Cost
n_i~n_j	$T_p(n_i)+ T_p(n_i \sim n_j)+ T_p(n_j)$
o(A,B)	cost(A) + cost(B)
a(A,B)	max(cost(A), cost(B))
c(A,B)	cost(A) + cost(B)-T_f

Figure 5: Estimating Time Costs of Schedules.

is the same as the execution time of the strict order minus a predefined time factor T_f. T_f indicates that in general, the optimizer prefers the choice operation over strict order, since the latter restricts flexibility of the optimizer in query scheduling. We ignore propagation times as they are negligible in this case. The optimizer chooses the schedule with minimum cost from a set of all possible complete schedules.

3.1.3 Integration of Basic Optimization Framework with Wireless Network Infrastructure

We design a query optimizer that executes at the coordinator node (e.g., base station) and selects the data transmission schedule. The schedule implements a monitoring query with optimal response time and energy consumption. A coordinator announces the generated schedule to relevant wireless nodes broadcasting beacons and modified superframe structures to the nodes. The superframe structure specifies mandatory transmission and reception time slots for each node. Note that, in addition to collision handling, DTA can be applied to exploit other factors which are currently not supported at the MAC layer. They include (a) the ability to have fully functional point coordinators that can assign Guaranteed Time Slots (GTS) to other nodes and (b) the availability of multiple frequency channels (3 in 802.11b,g, 8 in 802.11a and up to 20 in 802.15.4) to support concurrent transmissions.

Since the query delivery is tree-like, the sink node can act as the coordinator and assign GTS to the leaf nodes that are collecting the information based on the DTA. For example, consider the query tree shown in Figure 6a. Using DTA, the optimizer generates the following schedules c(*n1~n4*, c(*n2~n4, n3~n4*)) for delivery to node *n4* and c(*n4~n6*, *n5~n6*) for

delivery to node *n6*. The above two schedules can be executed concurrently. Then node *n4* can be the point coordinator and assign GTS to nodes *n1*, *n2*, *and n3*. Recognizing that concurrent transmissions are possible, *n6* can assign the first GTS to *n5* while *n4* is collecting information from its leaf nodes as shown in Figure 6a.

Another aspect that DTA can exploit is the availability of multiple frequency channels over which concurrent transmissions are possible. Currently, there are no mechanisms for ad hoc networks to operate using multiple frequencies because nodes are not aware of what frequencies to use for which connections and when to use them. If the DTA schedules generated also indicate the frequency channels to be used, concurrent transmissions are enabled where none existed previously. For example, in Figure 6 (Left), we could have a(*n1~ n4, a(n2~n4, n3~n4)*). Node *n4* must be capable of receiving communications on multiple frequency channels, which may not be possible with existing technology. Transmissions in the same collision domain with different source and destination nodes can exploit concurrency with different frequency channels even with today's technology.

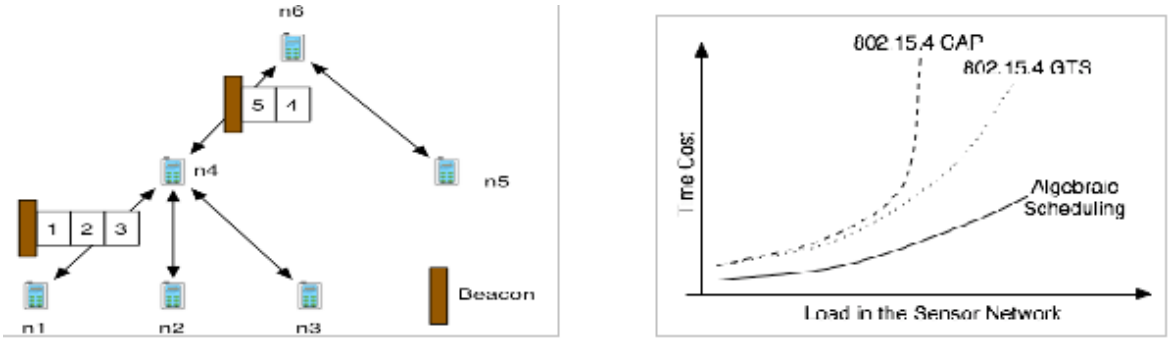

Figure 6: Left: Using GTS with DTA, Right: Time Cost Relationships between Different Scheduling Schemes.

3.2 DTA THEORY

In this section we introduce a DTA theory consisting of the DTA signature and DTA inference rules [27]. The DTA signature specifies basic DTA syntax, while DTA inference rules represent transformations of the well-formed DTA terms. The DTA signature specification is presented in Figure 7. It includes a set of sorts together with operations defined on them. The DTA signature includes two sorts ***Node*** and ***Schedule***. Elementary transmission (denoted ~) is a DTA operation that takes two nodes as input and outputs a schedule. The rest of the DTA operations (*o, c, a*) map two input schedules to an output schedule.

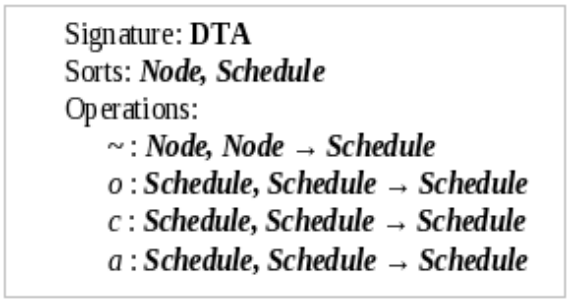

Figure 7: DTA Signature.

In order to introduce DTA inference rules we extend the basic DTA signature with a secondary operation *subs,* which for a given DTA schedule returns all its sub-schedules. The *subs* operation is specified as follows:

subs: ***Schedule***→**P*(Schedule)*,**

where **P*(Schedule)*** denotes the power set of schedules. The following equations complete the specification of *subs*:

$subs(X \sim Y) = \{ X \sim Y \}$.

subs(comp(S1,S2)) =

$\{comp(S1,S2)\} \bigcup subs(S1) \bigcup subs(S2)$,

where *comp* denotes any of DTA operations *o, c,* or *a*. DTA inference rules are repre-

sented in Figure 8. A DTA inference rule *Premise* → *Conclusion* reflects the fact that there is a one step inference from *Premise* to *Conclusion.* For example, using rule 1 (*order introduction*) we can infer a strict order of two elementary transmission if the destination node of the first transmission is also a source node of the second transmission. Rule 5 generates a strict order of DTA schedules X and S if there exists a sub-schedule S_i of the schedule S such that *o(X,*S_i*)* can be generated by the DTA rules. In order to infer *a(X, S)*, we should be able to infer *a(X,*S_i*)* for all sub-schedules S_i of the schedule S.

Definition 3.3.1 (DTA inferability): A DTA schedule t is *inferable* from a set of DTA schedules S (denoted $S \vdash t$) iff either $t \in S$, or t can be generated from S via finite applications of the DTA inference rules.

Example 1. Consider the following set of DTA schedules:

S = {*n4~n2, n2~n1, n5~n3, n3~n1, a(n4 ~ n2, n5~n3), a(n4 ~ n2, n3~n1)*},

which is a subset of the initial DTA specification from Figure 8. We can infer from S the following schedule:

a(n4 ~ n2, o(n5~n3, n3~n1)),

using rules 1 and 8:

(n5~n3), (n3~n1)$\rightarrow_{rule\ 1}$ o(*n5~n3, n3~n1*), *a(n4 ~ n2, n5~n3), a(n4 ~ n2, n3~n1)* $\rightarrow_{rule\ 8}$ *a(n4 ~ n2, o(n5~n3, n3~n1))*.

1.	Order introduction	N1~N2, N2~N3 → o(N1~N2, N2~N3)
2.	Order transitivity	o(X,Z), o(Z,Y) → o(X,Y)
3.	Choice commutativity	c(X,Y) ↔ c(Y,X)
4.	Overlap commutativity	a(X,Y) ↔ a(Y,X)
5.	Left sub-schedule order	($\exists S_i \in subs(S)$, o(X, S_i)) → o(X,S)
6.	Right sub-schedule order	($\exists S_i \in subs(S)$, o(S_i, X)) → o(S,X)
7.	Sub-schedule choice	($\exists S_i \in subs(S)$, c(X, S_i)) → c(X,S)
8.	Sub-schedule overlap	($\forall S_i \in subs(S)$, a(X, S_i)) → a(X,S)

Figure 8: Basic DTA Inference Rules.

3.2.1 DTA Semantics

We provide a logic-based specification of DTA semantics using Prolog-like Horn clauses [25]. A predicate looks as follows: *p(t1,t2,...,tn),* where p is a predicate name of arity n, each *ti* is a *term*, and *(t1,t2,...,tn)* is a tuple. A term is a constant or a variable, or a complex term constructed using function symbols. A name starting with a capital letter signifies a variable. A *rule* is a statement of the form

p :- q1, q2,..., qn,

where p and *qi* are predicates, p is the rule head, and the conjunction *q1, q2 ,..., qn* is the rule body. A rule with an empty body is called a fact. A rule may be used to define the predicate p, so that p holds whenever *q1, ...,qn* all hold. For example, the following rule defines that X is a grandparent of Y if X is a parent of Z and Z is a parent of *Y:*

grandparent(X,Y) :- parent(X,Z), parent(Z,Y).

Below we introduce the predicates used in the logical specification of DTA semantics. The predicates are grouped into *environment constraints*, which reflect basic properties of wireless transmission medium, and *query constraints*, which reflect data transmission patterns imposed by the query semantics. Finally, we use environment and query constraints to define the semantic validity of DTA schedules.

3.2.1.1 Environment Constraints The environment constraints reflect an actual sensor network with wireless nodes communicating via data transmissions. The transmissions can be either elementary (one-hop), or complex ones (consisting of several elementary transmissions). The following predicates specify the environment constraints:

wirelessNode(X). This predicate specifies that X is a wireless sensor node.

distance(X1,X2,D). This predicate specifies that D is the distance between wireless nodes *X1* and *X2.*

range(X,R). This predicate specifies that wireless node X can transmit in range R.

in_range(X1,X2). This predicate is true if wireless node *X1* is within transmission range of node *X2.* The following rule defines the *in_range* predicate:

in_range(X1,X2) :- range(X2,Range),

distance(X1,X2,Dist),

Range >= Dist.

reachable(X1,X2). This predicate is true if nodes *X1* and *X2* are within transmission ranges of each other:

reachable(X1,X2) :- in_range(X1,X2),

in_range(X2,X1).

starts(S,T). This predicate specifies that data transmission *S* (elementary or complex one) starts at time moment *T*.

ends(S,T). This predicate specifies that data transmission *S* ends at time moment *T*.

time_overlap(S1,S2). This predicate is true if transmissions *S1* and *S2* overlap in time. The following rule defines the *time_overlap* predicate we use semicolon to denote disjunction:

time_overlap(S1,S2) :- starts(S1,ST1), starts(S2,ST2),

ends(S1,ET1), ends(S2,ET2),

((ST2 = ET1, ET2 = ST1);

(ST1 = ET2, ET1 = ST2)).

member(X~Y, S). This predicate is true if the elementary transmission *X~Y* is included in a complex transmission *S.*

in_cd(X, X1~Y1). This predicate is true if node *X* is located in the collision domain of elementary transmission *X1~X2:*

in_cd(X, X1~Y1) :- in_range(X,X1); in_range(X,Y1).

can_collide(S1,S2). This predicate specifies that concurrent execution of transmissions *S1* and *S2* may result in collisions. The first rule considers the case when both *S1* and *S2* are elementary transmissions. The second rule deals with complex transmissions.

can_collide(X1~Y1,X2~Y2) :-

in_cd(Y1,X2~Y2); in_cd(X1,X2~Y2);

in_cd(Y2,X1~Y1); in_cd(X2,X1~Y1).

can_collide(S1,S2) :- member(CS1,S1),

member(CS2,S2),

can_collide(CS1,CS2).

collide(S1,S2). This predicate specifies that concurrent execution of transmissions *S1* and *S2* results in collisions.

collide(S1,S2) :-

can_collide(S1,S2),

time_overlap(S1,S2).

3.2.1.2 Query Constraints *strict_precede(S1,S2).* This predicate states that query semantics require schedule *S1* to be executed before schedule *S2*:

strict_precede(S1, S2) :- ends(S1, ET1),

starts(S2, ST2),

ST2 > ET1.

precede(S1,S2). This predicate states that schedules *S1* and *S2* must be executed in an order (either *S1* follows *S2,* or *S2* follows *S1*):

precede(S1,S2) :- strict_precede(S1,S2);

strict_precede(S2,S1).

no_conflict(S1,S2). This predicate states that transmissions *S1* and *S2* can be executed concurrently without violating any query-imposed orders and without risk of collisions:

no_conflict(S1,S2) :- not precede(S1,S2),

not can_collide(S1,S2).

3.2.1.3 Validity of DTA Schedules At this point we are ready to specify semantics of the DTA schedules in terms of the predicates introduced above. First, we should define a mapping of the DTA terms into our semantic domain. We assume an identity mapping function between ***Node*** sort of DTA signature and wireless nodes. We also assume that any DTA term X~Y will map in elementary transmission X~Y. DTA terms *comp(S1,S2),* where *comp* denotes one of the DTA operations *o, c,* or *a* will map in the complex transmission that includes all elementary transmissions $et_i \in subs(comp(S1,S2))$. We will represent a data transmission as a list of its elementary transmissions $[et_1, \ldots, et_n]$. The mapping is defined via the following *map* predicate:

map(X~Y,[X~Y]).

map(comp(S1,S2), Result) :-

map(S1,MS1), map(S2,MS2),

append(MS1,MS2,Result).

The first *map* rule specifies that any elementary transmission *X~Y* schedule is mapped to an actual data transmission represented as a list *[X~Y]* whose only member is the given elementary transmission. The second rule applies the *map* predicate recursively to the components of a complex schedule and generates the resulting complex transmission as a list of elementary transmissions appending the results of the component mappings. Predicate *append(L1,L2,R)* is true if list *R* is a concatenation of the lists *L1* and *L2*.

Example 2. Consider the DTA schedule inferred in Example 1: *a(n4 ~ n2, o(n5~n3, n3~n1))*. Using the *map* predicate it will be mapped in the following list of elementary transmissions: *[n4~n2, n5~n3, n3~n1]*.

The following *valid* predicate specifies semantics for each of the DTA operations:

valid(X~Y) :- wirelessNode(X), wirelessNode(Y),

reachable(X,Y).

valid(o(S1,S2)) :- valid(S1), valid(S2),

map(S1,MS1), map(S2,MS2),

strict_ precede(MS1,MS2).

valid(c(S1,S2)) :- valid(S1), valid(S2),

map(S1,MS1), map(S2,MS2),

precede(MS1,MS2).

valid(a(S1,S2)) :- valid(S1), valid(S2),

map(S1,MS1), map(S2,MS2),

no_conflict(MS1, MS2).

The definition of the *valid* predicate consists of four rules with one rule per DTA operation. Elementary transmission *X~Y* is valid if the participating wireless nodes *X* and *Y* are reachable from each other, i.e., *X* and *Y* are in each other's transmission ranges. Strict order *o(S1,S2)* is valid if both *S1* and *S2* are valid schedules, *S1* is executed before *S2 (*imposed by *strict_precede* predicate). Validity of *c(S1,S2)* and *a(S1,S2)* is defined using *precede* and *no_conflict* predicates.

Using *valid* predicate we can define a DTA semantic entailment, or logical implication relation:

Definition 3.3.2 (DTA semantic implication): DTA term t is semantically implied by a set of DTA schedules S (denoted $S \vdash t$) if t is valid whenever all $t_i \in S$ are valid.

3.2.2 soundness and completeness results

3.2.2.1 Soundness

Proof. The proof follows from the definition of *can_collide* predicate. □

Theorem 3.3.2.1 (DTA soundness):

For any set of DTA schedules S and a DTA schedule t, if S⊢t then S⊨t.

Proof. Soundness of rules 1 and 2 follows from the definition of *strict_precede* predicate. Soundness of rule 3 follows from the definition of *precede* predicate. Soundness of rule 4 follows from the commutativity of *can_collide* predicate (Lemma 1).

Sub-schedule order. First we prove *left sub-schedule order.* Consider two valid DTA schedules X and S and assume that there is a sub-schedule $S_i \in subs(S)$ such that $valid(o(X,S_i))$. This implies that there are elementary transmissions *et1*, *et2* and mappings $map(S_i,MS_i)$ such that $member(et2,MS_i)$ and $valid(o(et1,et2))$. Meanwhile, $S_i \in subs(S)$ also implies $member(et2,MS)$, where $map(S, MS)$. Then, from the definition of *strict_precede* we conclude $valid(o(X,S))$. The proof of *right sub-schedule order* is conducted in the same way.

Sub-schedule choice and Sub-schedule overlap. The structure of the proof has the same schema as the proof of *sub-schedule orders.* The proof is based on the definitions of the *precede* and *no_conflict* predicates. □

3.2.2.2 Completeness Generally speaking, the DTA inference rules are not complete, i.e., we cannot prove that for any set of DTA schedules S and a DTA schedule t, if $S \models t$ then $S \vdash t$. As an example consider two valid elementary transmissions *et1* and *et2* such that *not precede(et1,et2)*, and *not can_collide(et1,et2)*. Then $\{et1,et2\} \models a(et1,et2)$. However, we cannot infer *a(et1,et2)* from $\{et1,et2\}$ using the DTA rules. The reason is that DTA does not

utilize the low-level semantics of transmission ranges and collision domains. Meanwhile, we can prove that the DTA rules *are complete with respect to a given query*: if the initial DTA specification reflects basic environment and query constraints, then any valid DTA schedule is also DTA inferable. Below we formally introduce the concept of the query completeness (*q-completeness*) and prove *q*-completeness of the DTA inference rules.

Definition 3.3.2.1 (query tree): For a given query Q we define a query tree T_Q as a set of all elementary transmissions required to evaluate Q.

Example 3. Consider query Q in Figure 2. Then $T_Q = \{n4\text{~}n2,\ n2\text{~}n1,\ n5\text{~}n3,\ n3\text{~}n1\}$.

Definition 3.3.2.2 (q-complete set): A set of DTA schedules S_Q is query complete (*q-complete*) with respect to a query Q if it includes all elementary transmissions of the query tree T_Q and all valid schedules *c(eti,etj)* and *a(eti,etj)* over the elementary transmissions of T_Q.

More formally

$$S_Q = T_Q \bigcup$$

$\{c(eti,etj) | eti,etj \in T_Q \wedge \textbf{\textit{valid(}}\ a(eti,etj)\textbf{\textit{)}}\ \} \bigcup$

$\{a(eti,etj) | eti,etj \in T_Q \wedge \textbf{\textit{valid(}}\ c(eti,etj)\textbf{\textit{)}}\ \}$

Example 4. For query Q from Figure 2

$S_Q = \{n4\text{~}n2,\ n2\text{~}n1,\ n5\text{~}n3,\ n3\text{~}n1,$

$c(n2 \sim n1,\ n3\text{~}n1),$

$a(n4 \sim n2,\ n5\text{~}n3),\ a(n4 \sim n2,\ n3\text{~}n1),$

$a(n5 \sim n3,\ n2\text{~}n1)\}$

Theorem 3.3.2.2 (DTA q-completeness): For any query Q and a DTA schedule t, if $S_Q \models$ t then $S_Q \vdash$ t.

Proof. Assume that $S_Q \models t$, but *not* $S_Q \vdash t$. If t is elementary transmission or *comp(S1,S2),* where *comp* is either ***c*** or ***a*** and *S1*, *S2* are elementary transmissions, then by definition of S_Q: $S_Q \models t \Rightarrow t \in S_Q \Rightarrow S_Q \vdash t$. If t is a *o(S1,S2)*, where *S1* and *S2* are elementary transmissions, then t can be inferred from $S_Q$by finite number of applications of the DTA rules 1 (*order introduction*) and 2 (*order transitivity*). If t is a *o(S1,S2)* and at

least one of *S1* or *S2* is not elementary transmission, then t can be inferred from $S_Q$by finite number of applications of the DTA sub-schedule rules 2, 5 and 6. If t is a *comp(S1,S2),* where *comp* is either ***c*** or ***a*** and at least one of S1 or S2 is not an elementary transmission, then t can be inferred from $S_Q$by finite number of applications of the DTA rules 3, 4, 7, and 8. Thus, $S_Q \models t$ implies $S_Q \vdash t$. □

3.2.3 Derived DTA Rules, Executable Schedules and Deadlocks

In order to increase the performance of our algebraic optimization we use the basic DTA rules to infer a set of derived rules. Figure 9 shows some examples of the derived DTA rules. The derived rules are utilized by our randomized optimizer as valid moves between DTA schedules [34], [74], and [76].

Overlap associativity	a(a(X,Y),Z) ↔ a(X,a(Y,Z))
Left A/O exchange	a(X,o(Y,Z)) → o(a(X,Y),Z)
Right A/O exchange	a(X,o(Z,Y)) → o(Z, a(X,Y))
A/C exchange	a(X,c(Y,Z)) → c(a(X,Y),Z)
Left O/A exchange	o(a(X,Y),Z), a(X,Z) → a(X,o(Y,Z))
Right O/A Exchange	o(Z, a(X,Y), a(X,Z) → a(X,o(Z,Y))
C/A exchange	c(a(X,Y),Z), a(X,Z) → a(X,c(Y,Z))

Figure 9: Derived DTA Inference Rules.

It is interesting to note that expected order associativity *o(o(X,Y),Z) → o(X,o(Y,Z))* and choice associativity *c(c(X,Y),Z) → c(X,c(Y,Z))* are not sound inference rules. Consider a query tree in Figure 10. The following DTA schedule is valid: *o(o(a(n1~n5,n4~n8), n9~n11), n10~n12).* However, the schedule *o(a(n1~n5,n4~n8),o(n9~n11, n10~n12))* is not valid, since ***valid(*** *o(n9~n11, n10~n12)****)*** is not true. Similarly, *c(c(n7~n10,n8~n10), n6~n9)* is valid, while *c(n7~n10, c(n8~n10, n6~n9))* is not valid. With the tree topology of Figure 10 transmissions *n8~n10* and *n6~n9* can be executed concurrently, i.e., ***valid(*** *a(n8~n10, n6~n9)****)*** holds.

It should be noted that while being invalid the above schedules *o(n9~n11, n10~n12)* and *c(n8~n10, n6~n9)* are still executable. Indeed, the fact that query semantics do not impose a strict order on the transmissions *n9~n11* and *n10~n12* does not mean that we cannot execute them in the order *o(n9~n11, n10~n12).* The same is true about *n8~n10* and *n6~n9.* An interesting question is if it is possible to generate a valid DTA schedule which would not be executable. This question has a positive answer. An example of a valid non-executable schedule is a *deadlocked* schedule.

Definition 3.3.3.1 (deadlock-potential schedules): DTA schedules *S1* and *S2* are deadlock-potential if both o(*S1,S2)* and o(*S2,S1*) are valid.

For example, in the query tree in Figure 10 schedules *a(n1~n5, n10~n12)* and *a(n4~n8, n9~n11)* are deadlock-potential. This is implied by the fact that both *o(n1~n5, n9~n11)* and *o(n1~n5, n9~n11)* are valid. Then, strict order of the deadlock-potential schedules will make a valid non-executable deadlocked schedule. The following is an example of a valid deadlocked schedule:

o(a(n1 ~ n5, n10~n12), a(n4 ~ n8, n9~n11)).

In order to capture the deadlock semantics we extend the DTA semantic definition with *deadlock(T1,T2)* predicate stating that DTA transmissions *T1* and *T2* are deadlocked. The following rule provides a formal definition of the *deadlock* predicate:

deadlock(T1,T2) :- strict_precede(T1,T2),

strict_precede(T2,T1).

Note, that negation of the *deadlock* predicate in the body of the *valid(o(S1,S2))* rule would invalidate the deadlocked schedules. This, however, would add more complexity to DTA inference rules in order to maintain DTA soundness and completeness. In order to preserve reasonable complexity of the query optimization we allow DTA rules to generate deadlocked schedules. Instead of making DTA deadlock-free we implemented efficient deadlock handling strategies..

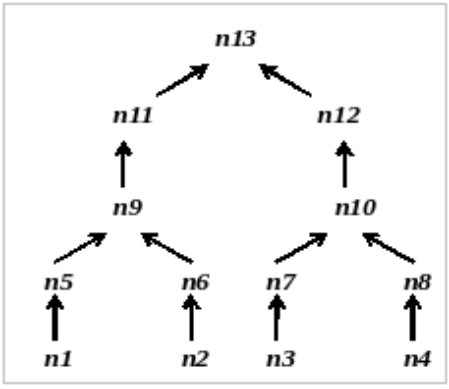

Figure 10: Invalidation of *order* and *choice* Associativity

3.3 COLLISION-FREE DTA SCHEDULES

In this section we will prove a key property of DTA: DTA inference rules generate only collision-free schedules.

Definition 3.3.3.1 (collision-free schedule): A DTA schedule S is collision-free if $\forall\, S_i, S_j \in$ *subs(S)*, $S_i \neq S_j$,*map*($S_i\, T_i$),*map*($S_j\, T_j$) implies *not collide*($T_i\, T_j$).

Theorem 3.3.3.1 (valid schedule is collision free):

For any DTA schedule t, if *valid(t)* then t is collision free.

Proof. If t is an elementary transmission then the fact that t is collision-free follows from Definition 5. Indeed, in this case $\forall\, S_i, S_j \in$ *subs(t)* $S_i = S_j = t$. Suppose t is a composed schedule *comp(S1,S2),* where *comp* is either o or c. Since t is valid, then there are mappings *map(S1,T1)* and *map(S2,T2)* such that *time_overlap(T1,T2)* is false (this follows from the definition of *strict_procede* and *procede* predicates). Thus, *collide(T1,T2)* is false, which implies that t is collision-free. Now assume that t is a composed schedule *a(S1,S2)*. Since t is valid, then there are mappings *map(S1,T1)* and *map(S2,T2)* such that *can_collide(T1,T2)* is false (this follows from the definition of *no_conflict* predicates). Thus, *collide(T1,T2)* is

false, which implies that t is collision-free. □

Corollary 3.4.3.1 (DTA-inferable schedule is collision free): For any set of valid DTA schedules S and a DTA schedule t, if $S \vdash t$ then t is collision free.

Proof. The proof follows from soundness of DTA (Theorem 1) and Theorem 3. □

4.0 PRACTICAL IMPLEMENTATION STRATEGIES

In this chapter, we discuss the practical implementation strategies for applying DTA and provide experimental evaluations.

4.1 SCALABLE OPTIMIZATION STRATEGIES

Basic DTA scheduling as described above may be expensive due to its combinatorial nature. The number of alternative schedules grows at least exponentially with the number of sensor nodes and elementary transmissions participating in query. In order to handle the optimization complexity, we utilize heuristic-based pruning methods that eliminate suboptimal alternatives and randomized algorithms [34]. Randomized algorithms are searching for a solution with the minimal cost performing random walks in the solution space via series of valid moves. Specific algorithms are different with respect to moving strategies and stopping conditions. Most well-known randomized optimization algorithms are Iterative Improvement (II) (Figure 11), Simulated Annealing (SA) and Two-Phase Optimization (2P0) ([51], [34]). In our case, possible solutions are DTA schedules. We define valid moves between DTA schedules on the basis of the DTA inference rules (Figure 8, 9).

4.1.1 Iterative Improvement Algorithm for DTA Optimization

Since, Data Transmission Algebra (DTA) enumerates all possible complete schedules it becomes very expensive to generate such schedules when the size of the tree does not remain reasonably small. Thus, we adapt Iterative Improvement algorithm to generate optimal

schedules while not trying to enumerate all possible schedules.

Input: A catalog containing all the nodes in the query tree, processing time for each of the nodes, elementary transmissions, transmission time for each elementary transmission, and order relation amongst elementary transmissions as obtained by using DTA.

Output: An optimal schedule obtained by transforming elementary transmissions from input catalog using DTA, and by comparing costs of different schedules obtained by random moves in the neighborhood.

```
Procedure II( ){
minS = Sser;
  while (not stopping_condition) do {
    S = random DTA schedule
    while (local_minimum(S)) do {
          S' = random DTA schedule
          in  neighbors(S)
          if cost(S') < cost(S) then
          S=S'
     }
          if  cost(S) < cost(minS) then
          minS=S
  }
return(minS)
}
```

Explanation of variable and parameters:

minS - current DTA schedule with minimal cost;
Sser – random serial DTA schedule;
S – random initial DTA schedule;
neighbors(S) – a set of schedules that can be generated from S via one valid move;
stopping_condition – number of considered initial schedules;
local_minimum(S) – a number of neighbors of S to be tested, of which none has lower cost than S. If the test is successful, S is considered to be a local minimum.

Figure 11: II Algorithm for DTA Scheduling

4.1.2 Experimental Analysis

4.1.2.1 Behavior of DTA Scheduler For this study we developed a simulation tested based on the ns-2 simulator with the CMU wireless extension [56]. We have implemented an Iterative Improvement DTA scheduler using Arity Prolog 32 version 1 [1]. First, we report on the behavior of the DTA scheduler. Then we compare DTA scheduling with 802.15.4 MAC.

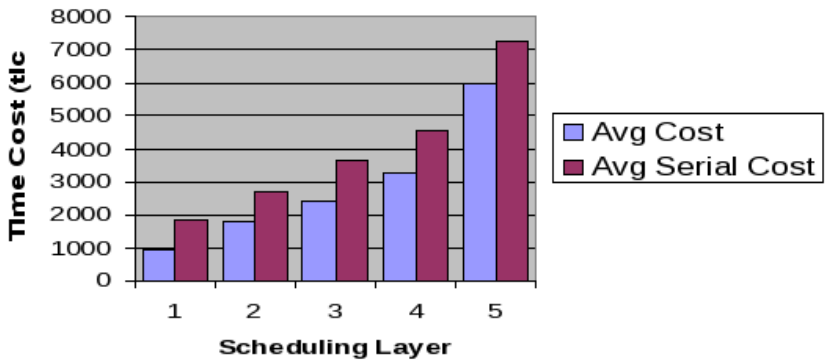

Figure 12: Comparison of DTA Scheduling with Serial Scheduling

Figure 12 shows the average query execution time for different scheduling stages. We compare the DTA scheduling with a *serial scheduling strategy* that performs elementary transmissions sequentially. For each scheduling stage we report the average execution time of all its schedules. We observe that at each scheduling stage, the approach that uses DTA considerably outperforms serial scheduling. We report on the behavior of the DTA scheduler for a medium complexity query tree involving ten sensor nodes with overlapping collision domains. Processing and transmission costs were generated randomly using Gaussian distributions. The DTA scheduler generated schedules stage by stage starting from initial schedules with two elementary transmissions (stage 1). Stage 2, 3 and 4 represent schedules with 3, 4 and 5 scheduled transmissions. Stage 5 includes complete schedules covering all elementary transmissions of the query tree. Figure 13 reports on the average benefit that each scheduling stages gains from concurrent transmissions. Intuitively, the benefit is part of the time cost that the DTA scheduler is able to "hide" scheduling some transmissions concurrently. The benefit is defined recursively for each of DTA operations. The benefit of a(X,Y) is equal to minimum of costs cost(X) and cost(Y). For the rest of the DTA operations the benefit is equal to zero. Thus, any serial schedule has a zero benefit.

Figure 13(a) compares values of average time cost and average benefit for each scheduling stage. With the increase of the number of transmissions the benefit grows, but not as fast as

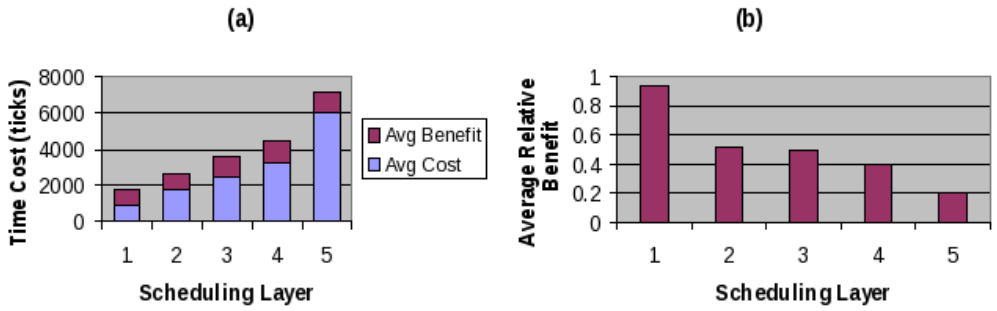

Figure 13: (a) Time Cost and (b) Relative Benefit of DTA Scheduling

the time cost. Figure 13(b) plots the average relative benefit as a percentage of the overall average time cost per scheduling stage. We observe that for simple initial concurrent schedules the benefit is almost equal to the time cost. This is an expected behavior. Elementary transmissions have comparable time costs. By scheduling them concurrently, DTA hides on average one half of the time cost of their serial execution. However, for complete schedules (stage 5) the average relative benefit is as low as 0.2, which means that only 20% of the total serial cost has been hidden. This is also an expected behavior, since complete schedules are composed of non-elementary transmissions (sub-schedules) with higher variance in their time cost. Thus, it is more challenging for the DTA scheduler to hide time costs of non-elementary sub-schedules.

4.1.2.2 Evaluation of the II-based DTA Scheduler Figure 14 shows some of our experiments that evaluated the performance of the Iterative Improvement (II) algorithm for DTA scheduling. It reports average time cost and benefit of all considered schedules (avg_cost and avg_benefit) and time cost and benefit of the winner schedule chosen by II algorithm (win_cost and win_benefit). In addition to costs and benefits of the schedules, we also report a value of average gain received from the local minimum phase of the algorithm (avg_lm_gains). The local minimum gain occurs when II algorithm improves a random initial schedule via given number of random moves. This number should be no greater than the local minimum condition. The upper left graph in Figure 14 illustrates a consistent improvement of II

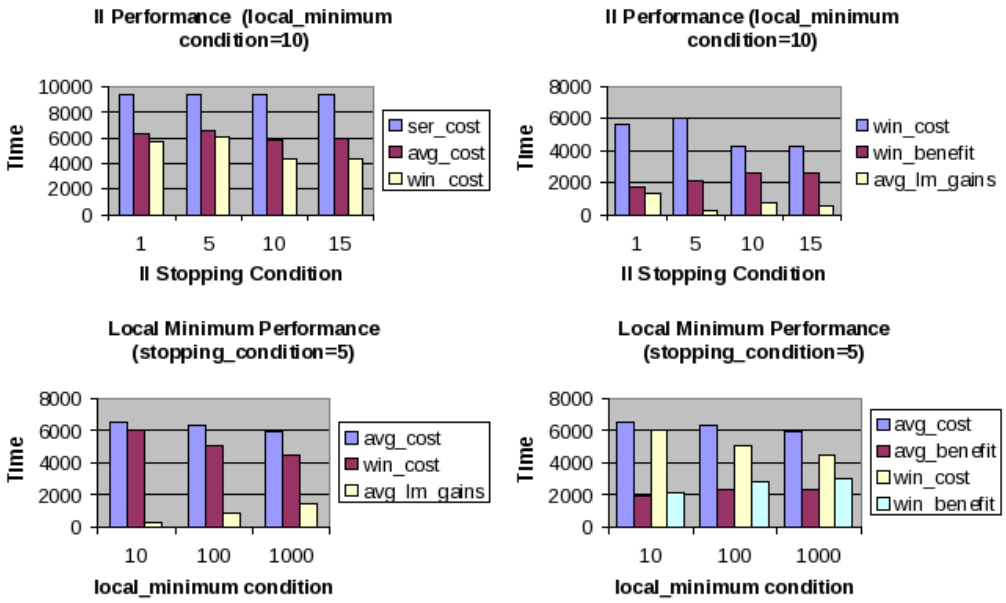

Figure 14: Performance of II-based DTA Scheduler

performance as we increase the values of the stopping condition with fixed local minimum condition of 10. We also provide a time cost of a serial schedule (ser_cost) as a reference point and a worst case scenario. The upper right graph also reports on benefit and local minimum gains of the winner schedule. While we observe steady increase of the benefit value, the local minimum gain behaves quite sporadically. This is an expected behavior, since for each value of II stopping condition we set the same local minimum condition. Thus, in general we should expect a random value of avg_lm_gains. In order to explore the performance of the local minimum phase we plot the cost, benefits and local minimum gains for different values of the local minimum conditions (lower two graphs of the Figure 14). We observe that the performance of the II algorithm consistently improves as we increase the values of the local minimum conditions.

4.1.2.3 Comparison of DTA with CSMA/CA Figure 15 reports on packet success ratio representing the percentage of transmitted packets that reach a destination (sink) node successfully. Recall, that a low packet success ratio that caused network collapses for higher

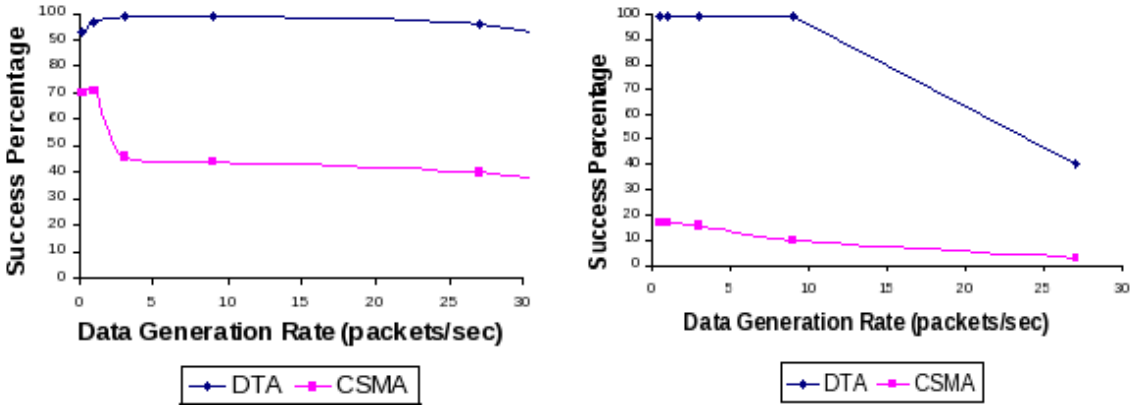

Figure 15: Packet Success Ratio for Medium and Large Networks.

traffic loads from critical monitoring applications, so it is important to observe how DTA helps to maintain it comparing to typical IEEE 802.15.4 that uses carrier sense multiple access with collision avoidance (CSMA-CA) [78]. Again, we experimented with medium and large star-like networks using CMU wireless and mobility extensions to ns-2 simulator. The medium network consists of 25 nodes positioned within a 150x150 meters flat area while the medium network includes 73 nodes. All nodes (except the central sink node) deliver packets to the sink in multi-hop fashion. We used 250 Kbps channel data rate with the sensor transmission range of 15 meters.

Figure 15 (left graph) shows the simulation results for a medium network. We observe that at lower loads, the packet success ratio of DTA is around 98%. It decreases to 80% as the data generation rate increases to 40 packets/second. This slight degradation is caused by insufficient queue buffer size of sensor nodes. Meanwhile, even at very low traffic loads (0.5 and 1 packet/sec) CSMA delivers only 70% of data to the sink node. When the load increases, the CSMA network becomes overloaded with collided or lost packets and the packet success ratio drops to 30%.

The benefit of DTA becomes even more obvious for a large network (Figure 15, right graph). At lower rates, the DTA packet success ratio is around 95%. The ratio decreases to 40% as the data generation rate increases to 27 packets/second. This is because the traffic

load exceeds the channel data rate (250 Kbps). Meanwhile, the packet success ratio with CSMA drops to less then 20% almost immediately.

4.2 INTEGRATING THE CROSS-LAYER OPTIMIZATION WITH EFFICIENT NETWORK INTERROGATION STRATEGY

In order to further facilitate the optimization, we propose to utilize efficient network interrogation strategies. Such strategies can decompose the wireless network into sub-networks reducing the scheduling complexity considerably. Below we propose a *Whirlpool* data delivery technique that splits the network in *sectors* and performs a *rotating interrogation* of the network, where groups of non-colliding sectors are queried in a rotation order.

4.2.1 Whirlpool: Rotating Interrogation

Consider a set of wireless sensor nodes spread uniformly over a monitoring area, and a base station is located at an approximate *center* of the network. The sensor network can be split into *sectors* as shown in Figure 16a, b, c. The number and size of sectors can vary depending on the monitoring requirements. We define two sectors S1 and S2 as *colliding,* if at least one transmission of S1 collides with any transmission of S2. We assume that adjacent sectors are always colliding. Transmissions within a group of non-colliding sectors can be conducted concurrently. This also introduces *inter-sector concurrency*, as opposite to intra-sector concurrency, where the DTA optimizer schedules concurrent transmissions within one sector. In Figure 16b, transmissions within each of (S1, S3) and (S2, S4) sector groups could be scheduled concurrently.

We investigate practical improvements of the basic Whirlpool technique, such as introducing *interlayer concurrency* that allows the optimizer to schedule last hops concurrently with initial transmissions of the next rotation. We also explore specific Whirlpool tuning for the purposes of efficient monitoring. For example, the Whirlpool can be fine-tuned with

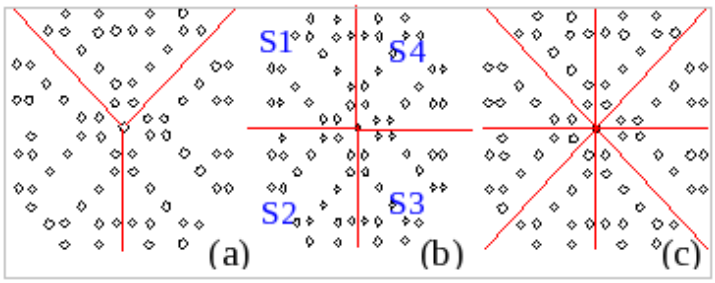

Figure 16: Sectoring of Sensor Network.

respect to the *number and size of sectors*. As the number of sectors increases and the number of nodes in each sector decreases, DTA query scheduling becomes more efficient (less complex) as the optimizer applies DTA to smaller sectors. At the same time, smaller sectors can also reduce the number of potential "good" query schedules explored by DTA.

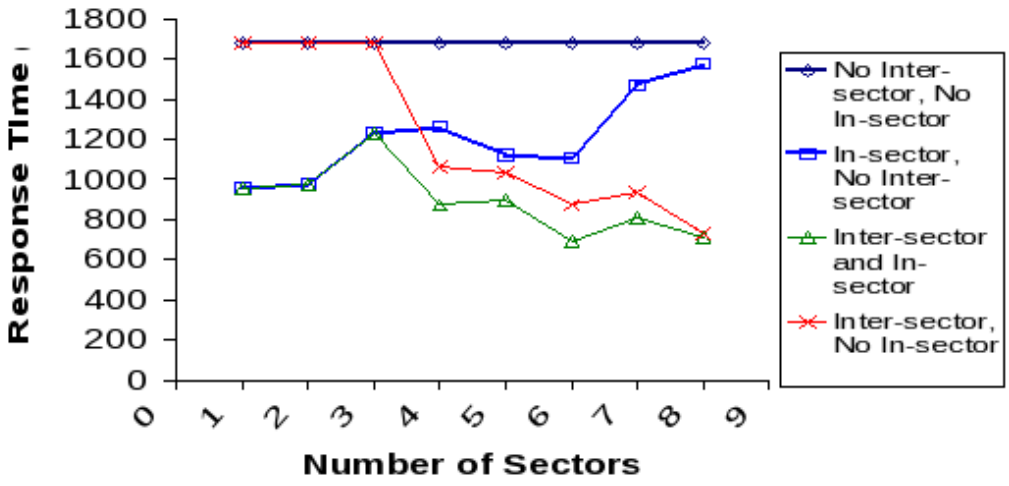

Figure 17: Effect of Whirlpool Concurrencies.

Figure 17 reports some preliminary simulation results on the ability of Whirlpool to utilize inter- and intra-sector (in-sector in Figure 17) concurrency. For this experiment, we used a simulated sensor network with 75 sensors uniformly spread over an area of 100×100 meters. A preliminary prototype of basic DTA and Whirlpool scheduling was implemented

in Arity Prolog 3.2. We report on one Whirlpool rotation with different sectoring. As a baseline, we used a serial strategy that executes transmissions in Whirlpool sectors one by one. This corresponds to the "No Inter-sector, No In-sector" curve in Figure 17. The next option is "Inter-sector, No In-sector" case where Whirlpool executes non-conflicting sector groups concurrently. The "In-sector, No Inter-sector" curve corresponds to DTA scheduling within each sector, while executing sectors one by one. Finally, we combine inter-sector concurrency with DTA scheduling ("Inter-sector, In-sector" curve) in an attempt to maximize the concurrency benefit. Figure 17 shows that, in general, any kind of Whirlpool concurrency improves upon the base line. Inter-sector concurrency does not give any benefit for 1, 2 and 3 Whirlpool sectors, since in this case any sector is adjacent to the rest of them and we do not have non-colliding sector groups. With an increase in the number of sectors, the benefit of inter-sector grows, which is the expected behavior. Indeed, since execution time of a non-conflicting sector group is equal to the execution time of its maximal sector, the smaller sector sizes improve performance due to the benefits of inter-sector parallelism. Meanwhile, the benefit from DTA scheduling only ("In-sector, No-Inter-sector") degrades with a larger number of sectors. The reason is that smaller and narrower sectors provide fewer opportunities for concurrent transmissions. We observe that with 8 sectors DTA alone is approaching pure serial Whirlpool execution, which means that only a few transmissions were scheduled concurrently. As expected, combining inter-sector concurrency and DTA scheduling demonstrates the best performance.

Thus, Whirlpool should be tuned for an optimal number and size of sectors so as to utilize the performance benefits of DTA scheduling while keeping the scheduling complexity reasonably low. Another important parameter of Whirlpool is its *speed of rotation*, which can be considered as the time spent by Whirlpool in each sector. It can also be interpreted as the size of the data sample received from each sector during one rotation. Faster rotation and smaller data sample size would also assume lower quality of data. Tuning the Whirlpool rotation speed, we can either deliver approximate data faster or accurate data slower. This provides additional opportunities for trading query response time for QoD, which is especially important in mission-critical monitoring applications.

4.2.2 Fine-tuning Whirlpool

The whirlpool can be fine-tuned with respect to the *number and size of sectors*, as well as the *number and speed of rotations*.

Number and size of sectors: As the number of sectors increases and the number of nodes in each sector decreases, query scheduling becomes more efficient. In case of using DTA, query scheduling becomes less complex as the optimizer applies DTA to smaller sectors. This results in a higher degree of intra-sector concurrency. Meanwhile, the larger number of sectors increases chances of *inter-sector concurrency*. However, while considering DTA based approach smaller sectors can also reduce the number of potential "good" query schedules explored by DTA. Whirlpool should be tuned for an optimal number and size of sectors so as to utilize the performance benefits of specific scheduling framework while keeping the scheduling complexity reasonably low.

Number and speed of rotations: Higher numbers of rotations imply that more data is collected from each sector. The number of rotations should be tuned on the basis of application requirements. The most important parameter of whirlpool is its *speed of rotation*. Speed of rotation can be considered as the time spent by whirlpool in each sector for one rotation. It can also be interpreted as the size of data sample received from each sector during one rotation. In general, the speed of rotation is higher for whirlpools with smaller sectors. Faster rotation would also assume lower quality of data. By tuning the whirlpool rotation speed, we can either deliver approximate data faster or accurate data slower.
To summarize, whirlpool introduces the following major advantages:

1. *Complexity Reduction in Scheduling*
2. *Introduction of Inter-sector Concurrency*
3. *Increase of Network Utility in terms of Control over the Behavior*

The last item requires more explanation. We define sensor network utility as the ability to provide a better quality of data relevant to understanding the behavior of the monitored system within shorter periods of time. Variable Quality of Data (QoD) that achieved by

whirlpool provides additional opportunities in trading query response time, energy and QoD, which is especially important in mission-critical applications such as structural monitoring for early instability detection. Using the whirlpool monitoring system we can localize the damage in particular network sectors and also evaluate propagation of unstable behavior.

4.2.3 Whirlpool Algorithms

4.2.3.1 Basic Algorithm: Serial Last-hops Consider a whirlpool structure with 4 sectors S1, S2, S3, S4 and a base station placed in the center of the network (Figure 18). We assume that any whirlpool sector has a single *last hop trans-mission* that reaches the base station. In Figure 18, t_{1lh} is a last hop transmission of S1, t_{2lh} is the last hop transmission of sector S2 and so on.

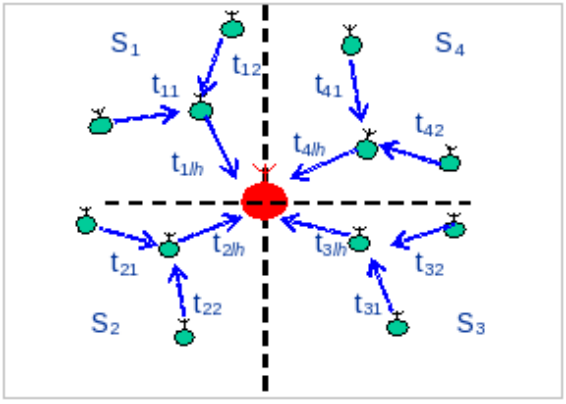

Figure 18: Whirlpool with 4 Sectors.

A *complete* sector schedule is a DTA schedule that includes all elementary transmissions of the sector[1]. For example, o(c(t_{11},t_{12}),t_{1lh}) is a complete schedule for sector S1. A *sub-complete* sector schedule is a DTA schedule that includes all elementary transmissions of the sector except its last hop transmission. For example, c(t_{11},t_{12}) is a sub-complete schedule for S1. Figure 19 represent the basic whirlpool algorithm (BA). First, the non-conflicting groups are formed. For the example in Figure 18 $NCGrp$ = [{S1,S3}, {S2,S4}]. Then the schedules are grouped by non-conflicting sectors as follows: Sch = [{c(t_{11},t_{12}),c(t_{31},t_{32})},{c(t_{21},t_{22}),c(t_{41},t_{42})}]

[1] We can use any efficient scheduling technique here. Whirlpool is orthogonal to DTA in this sense.

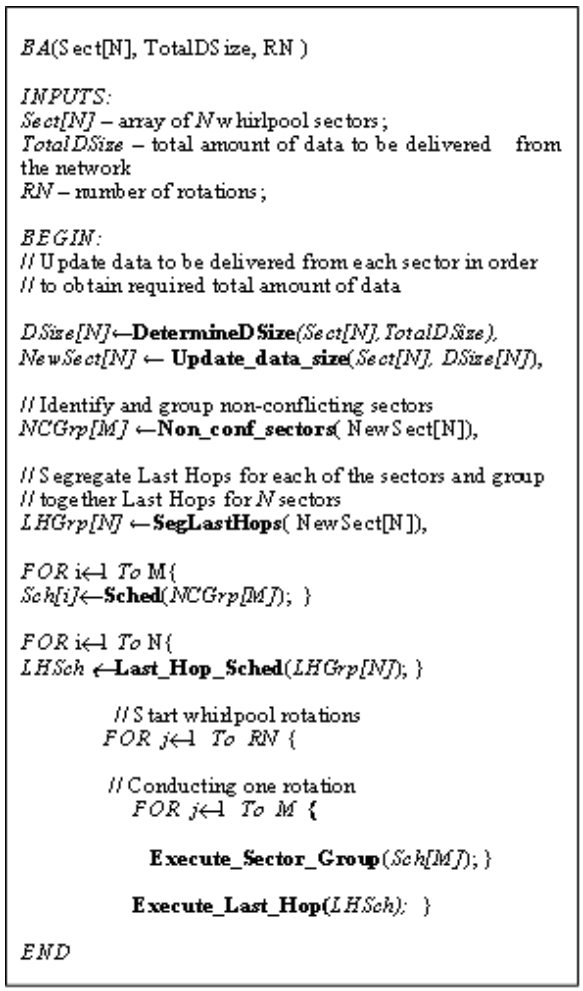

Figure 19: Basic Whirlpool Algorithm.

Each of the sector groups *Sch[1]* and *Sch[2]* are executed in the order of whirlpool rotation. Meanwhile sub-schedules within each group are executed concurrently. We can express this inter-sector concurrency using DTA overlap operation a as shown in the following expression:

$$o(\ a(\ c(t_{11},t_{12}),c(t_{31},t_{32})\ ),\ a(\ c(t_{21},t_{22}),c(t_{41},t_{42})\ )\ ) \quad \textbf{(1)}$$

The above DTA expression schedules all sub-complete schedules of the whirlpool in Figure 18. In order to complete the scheduling we have to take care of last hop transmissions. Since all last hop transmissions share the same destination, the optimizer schedules them serially using DTA non-strict order operations:

LHSch=c(t_{1lh}*,c(*t_{2lh}*,c(*t_{3lh}*,*t_{4lh}*)))* **(2)**

Combining schedules **(1)** and **(2)** using strict order operation we obtain a DTA expression for one whirlpool rotation:

o (o(a(c(t_{11},t_{12}),c(t_{31},t_{32})),

a(c(t_{21},t_{22}),c(t_{41},t_{42}))) ,

c(t_{1lh},c(t_{2lh},c(t_{3lh},t_{4lh})))).

With basic whirlpool algorithm, there is a bottleneck for the last-hops. In order to deal with this bottleneck, we propose a modified algorithm.

4.2.3.2 OHT Algorithm: One Last-hop through In this algorithm we allow the last-hop for one of the sectors to occur concurrently while other sectors still deliver the data to the nodes before the last hop node. Similar to the basic algorithm sub-complete DTA schedules Sch[i] are generated for each of the whirlpool sectors. However, this time one last hop also scheduled in the sub-complete schedule. Consider again Figure 18. In case OHT the groups of non-conflicting sectors will be as follows:

Sch = [{o(c(t_{11},t_{12}),t_{1lh}),c(t_{31},t_{32})},

{o(c(t_{21},t_{22}),t_{2lh}),c(t_{41},t_{42})}]

Each of the sector groups *Sch[1]* and *Sch[2]* are executed in the order of whirlpool rotation. Inter-sector concurrency is expressed as:

o(a(o(c(t_{11},t_{12}*),*t_{1lh}*),* c(t_{31},t_{32}*)),*

*a(*o(c(t_{21},t_{22}),t_{2lh}),*c(*t_{41},t_{42}*))).* **(3)**

To complete the scheduling remaining two last hops are scheduled serially using DTA non-strict order operations:

LHSch = c(t_{3lh},t_{4lh}*)* **(4)**

Combining schedules **(3)** and **(4)** as in the basic algorithm gives a complete schedule for one rotation:

*o (o(a(*o(c(t_{11},t_{12}),t_{1lh}),*c(*t_{31},t_{32}*)),a(*o(c(t_{21},t_{22}),t_{2lh}),*c(*t_{41},t_{42}*))) , c(*t_{3lh},t_{4lh}*)))).*

4.2.3.3 Layered Whirlpool In order to consider more concurrent transmission opportunities, we introduce layering in whirlpool. We consider how the whirlpool algorithms can

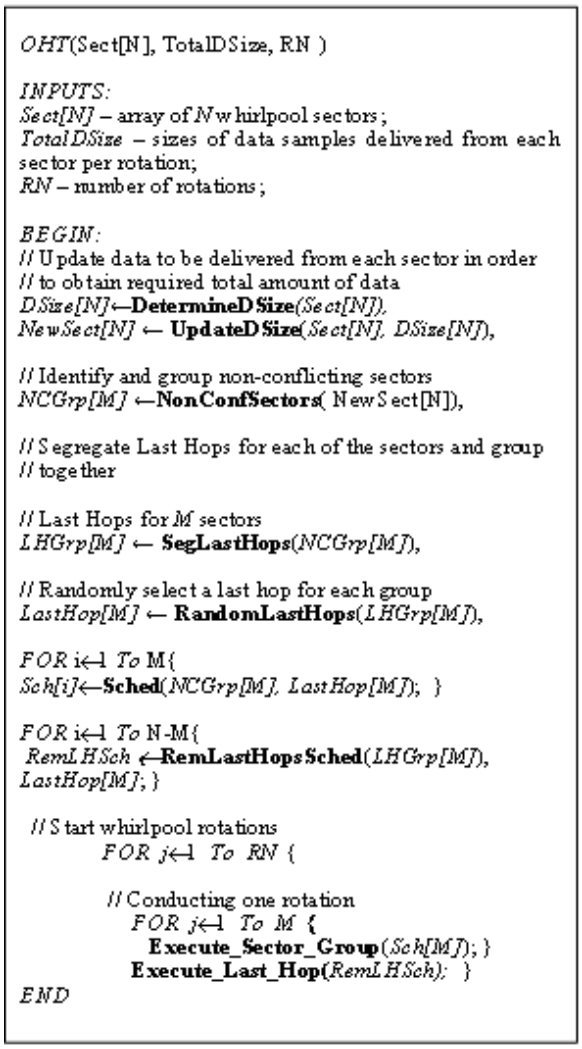

Figure 20: One-Hop Through (OHT) Algorithm.

be improved by introducing *interlayer concurrency*. First we explain Layered Basic Algorithm (LBA) where all last hops of the whirlpool are executed serially. Consider Figure 21a where the same color layer-sectors can be scheduled simultaneously. We assume that there is no conflict in transmissions across the layers. The outer-layer dark color sectors are scheduled initially to begin the whirlpool. The inner layer-sectors (closer to the base station) start transmitting only after the corresponding outer layer-sector transmissions have been completed. The layered whirlpool continuously acquires data from outer-layers and passes

it through inner-layers to the center. Figures 21b, 21c and 21d represent three execution stages during one whirlpool rotation. All dark-color sectors at each stage can be executed simultaneously. After the stage 3 (Figure 21d), the last hops are executed serially. Figure 23 provides the LBA specification.

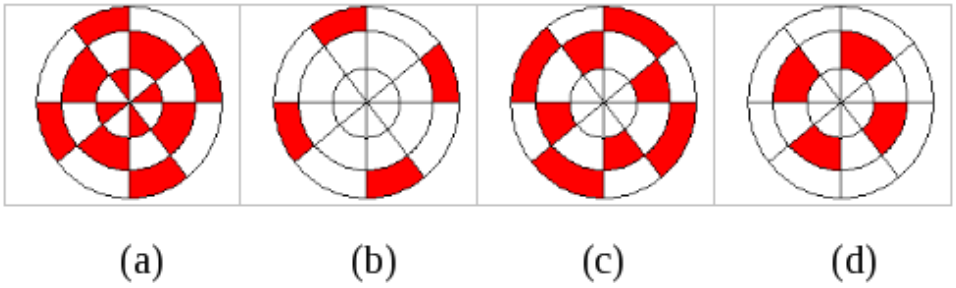

Figure 21: Explaining Layer-sectored Whirlpool Execution.

The second layered whirlpool strategy is the *LOHT* algorithm, which is the layered version of the OHT technique. The LOHT algorithm groups non-conflicting sectors and executes them along with one last hop from one of the sectors in the group. We also developed several advanced whirlpool algorithms maximizing the benefits of *continuous* concurrent transmissions. For example, Figure 22 illustrates the approach where transmissions from consequent whirlpool rotations co-occur. Figure 22d corresponds to the case where the last hop sectors are being executed while the outer-most layer starts executing transmissions for the second rotation. This maximizes the overall concurrency.

4.2.4 Experimental Analysis

Figure 24 compares overall response times for each of the whirlpool algorithms. Here we consider whirlpool with 3 layers. We observe a consistent improvement in the performance

of the whirlpool algorithm comparing to the BA strategy. We also observe that the response time decreases as the serial bottleneck for the last-hops is rectified. Meanwhile, most considerable impact on response time comes from the layering. The LOHT strategy wins over

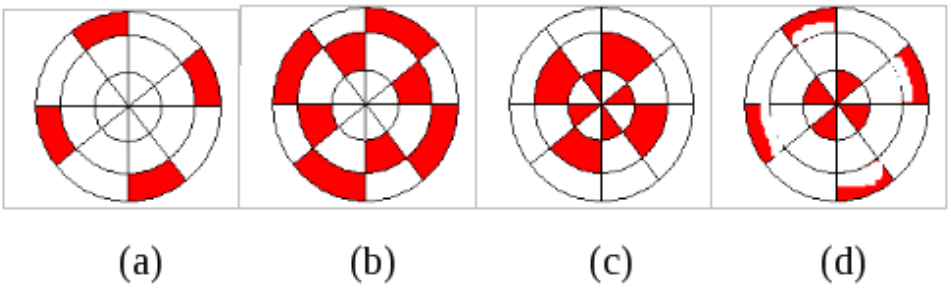

Figure 22: Whirlpool Execution using Advanced Algorithm.

all other algorithms for almost all number of the whirlpool sectors. The only exceptions are 1 and 2. In this case the performance of LOHT degrades since the execution time of the additional last hop becomes comparable with the execution time of a next layered-sector. For 3 or more sectors, LOHT behaves better than other algorithms, since in this case it considerably eliminates the last-hop bottleneck. However, with the increase in the number of layer-sectors the layering effect is diminished since the smaller size of the layer-sectors restricts intra-sector concurrency. It should also be noted that the complexity of the LOHT scheduling may enforce choosing simpler but less efficient algorithms such as LBA and OHT.

In order to investigate the utility of whirlpool for timely data delivery, we evaluated the data delivery profiles for each of the whirlpool algorithms. Figure 25 presents the results of this evaluation for the whirlpool structure with 8 sectors and 3 layers. We observe that OHT algorithm starts delivering data much earlier than BA. Similar relationship exists between LOHT and LBA. As expected the both LBA and LOHT over perform BA and OHT. The ability of whirlpool to deliver data early is very useful for mission-critical system monitoring. The crossover in the data delivery profiles indicates performance trade-offs that should be explored by our system. For example, we may choose OHT over LOHT if the early data delivery is preferable. Meanwhile, if faster overall response time is required then LBA should be chosen.

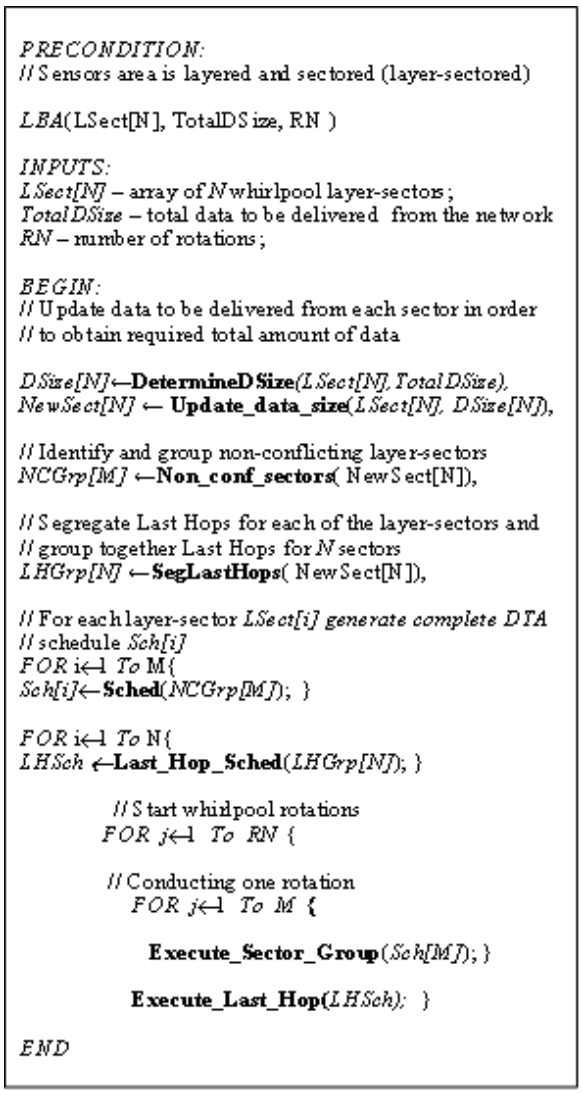

Figure 23: Layered Basic Algorithm for Whirlpool.

Figure 26 illustrates further possibility of improvement in the whirlpool algorithms via better time synchronizations. We compare the LOHT strategy with the two advanced algorithms that perform synchronized whirlpool interrogation using the idea outlined in Figure 112. Adv1 and Adv2 are improvements over LBA and LOHT algorithms respectively. Figure 26 represents benefits from one rotation only. Meanwhile the performance gain will increase with increase in number of rotations, since some of the outer-layer transmissions can be

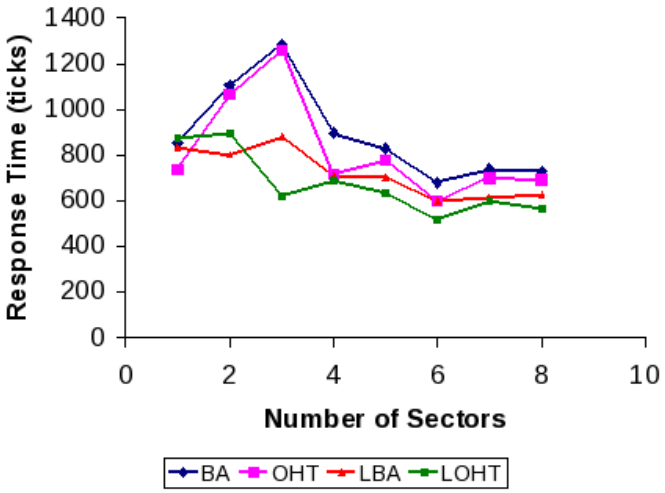

Figure 24: Response Time for whirlpool algorithms.

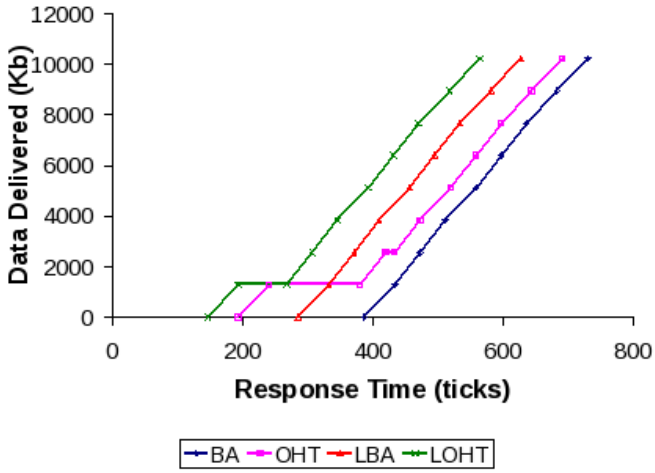

Figure 25: Data Delivery Profile for Whirlpool Algorithms.

executed concurrently with the last hops of the previous whirlpool rotations. Although it is beneficial yet, the time-synchronization is difficult to implement. We are currently exploring the practical applicability of these algorithms.

To summarize, our experiments demonstrated high utility of the whirlpool-based strate-

gies for timely data delivery in sensor networks. We also demonstrated how whirlpool algorithm can be tuned to maximize the transmission concurrency in order to meet the given performance target.

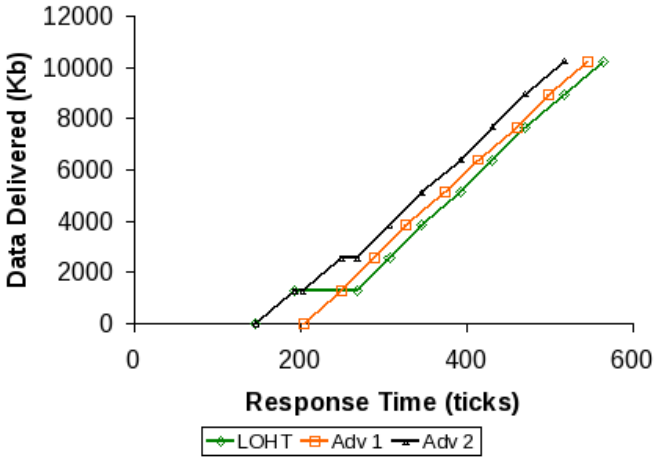

Figure 26: Profile for Advanced Whirlpool Algorithms.

4.3 CASE STUDY: DETECTING STRUCTURAL INSTABILITY

4.3.1 Background

Non-intrusive Structural Health Monitoring is a procedure where the natural dynamics of a structure are observed for changes that indicate damage or instability [54], [22]. By quantifying these changes, the system may detect and locate the damage. Recently, progress has been made in using techniques from non-linear dynamics based on analysis of chaotic excitation signals in detecting structural instability. In general, structural behavior is considered to be stable if observed structural parameters (e.g., vibration) are predictable, i.e., they are either constant or periodic (Figures 27a and 27b). A structure qualifies as unstable if its behavior is *chaotic*, i.e., if it exhibits a deterministic but non-predictable evolution (Figure 117c) [8], [68].

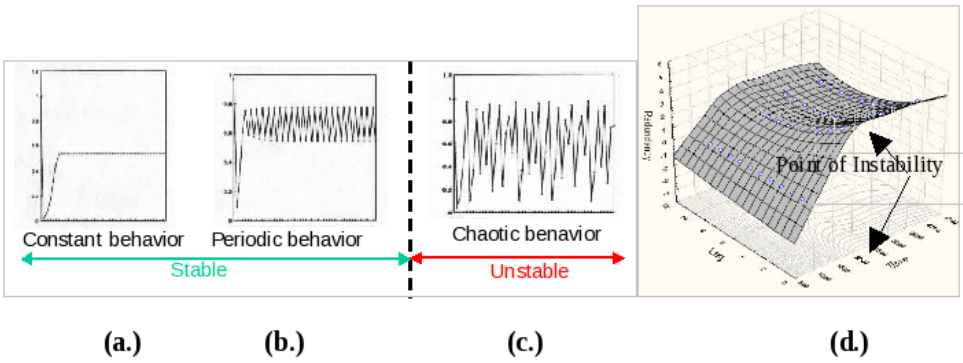

Figure 27: Data Patterns for Stable and Unstable Systems and Redundancy as Indicator of System Instability.

In order to detect chaotic behavior, a variety of measures have been used (e.g., Lyapunov exponent, Kolmogorov-Sinai entropy, etc.) [68]. We performed preliminary experiments with redundancy-based estimation of Kolmogorov-Sinai entropy and found it quite natural and efficient to detect chaotic behavior with WSNs. In this case, the data sample sensed by the sensors is time-shifted D times with a lag of m (D specifies the number of dimensions in the lag space). The resulting lagged time series Xt, $Xt+m, \ldots Xt+(D-1)m$ are compared with the original time-series for similarity that can be captured by information theoretical *redundancy measure* [68]. For stable data, the relation between redundancy and lag is a horizontal line at a constant ordinate value. For chaotic data, the plot gives us a downward sloping line. Figure 117d indicates the beginning of instability at time moment 2000, where we observe a change in redundancy.

4.3.2 Optimizing Detection of Structural Instability

The proposed cross-layer optimization can be effectively tuned to perform the instability detection as explained above. We assume that each sensor node accumulates a data sample every Ta seconds (*sample arrival rate*). Once a sample of data is received at the base station, it is tested for redundancy with a suitable lag and a chosen embedding dimension. The time interval from the beginning of transmission of a sample to the receipt of the sample

at the base station is known as the *propagation delay (Tp)*. As soon as a sample of data reaches the base station, it can be analyzed to detect any indications of instability. Consider a simple Whirlpool structure and a plot of system-generated vibration data in Figure 118. First, the sensor nodes in one of the non-conflicting group of sectors, e.g. {1, 1}, sample the vibration data (labeled by 1 in the vibration plot). Then, the collected data is transmitted to the base station that performs redundancy calculations. Note that the system misses the readings that occur during the data propagation delay. When the second group {2, 2} is sensing and transmitting, the first group {1, 1} stays idle and vice-versa. Thus, in this example the system is being interrogated twice per whirlpool rotation. Choosing specific Whirlpool sectoring and rotation speed, we can control the propagation delay, sample size, data freshness and amount of missed data.

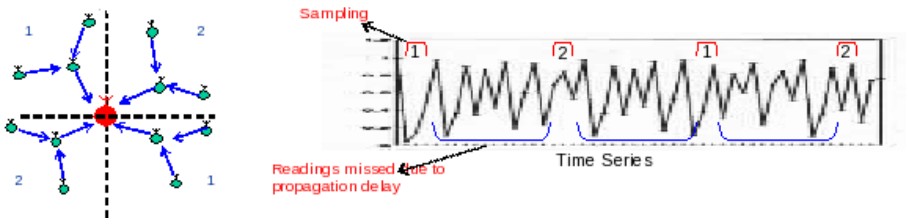

Figure 28: Simple Whirlpool for Instability Detection.

The choice of specific sectoring and rotation speed also impacts the accuracy of instability detection. If the sample size is too large, instability will be detected with a considerable delay. Meanwhile, estimating redundancy on a smaller sampling interval typically results in less notable redundancy changes compared to larger sampling intervals. Smaller samples may not provide enough data for accurate redundancy estimates. This is illustrated in Figure 29. The leftmost graph represents redundancy estimated on a sampling interval of 500, while the right redundancy graph corresponds to a sampling interval of 60. We observe steep redundancy degradation for the larger sample and smooth redundancy change for the smaller sample. Although a smaller sampling interval may have a potential for providing more timely instability detection, it risks missing the instability due to redundancy miss-estimation. Another problem may occur when a sample is collected in proximity of the

unstable region. Case (a) in Figure 29 corresponds to a scenario where the sample includes both readings from stable and unstable system states. The amount of stable readings can hide the chaotic pattern of the unstable region during the redundancy estimation. This can result in a missed instability point. Similar situations may occur when the system returns to stable behavior after some instability period (case (b) in Figure 29). In this case, a false alarm may be raised.

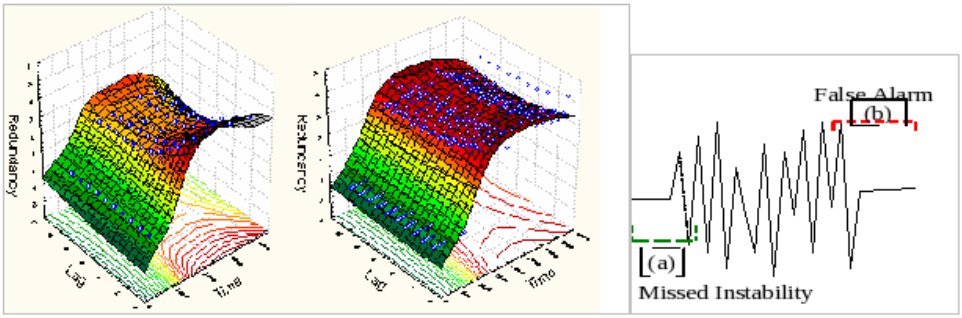

Figure 29: Effect of Sampling on Redundancy Increment, Missed Instability and False Alarms.

We explore applicability of our specific optimization techniques and tuning strategies in order to provide accurate and timely instability detection in SHM systems.

4.3.3 Experimental Analysis

In this section, we provide some experimental results evaluating utility of the Whirlpool technique for the task of SHM. We configured a simulated sensor network with 75 sensors uniformly spread over a monitoring area. We simulated Zigbee WSN [63] with a frequency band of 915 MHz and a nominal data transmission rate of 40 kbps. We set the actual transmission rate at 30 kbps with a packet size of 120 bytes. We also generated several vibration time-series with patterns of instability. Figure 30 plots the instability detection time for different numbers of the Whirlpool sectors and sample sizes reflecting the Whirlpool rotation speed. We observe that, in general, instability is detected earlier with more number of sectors. Meanwhile, the overall Whirlpool performance slightly decreases as the number of sectors increases. The reason is that smaller and narrower sectors provide fewer opportunities

for concurrent transmissions. We also observe that smaller sample size, which corresponds to faster Whirlpool rotation, allows Whirlpool to detect the instability earlier.

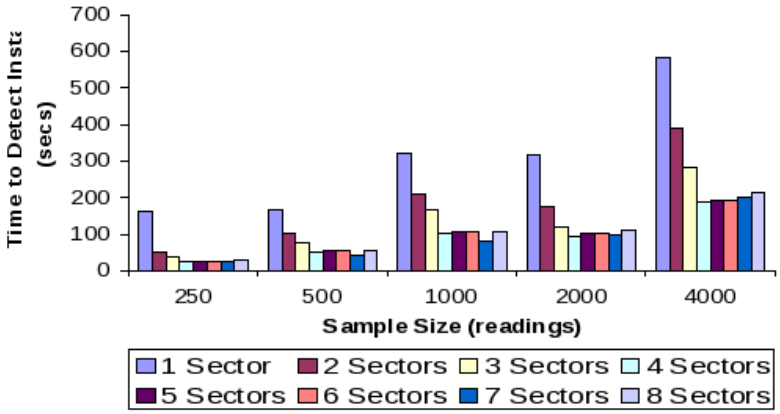

Figure 30: Instability Detection Time for Different Number of Sectors at Different Sampling Intervals.

At the same time the Whirlpool tuned for smaller samples is more vulnerable to false and missed alarms. This can be observed in Figure 31 that plots an average number of the false and missed alarms versus sample size. In general, an optimal choice of the sample size and cut-off accuracy is critical for accurate instability detection. From our experiments we found that a cut-off accuracy of 0.25 with sampling interval of 250 performs reasonably well.

4.3.4 Conclusion

We introduced a novel algebraic framework for specifying and analyzing data transmissions along with constraints imposed by a query in wireless sensor networks. Our framework enables flexible *cross-layer* query optimization techniques that utilize information about the MAC layer. We undertook a comprehensive experimental and theoretical study of our framework. It included the implementation and testing of our framework in simulated environments, as well as exploring its soundness and completeness. We also introduced a *rotating interrogation* technique called *Whirlpool* that provides opportunities to optimize data delivery in time-critical monitoring applications. We explored concurrency opportunities with

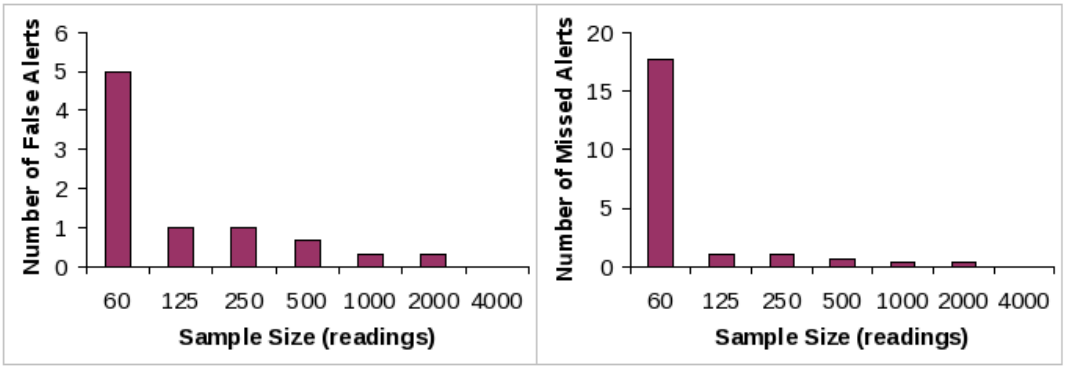

Figure 31: False Alarms and Missed Alerts.

different whirlpool algorithms. We also demonstrated extra benefits of using whirlpool in combination with our algebraic query optimization framework. Then, we studied the applicability of the DTA and whirlpool for the task of non-intrusive Structural Health Monitoring.

5.0 TRUST-BASED ROUTING

The majority of research on sensor networks is focused mostly on time/energy efficiency and various approaches addressing this issue have been proposed (e.g., maximizing collision-free concurrent data transmission [73]). Meanwhile, in many applications the reliability and trustworthiness of gathered data may be critical. In this chapter, we apply DTA framework to handle location-based trust in WSNs. The idea is to extend the DTA cost-model with trust estimation.

Consider a sensor query to collect temperature measurements from inside of a monitored building. Validity of these measurements is critical to discover, say, the spread of fire in the building in a timely manner. Therefore, we need to make sure that the received data are not changed while routed via the network. Meanwhile, there are many different routes to deliver data from temperature sensors. Some of them involve only sensors located inside the building. Other routes may be shorter and more time/energy efficient, but data should be routed through sensors located in a less protected outside area. Such common routing would ignore trustworthiness. As a result the last (possibly shorter) route would be preferred. The required level of trustworthiness could still be achieved assuming that cryptographic protection is applied and the measurements are encrypted. However, in this case the time/energy efficiency would decrease, since both encryption and decryption are time and energy consuming. Moreover, in many cases such cryptographic protection is not available or provides only limited protection due to resource constraints of sensor nodes. Even if the cryptographic protection is available it may be not sufficient. In order to use cryptography to provide authenticity and integrity of data, every participant should maintain a trusted key and a key management system for generation and distribution of cryptographic keys [19], [20], and [28]. This increases complexity and imposes additional time and energy requirements.

In our approach, a trust-aware routing assumes that all sensors in the network are assigned an initial trustworthiness level. Initially, trustworthiness of each sensor or groups of sensors can be determined from their context (e.g., properties of deployment area, sensor design, etc.). For example, deployment inside buildings may provide more physical protection to sensors compared to outdoor deployment, where intruders can have easier access to the sensor.

The trust-awareness based approach proposed in this chapter can be used when cryptographic protection is not available or is too resource demanding. The required level of trustworthiness can be achieved by routing data via trusted sensors, even though such trusted routes are longer and may be more time/energy consuming.

Sometimes, it would be more practical to choose an intermediate solution that combines those described above. Given some trustworthiness requirements, we can select a route to transmit data from the sensors to more powerful intermediate nodes that protect data cryptographically, and then route protected data through lower trustworthiness sensors.

5.1 SUBJECTIVE LOGIC

In this section, we consider how to express the level of trust in some metric. Following [37],[38] we first define the term opinion, denoted ω, that expresses opinion about trustworthiness level.

Let t, d and u be such that $t + d + u = 1$,$\{t, d, u\} \in [0, 1]^3$. Then a triple $\omega = \{t, d, u\}$ is called an *opinion,* where components t, d and u represent levels of trust, distrust and uncertainty respectively. The level of trustworthiness is expressed by opinions. For example, the trustworthiness level associated with distrust could be expressed as opinion $\omega_1 = \{0.0, 0.9, 0.1\}$, but trustworthiness level associated with maximum trust could be expressed as opinion $\omega_2 = \{0.96, 0.00, 0.04\}$. Varying these parameters, we can express different levels of trust. Expressing trust using three parameters instead of just one trust level provides a more adequate trust model of real world uncertainties. When different opinions are combined, these parameters are not treated equally.

The subjective logic defines a set of logical operators for combining opinions including conjunction, recommendation, and consensus, which we describe below.

Let $\omega_p^A = \left\{t_p^A, d_p^A, u_p^A\right\}$ denote an opinion of entity A about logical statement p being true. In the context of this chapter, A is a sensor node and statement$p$ corresponds to "data received by A reflects unchanged result of measurement".

If some entity A has opinion $\omega_p^A = \left\{t_p^A, d_p^A, u_p^A\right\}$ about some statement$p$, and opinion $\omega_q^A = \left\{t_q^A, d_q^A, u_q^A\right\}$ about some statement$q$, then$A$'s opinion about truthfulness of statement $p \wedge q$ is defined as following [37]:

$$\omega_{p\wedge q}^A = \omega_p^A \wedge \omega_q^A = \left\{t_{p\wedge q}^A, d_{p\wedge q}^A, u_{p\wedge q}^A\right\}$$

where $t_{p\wedge q}^A = t_p^A t_q^A$, $d_{p\wedge q}^A = d_p^A + d_q^A - d_p^A d_q^A$, $u_{p\wedge q}^A = t_p^A u_q^A + u_p^A t_q^A + u_p^A u_q^A$.

Let A and B be two entities that in context of this chapter represent sensors. Then $\omega_B^A = \left\{t_B^A, d_B^A, u_B^A\right\}$ denotes an opinion of entity A about trustworthiness of a recommendation given by$B$. Assume that B gives its recommendation to A about trustworthiness of a statement p in form of its opinion ω_p^B. Since entity A does not have any direct opinion ω_p^A about p, it will try to deduce some indirect opinion about trustworthiness of p based on the given recommendation, denoted ω_p^{AB}. For this purpose, subjective logic introduces the recommendation operator, denoted $\otimes$, as follows:

$$\omega_p^{AB} = \omega_B^A \otimes \omega_p^B = \left\{t_p^{AB}, d_p^{AB}, u_p^{AB}\right\}$$

where $t_p^{AB} = t_B^A t_p^B$, $d_p^{AB} = t_B^A d_p^B$ and $u_p^{AB} = d_B^A + u_B^A + t_B^A u_p^B$.

In the case where there are several independent opinions about the same statement, we can use consensus operator $\oplus$ to get combined opinion about the same statement. Let ω_p^A and ω_p^B be two opinions of entities A and B about statement p. Then the combined consensus opinion is defined as

$$\omega_p^{A,B} = \omega_p^A \oplus \omega_p^B = \left\{t_p^{A,B}, d_p^{A,B}, u_p^{A,B}\right\}$$

where $t_p^{A,B} = \left(t_p^A u_p^B + t_p^B u_p^A\right) / \left(u_p^A + u_p^B - u_p^A u_p^B\right)$, $d_p^{A,B} = \left(d_p^A u_p^B + d_p^B u_p^A\right) / \left(u_p^A + u_p^B - u_p^A u_p^B\right)$, $u_p^{A,B} = \left(u_p^A u_p^B\right) / \left(u_p^A + u_p^B - u_p^A u_p^B\right)$.

Now we are ready to describe how trustworthiness of different query trees can be found. For more details related to subjective logic the reader is recommended to consult [37],[38].

5.2 SYSTEM MODEL

At the beginning, we have to assign a level of trust to each sensor based on its location, design, etc. We consider a system model as described in [73]. We assume that a sensor query originates at a base station propagating it to all sensor nodes in the network and collecting the results back from the sensors. Query processing can take place at the base station and/or within sensor nodes, depending on the type of query and capabilities of sensors.

We also assume that a query optimizer runs at a base station and generates a set of alternative query routing trees. The optimizer selects a query tree that delivers results satisfying trustworthiness requirements (in addition to other requirements, such as finding maximizing concurrent transmissions). The query optimizer has to assess trustworthiness of all data gathered using the given query tree.

We assume that the sensor network contains two types of nodes: sensors and base stations. Every sensor can be associated with one or several base stations. As a basis for our trust measurement, we assume that each sensor and each base station have an assigned opinion about the trustworthiness of the sensors it queries, and each base station knows opinions about sensors it controls, as well as opinions about trustworthiness of recommendations of sensors that it controls as well as other base stations. Using subjective logic any base station can calculate the trustworthiness of data received from underlying sensors and from other base stations.

Let us consider the sensor network presented on Figure 32, consisting of one base station and six sensors aggregating data.

Let n_1 be a base station that gathers measurements from sensors n_2, n_3, n_4, n_5. Since n_1 has a direct connection to only sensors n_2 and n_3 the trustworthiness of measurements from other sensors n_4, n_5, n_6, n_7 should be evaluated with respect to corresponding trustworthiness

recommendations of intermediate sensors n_2 and n_3. Additionally, we assume that sensor n_3 aggregates independent opinions about the same statement p assessing data from n_3, n_6 and n_7, and forwards the consensus opinion to n_1.

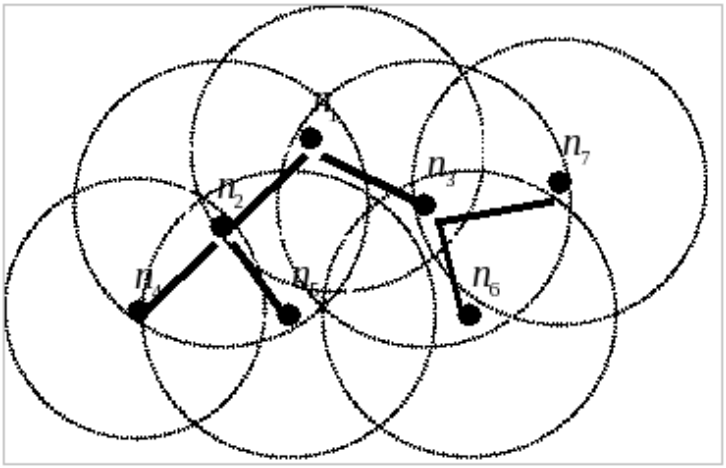

Figure 32: Sensor network with base station $n_1$and aggregation sensor node n_3.

Following the rules of subjective logic presented in Section 3, we can find that trustworthiness of data from collected from sensor nodes as following:

$$\omega_{p_2}^{n_1 n_2} = \omega_{n_2}^{n_1} \otimes \omega_{p_2}^{n_2}$$

$$\omega_{p_4}^{n_1 n_2 n_4} = \omega_{n_2}^{n_1} \otimes \omega_{n_4}^{n_2} \otimes \omega_{p}^{n_4}$$

$$\omega_{p_5}^{n_1 n_2 n_5} = \omega_{n_2}^{n_1} \otimes \omega_{n_5}^{n_2} \otimes \omega_{p_5}^{n_5}$$

$$\omega_{p}^{n_5} = \omega_{n_2}^{n_1} \otimes \omega_{n_5}^{n_2} \otimes \omega_{p}^{n_5}$$

$$\omega_{p}^{n_1 (n_3, n_6, n_7)} = \omega_{n_3}^{n_1} \otimes \left(\omega_{p_3}^{n_3} \oplus \omega_{p_6}^{n_6} \oplus \omega_{p_7}^{n_7} \right)$$

where statement p_i is true if delivered measurement of sensor n_i correctly represents the original measurement of n_i for $i = 2, 4, 5$ and p_3, p_6, p_7 are three equivalent statements denoted as p. Therefore, trustworthiness of data gathered by base station n_1 can be expressed as following:

$$\omega_{p_2 \wedge p_4 \wedge p_5 \wedge p}^{n_1} = \omega_{p_2}^{n_1 n_2} \wedge \omega_{p_4}^{n_1 n_2 n_4} \wedge \omega_{p_5}^{n_1 n_2 n_5} \wedge \omega_{p}^{n_1 (n_3, n_6, n_7)}$$

5.3 TRUST-AWARE DATA GATHERING

Let us consider an example of a sensor network with one base station (see Figure 33). The network consists of the nodes $n_1, n_2, \cdots, n_{10}$ where n_1 denotes the base station. The dashed red rectangle outlines a protected area in the sense that sensors n_1, n_2, n_3, n_5, n_7 located inside the rectangle have higher trustworthiness compared with sensors $n_4, n_6, n_8, n_9, n_{10}$ located outside of the protected area.

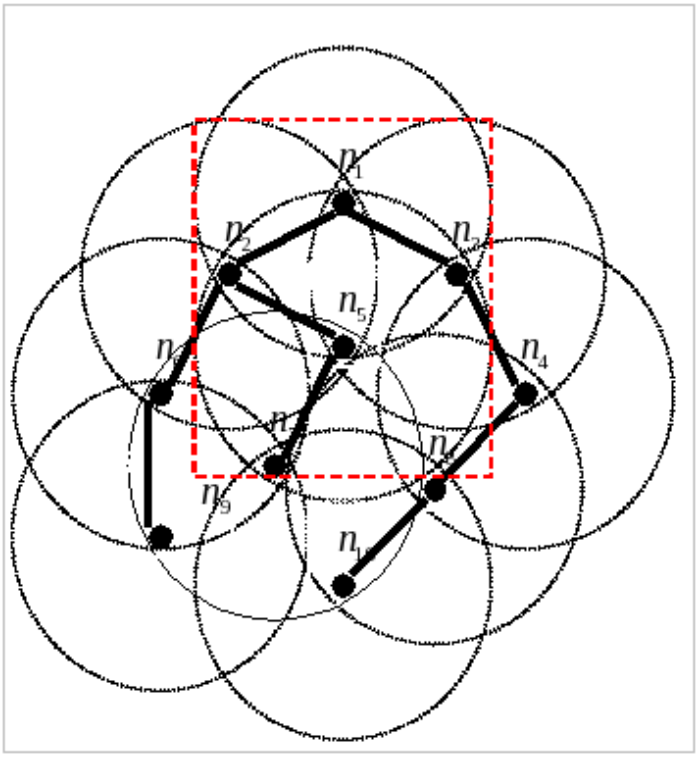

Figure 33: Routing query tree optimized without consideration of trustworthiness requirements

According to our model, node n_1 knows opinions about the trustworthiness of data produced by all sensor nodes, that is, $\omega_{p_2}^{n_2}, \cdots, \omega_{p_{10}}^{n_{10}}$ and trustworthiness of recommendations $\omega_{n_j}^{n_i}$ for all nodes n_i and n_j, $i \neq j$, that may directly communicate with each other.

Following subjective logic rules from Section 3, we can calculate the trustworthiness of data gathered by n_1 using the query tree presented in Figure 33 as following:

$$\omega^{n_1}_{t_2\wedge\cdots\wedge t_{10}} = \omega^{n_1n_2}_{t_2} \wedge \omega^{n_1n_3}_{t_3} \wedge \omega^{n_1n_2n_4}_{t_4} \wedge \omega^{n_1n_2n_6}_{t_6}$$
$$\wedge\omega^{n_1n_2n_5}_{t_5} \wedge \omega^{n_1n_2n_5n_7}_{t_7} \wedge \omega^{n_1n_2n_5n_7n_8}_{t_8}$$
$$\wedge\omega^{n_1n_2n_5n_7n_9}_{t_9} \wedge \omega^{n_1n_2n_5n_7n_{10}}_{t_{10}}$$

In cases where trustworthiness requirements are higher, we have to choose a non-optimal routing tree presented in Figure 34. It shows the same sensor network with a query routing tree having a higher level of trustworthiness.

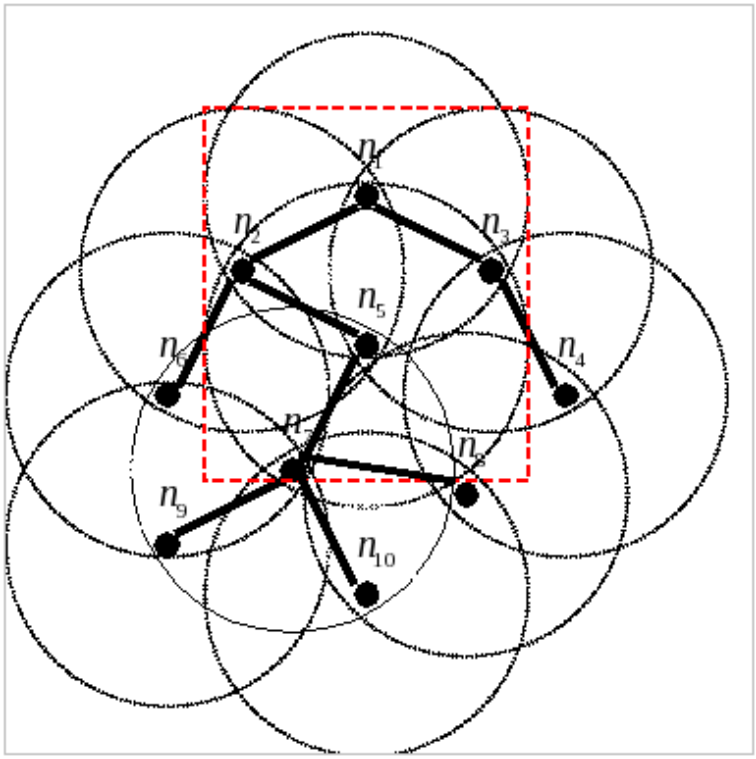

Figure 34: Routing query tree optimized with respect to trustworthiness requirements

Compared to the routing query tree in Figure 33, data from node n_9 will be routed via the more trustful node n_7, not n_6. The same holds for nodes n_8 and n_{10}. Trustworthiness for the case described in Figure 34 can be calculated similarly as it was demonstrated for the case in Figure 33.

5.4 EXPERIMENTS

First we tested our trust-aware processing scheme on a wireless sensor network topology consisting of 10 nodes (Figure 35).

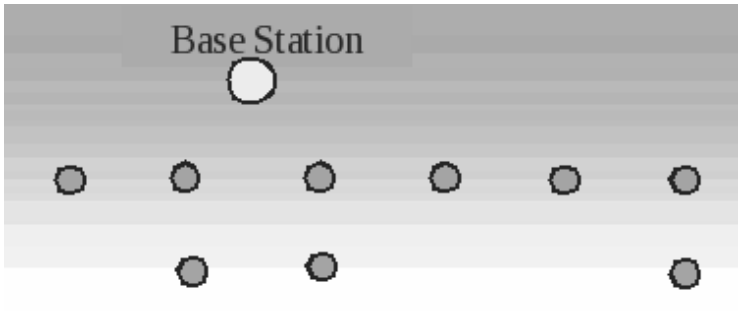

Figure 35: Small network topology.

We assumed that the network is data intensive. The data from all the nodes is sent to the *base station* using multi-hop delivery. Each node transmits at least 128 Kb of its own sensed data and additionally, relays the data that it received from its children nodes to its next hop. Each node is also associated with an *opinion i.e.* a triple of {*trust, distrust, uncertainty*}. The initial *trust* values are high and randomly set to be in the range from 0.97 to 0.99. Although, this range is pretty small, any small change in the second decimal place values of trust semantically reflects a bigger magnitude of change in trust because of extreme sensitivity of trust as defined in the subjective logic recommendation and conjunction operators [38]. *Distrust* and *uncertainty* are randomly assigned to each node such that for each node the expression $trust+distrust+uncertainty = 1$. We evaluated time, energy and trust values for this topology over a set of five different routes shown in Figure 36.

Figure 37 shows the time cost for each of routes illustrated in Figure 36. We observe that data delivery via route 3 has the minimum latency, whereas route 4 has the worst latency. Time cost for routes 1, 2 and 5 are almost similar.

Meanwhile, route 1 consumes minimal energy (Figure 38). Route 4 has the worst energy usage. Also, from Figure 39, we observe that route 4 has worst trust and highest uncertainty too. A trust-aware optimizer would prefer to choose route 1 or route 3, which have

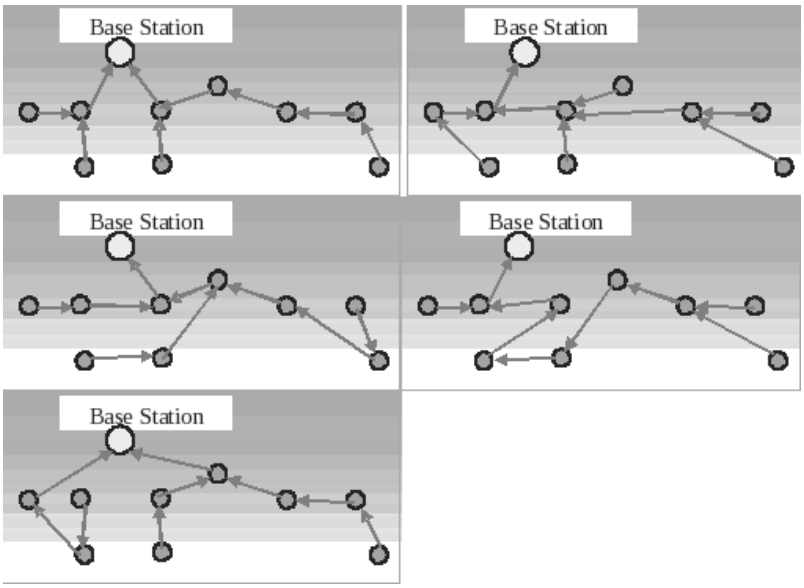

Figure 36: Five different data delivery routes for the topology in Fig.35.

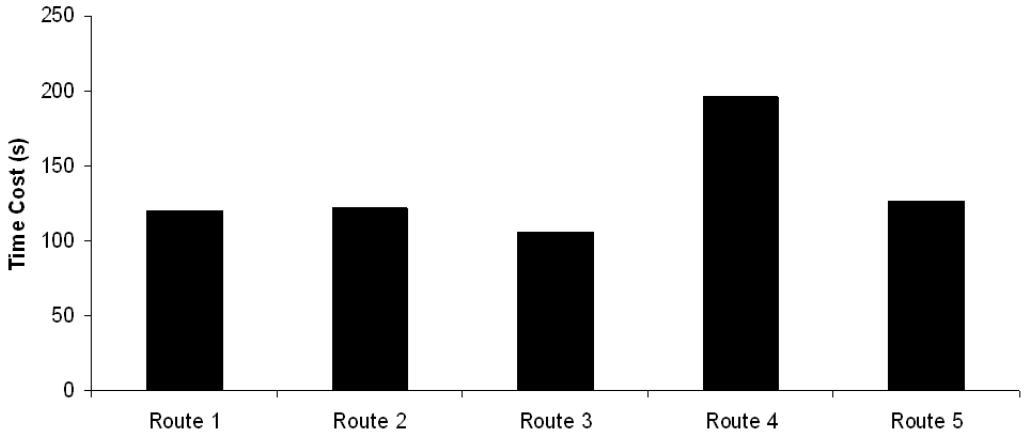

Figure 37: Time cost for 5 routes in Fig. 36

comparable time, energy and trust.

Next we performed a larger-scale experiments with 73 nodes positioned over a 1200mX1200m area as shown in Figure 40. Each node has a range somewhere between 100m – 150 m. Here, the central node is our base station. All the other nodes are either source or relay

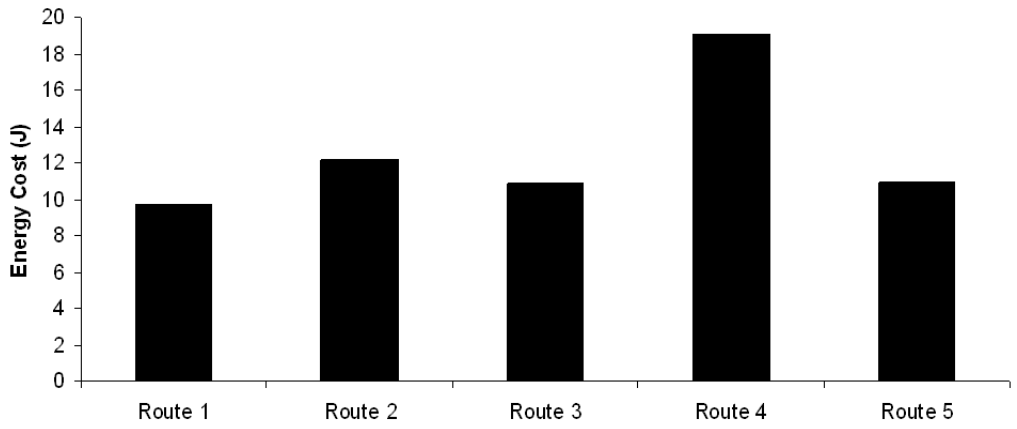

Figure 38: Energy cost for 5 routes in Fig. 36

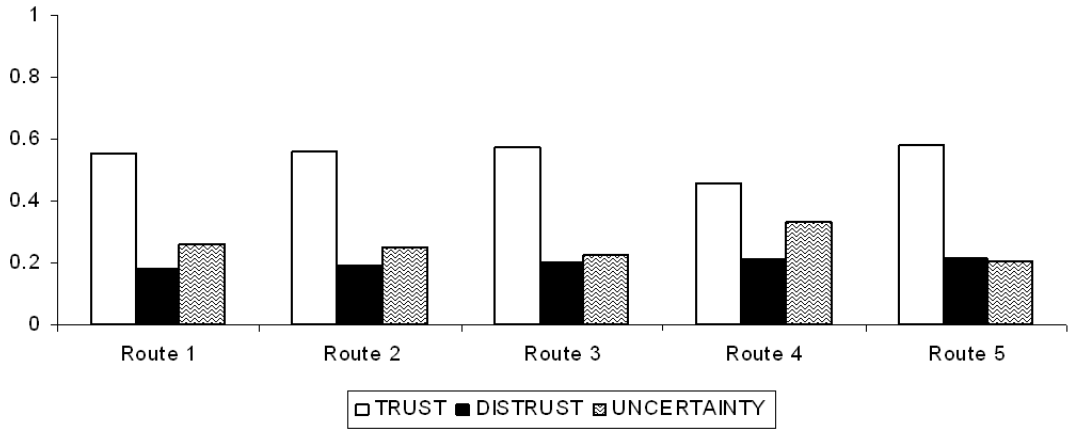

Figure 39: Trust, Distrust and Uncertainty for the 5 routes in Figure 36

nodes. The source nodes receive data from the environment and send these data to the base station via multi-hops using the intermediate relay nodes. Relay nodes do not contribute their own data. Each source node transmits at least 128 Kb of its own data. We identified 6 almost equally distant peripheral nodes as source nodes. Each source node utilizes 3 alternative paths to transfer its data to the base station. For each simulation run each of the 6 nodes selects one of the available paths and delivers data using that path. Hence, for

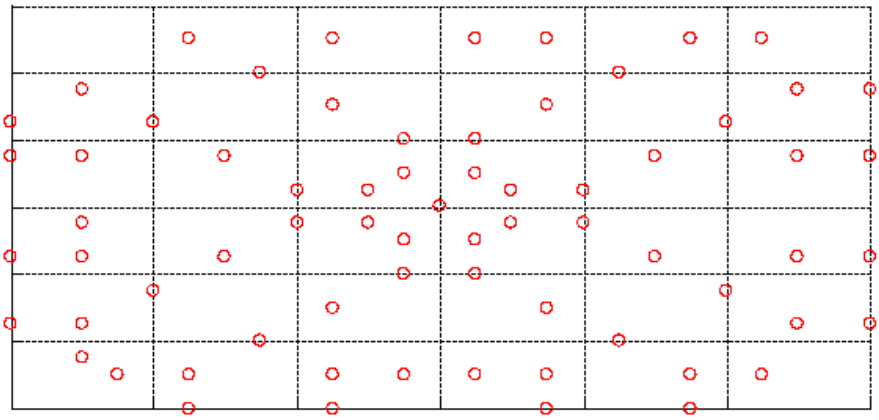

Figure 40: 73 nodes topological grid for simulation experiments.

every simulation we obtained a *spider* shaped topology (a "spider leg" corresponds to a path from source node to the base station node). We performed simulation studies for 729 such topologies, and report results based on few of these, which provide a representative behavior observed from all the simulation studies.

For each spider topology, our optimizer found optimal schedules based on time cost, energy cost and trust values. Our optimizer has a cost model to estimate both response time and energy consumption of a DTA schedule. Details of the time and energy model are in chapter 3 ([75]). As for the small network, the initial *trust* values were randomly set in the range from 0.97 to 0.99 since even small change in the second decimal place values of trust semantically reflects a bigger magnitude of change in trust because of extreme sensitivity of trust as defined in the subjective logic recommendation and conjunction operators. *Distrust* and *uncertainty* are randomly assigned to each node such that for each node the expression $trust+distrust+uncertainty = 1$. In order to perform the Multi-criteria Optimization Problem (MOP) our optimizer utilized randomized algorithms [34] to generate Pareto fronts for large query trees. Randomized algorithms will search for a Pareto optimal solution by performing random walks in the solution space via a series of valid moves [76].

We obtained a variety of time and energy optimal schedules with a distribution of trust, distrust and uncertainty values as shown in Figure 41. The trust values for the schedules

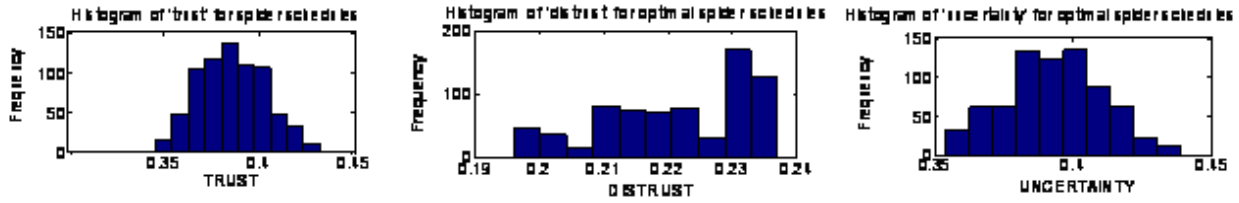

Figure 41: Distribution of opinions for the obtained sub-optimal schedules.

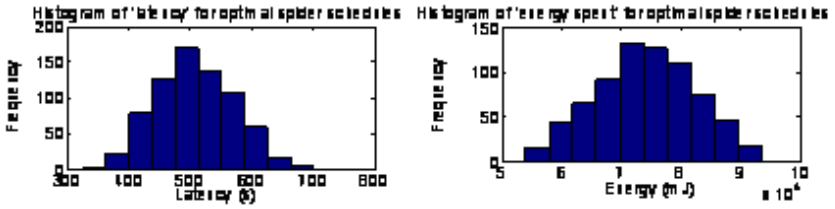

Figure 42: Distribution of latency and energy spent for the obtained sub-optimal schedules.

range somewhere between 0.35 and 0.45; and so do the uncertainty values. Distrust ranges from 0.19 to 0.24. Figure 41 represents the histograms of the opinion distributions for the generated schedules. We observe that most of the obtained sub-optimal schedules are associated with 0.38-0.4 trust values. Figure 42 represents latency and energy histograms. We observe that most of the schedules have latency in the range of 400- 600s and energy cost from 70 to 80 J. There is a clear winner schedule with the lowest latency and similarly, a clear winner with least energy cost and trust.

Experiments with varying trust levels High trust: ranging from 0.91-0.99

Figure 43 shows the time cost; energy cost and trust values for various schedules explored for optimization where the initial trust values were in the high range varying from 0.91 to 0.99. Here, most of the generated schedules had trust values between 0.1 to 0.13 with latency ranging from 450s to 600s and energy costs of 70 to 80J. These schedule trust values are much higher than 0.008 to 0.014 (Figure 44) trust values which were generated for the schedules

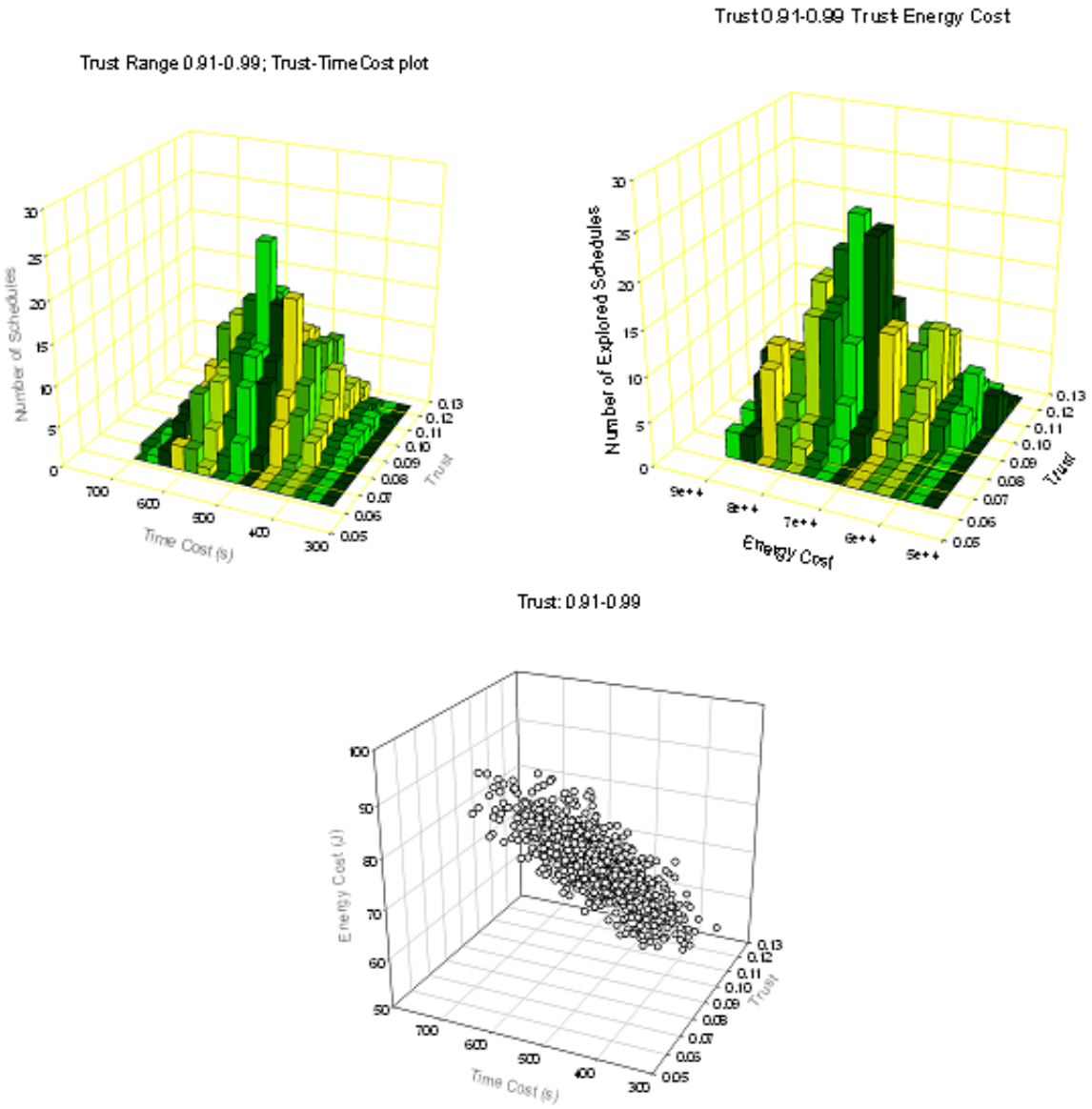

Figure 43: Time, energy and trust for various schedules with initial high trust values.

where initial trust values ranged from 0.8 to 0.99, although the time cost and energy costs were still comparable.

Medium trust level: trust ranging from 0.8-0.99

Heterogeneous trust levels

To experiment with a network associated with heterogeneous trust values, we kept 19 nodes

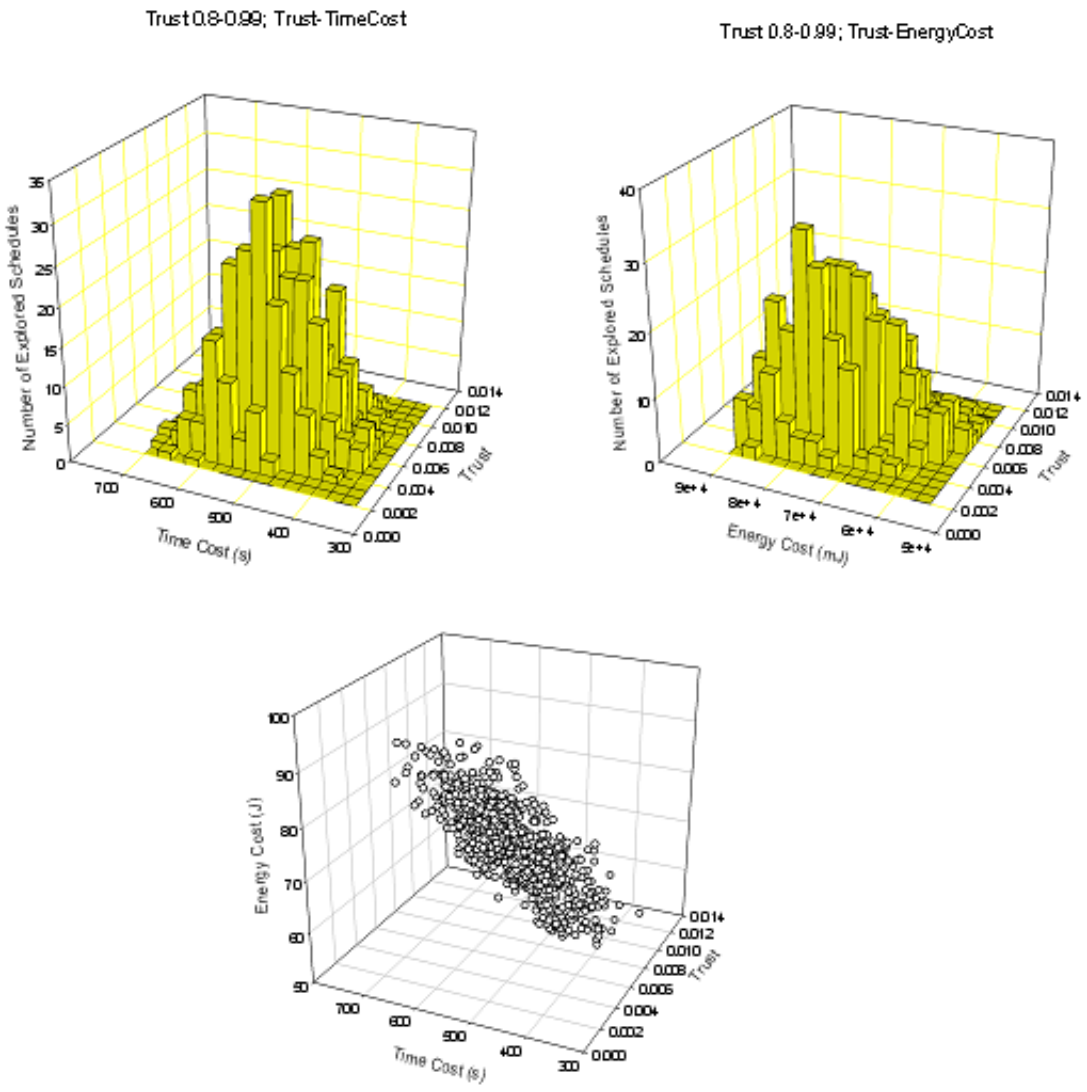

Figure 44: Time, energy and trust for various schedules with initial medium trust values.

with trust=1 (well-protected nodes), 6 nodes with trust=0.3 (unprotected nodes) and rest of the nodes were assigned trust randomly between 0.8 and 0.97 (medium protection). Figure 45 shows that the schedules generated were more biased in terms of their final trust values and energy cost. We could clearly find the schedules that were high trust and that still maintained low energy costs. Further, we explore such optimization opportunities.

Trust-aware optimization: Table 1 summarizes results from our exploration of various

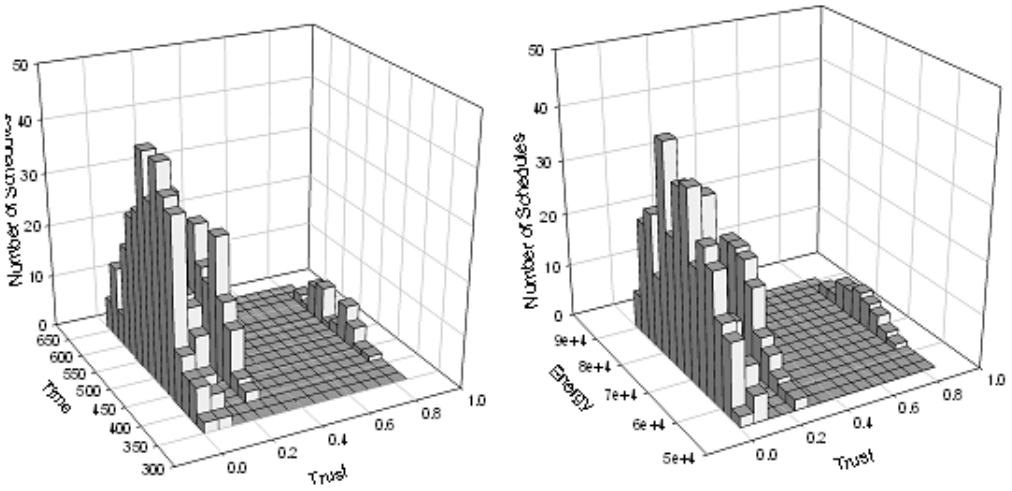

Figure 45: Time, energy and trust for various schedules with heterogeneous initial trust values.

possible schedules in the network. We may want to generate schedules with maximum trust, minimum distrust, minimum uncertainty, minimum latency and minimum energy for meeting particular performance targets. However, we would expect our optimizer to generate an ideal schedule that would have maximum trust, and minimum distrust, uncertainty, latency and energy cost. So, the optimizer would aim to generate a schedule as close to the best desired schedule represented in Table 2.

Table 1: Opinion, latency and energy cost values for some interesting schedules

	Trust	Distrust	Uncertainty	Latency (s)	Energy (mJ)
Max Trust Schedule	0.432	0.204	0.364	472.5	58241.9
Min Distrust Schedule	0.379	0.196	0.425	550.1	75803.0
Min Uncertainty Schedule	0.411	0.235	0.354	505.0	66746.4
Min Latency Schedule	0.419	0.214	0.367	325.9	58900.3
Min Energy Schedule	0.424	0.2	0.376	376.7	53909.5

In order to do so, our optimizer explores Pareto fronts reflecting continuous improvement of one optimization criteria while not degrading another one. Figure 46 represents Pareto fronts illustrating the trade-off between trust and latency explored by the optimizer. For example, for time cost 479 s, the optimizer could explore 3 schedules each with trust values 0.376, 0.399, 0.424 respectively; hence, showing improvement in trust. Similarly, in Figure 47, we show the Pareto fronts reflecting trust/energy trade-off.

Table 2: Opinion, latency and energy cost values for the best desired schedule

	Trust	Distrust	Uncertainty	Latency (s)	Energy (mJ)
Best Desired Schedule	0.432	0.196	0.354	325.9	53909.5

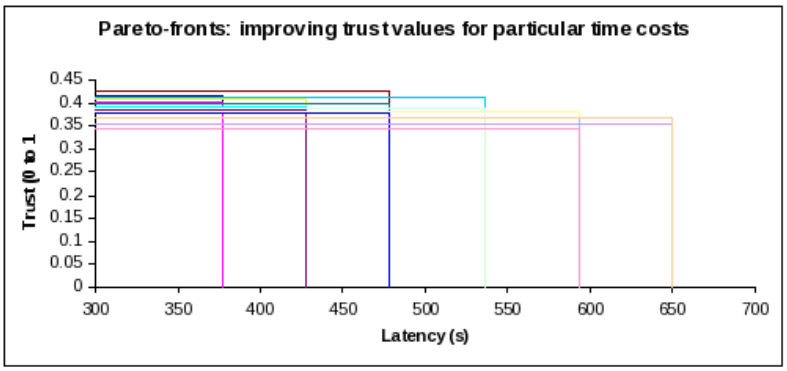

Figure 46: Pareto-fronts showing improved trust values for same latency.

We observe that the optimizer has an opportunity to increase trust and decrease energy cost while maintaining reasonable latency. This motivates further research on various scenarios where efficient and trustworthy schedules can be obtained.

5.5 CONCLUSION

In this chapter, we propose a novel approach to efficient trust-aware routing in data intensive sensor networks with trust metrics based on subjective logic. Our approach compliments

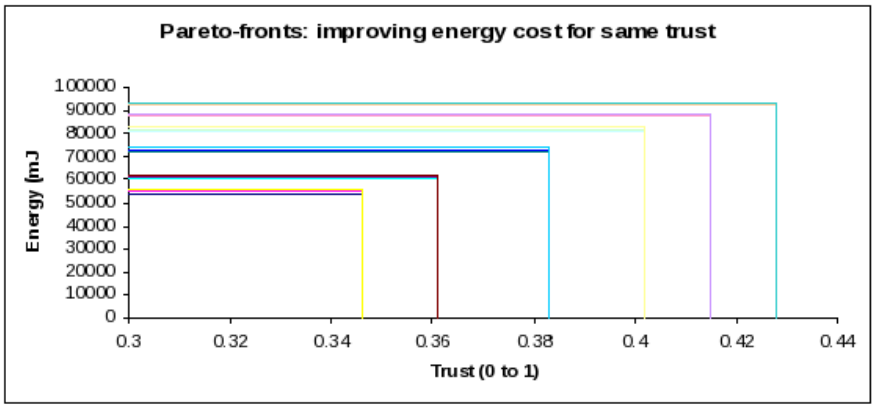

Figure 47: Pareto-fronts showing improved energy cost for same trust values.

DTA based time/energy sensor query optimization and can provide a basis for design and deployment of highly reliable data intensive sensor networks.

6.0 RELAXED OPTIMIZATION TECHNIQUES

Algebraic optimization as explained in previous chapters, to some extent, relies on global knowledge about the sensor network and requires a central coordinating node to perform the optimization. As it was mentioned earlier, this approach can be efficiently applied in medium-scale special-purpose networks (e.g., SHM networks). Although, DTA optimization can be implemented in a more distributed way, its pereformance will inevitably degrade as the size of the network grows.

In this chapter, we explore less centralized and more local-knowledge dependent optimization techniques that build on the following principles:

1. **Localized Decision Making**

   Localized decision making in WSNs is based on interactions between sensors with in a restricted vicinity. Estrin et al [21] pointed towards the significant robustness and scalability advantages in designing applications using localized algorithms. We propose to design a system that uses the local knowledge but collectively achieves a desired global objective.

2. **High Capacity Network Design**

   In his lecture notes [11], Hari Balakrishnan identfied three main ideas that should be used as principles while designing a high-capacity wireless network: *make every transmission count*; *control erroneous data transmissions*; and *maximize concurrent data transmissions*. He further states that a complete system design of a wireless network involving all these knobs is still an open question, which we attempt to solve here.

The previous chapters discussed DTA, whose main goal is to maximize concurrent transmissions while minimizing collisions. This capter starts with a discussion on decentralized de-

sign of the data delivery network. We then describe how to model a data delivery mechanism that minimizes the number of transmissions that do not lead to a useful packet transmission and resolves various network bottlenecks.

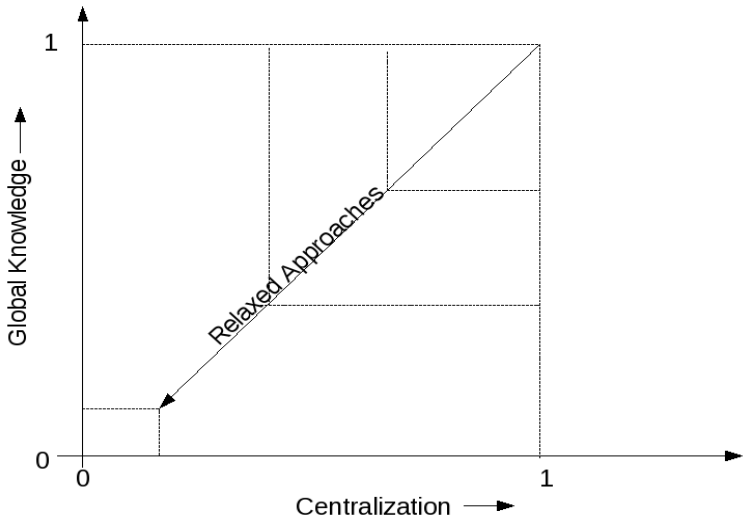

Figure 48: General taxonomy of relaxed optimization techniques

Figure 48 explains general taxonomy of the proposed approaches. We observe that DTA is associated with higher degree of centralization and global network knowledge. A more relaxed scheme would require lower degree of global knowledge and centralization. Thus, with this taxonomy, exploring the relaxed optimization scheme can be viewed as a top-down descent from (1, 1) to (0, 0) point. Here, (0, 0) point corresponds to minimum degree of global network knowledge and maximum decentralization.

6.1 MOTIVATION

It has become increasingly popular to consider WSNs as an implementation platform for data-intensive applications with high bandwidth needs (e.g., continuous monitoring of the integrity of civil and military structures, dynamic emergency assessment, disaster management, fire evacuation etc.). Meanwhile, the well-known resource constraints have a much

stronger impact in the Data Intensive Sensor Networks (DISNs), where each node is continuously sensing and delivering information to a base station. While DISNs are intended to handle large streams of data, their performance dramatically degrades with increase in network size and data rate.

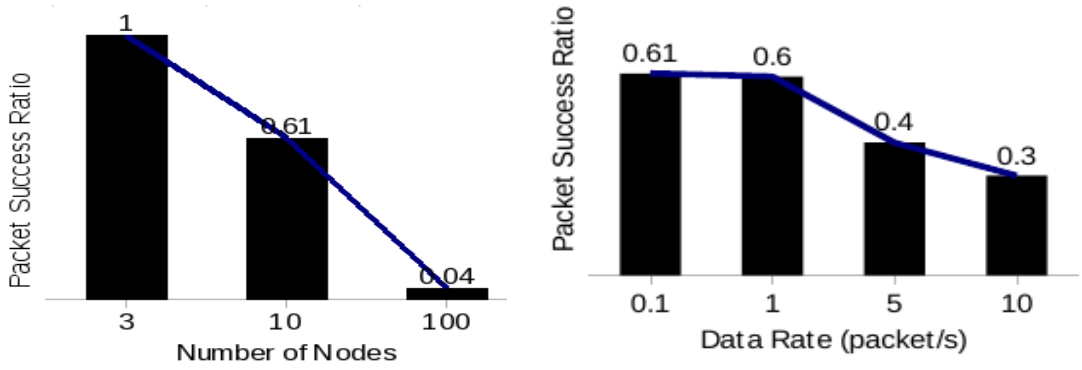

Figure 49: Left: Packet Success Ratio (PSR) for 3, 10 and 100-node networks with data rate of 1 packet/s. Right: PSR for a 10-node network when data rate is increased from 0.1 to 10 packet/s.

As an example, consider data delivery performance for WSNs with topologies of 3, 10 and 100 nodes simulated using ns-2 with IEEE 802.15.4 extension. The 10 and 100 node topologies were generated randomly. Figure 49 shows the percentage overall successful packet deliveries at the sink (i.e., ratio of total packets received successfully to the total packets sent by the sources, referred to as PSR hereon). In Figure 49 (Left), the case when data rate was 1 packet/s/node, PSR decreased form 100% packet success for the 3-node network to only 61% packet success for the 10-node network. Further, the 100-node network at the same data load could achieve only 4% of successful data delivery. Figure 49 (Right) shows PSR for the 10-node network as the data rate is changed from 0.1 to 10 packet/s. We observe a notable decrease in PSR from 0.61 to 0.3.

These observations agree with the fact that data delivery perfromance in medium to large wireless sensor networks at higher data rates degrades steeply and does not meet requirements of data-intensive applications. This also concurs with the results reported in related studies (see chapter 2). It is common for sensors in a structural health monitoring

system to generate 6-8 packet/s of vibration data. Thus, the DISN should provide more graceful degradation of the successful information delivery as the network size and data rates increase. A major challenge in achieving this objective is that the factors impacting successful information delivery in DISN are numerous and their combined effect is hard to assess for different network configurations and applications. While numerous techniques were proposed to handle data losses due to link quality degradation, congestions, route unavailability and packet collisions, it is not feasible even for sensor networks to explicitly tune up and optimize the complex interplay between these factors.

In this chapter, we introduce a light-weight adaptive approach that considerably improves performance of data intensive sensor-nets. Instead of devising a refined network cost model that would account for various data losses, we propose to optimize information delivery in DISN using a macroscopic view of the network as a complex adaptive system (CAS) [49], where simple localized decision made by individual sensors converge to a desirable information delivery pattern. Our approach is based on local data rate adaptation by each sensor with respect to the locally observable network conditions and network level power adaptation. The emergent network behavior reflects notable improvement in the information delivery performance. While exploiting this emergent behavior of the WSN CAS, we recognize that various reasons for packet drops in a sensor-net are collectively manifested as WSN bottlenecks. A bottleneck is formed by one or several sensor nodes interfering with the rest of the network, often without contributing to the successful delivery of data. In general, detecting bottlenecks is hard. It would require intensive WSN self-monitoring and may be prohibitively expensive for resource-constrained sensor-nets. Besides, even after being successfully detected, the bottleneck resolution would need even more considerable efforts in network assessment and analysis. Performing it in real time on the top of a heavily-loaded DISN is not realistic.

We demonstrate that our proposed adaptive strategy automatically detects and resolves WSN bottlenecks. Following simple adaptation logic, sensors can automatically tune-up their performance according to their contribution to successfully delivered data. Thus, our approach can efficiently recognize and resolve the network bottlenecks and dramatically improve information delivery in DISN.

6.2 BACKGROUND

The RF nature of wireless sensor networks leads to inherent uncertainty in the network behavior. The RF links can become unpredictable from time to time just because of simple changes in moisture levels in air. The RF link fades due to path loss and drops packets due to interference from other nodes in the network or single shared channel. Also, the limited buffer availability and varying data rates lead to congestion in the wireless sensor network. Below, we consider the major factors that cause this uncertainty. These factors manifest themselves with different strengths under different network configurations and data rates.

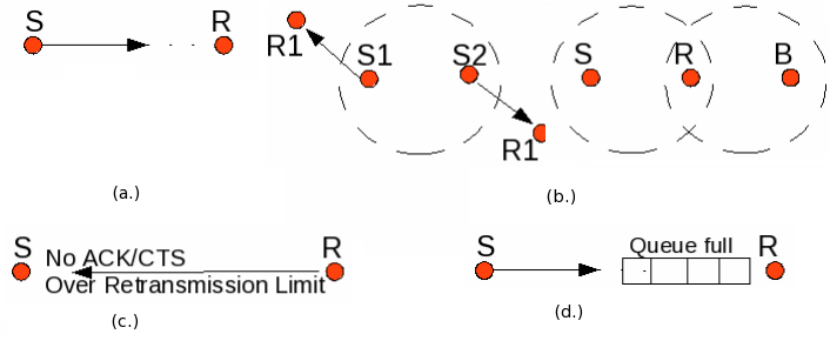

Figure 50: Various causes of packet loss (a.) Poor Link Quality, (b.) Collisions, (c.) No Route Availability, (d.) Congestion. (S-sender; R-receiver).

Link Quality Degradation. In order to perform a multi-hop data delivery, sensors identify their neighbors and maintain wireless links to them. If a sensor has low battery life, or is obstructed by a physical interference, it may not be able to maintain a reliable link. The packets sent over an unreliable link are dropped (Figure 50a).

Packet Collisions. Large number of packet exchanges between the nodes in DISN may interfere with each other (Figure 50b). Packets collide when several sensors within each other's range transmit simultaneously, or if concurrent transmissions are simultaneously received at the same destination. Packet collisions result in numerous packet drops and retransmissions. Earlier, in chapter 2, we introduced DTA that mostly handles packet collisions.

No Route Availability. Each sensor in DISN uses some routing protocol (e.g., AODV [55]) to establish and maintain data delivery routes. Sometime nodes forming a route become unavailable, or a route update turns out to be faulty. After retransmission limit is exceeded, all packets on such a route are dropped (Figure 50c).

Congestion. Sensors forming a data delivery route can exceed their buffer capacity and become congested under high data rates. Congested sensor nodes start dropping packets causing additional data loss (Figure 50d).

Complex interplay of above factors dramatically impact successful data delivery in DISN. Next we discuss their combined effect in more details for different network topologies and data rates.

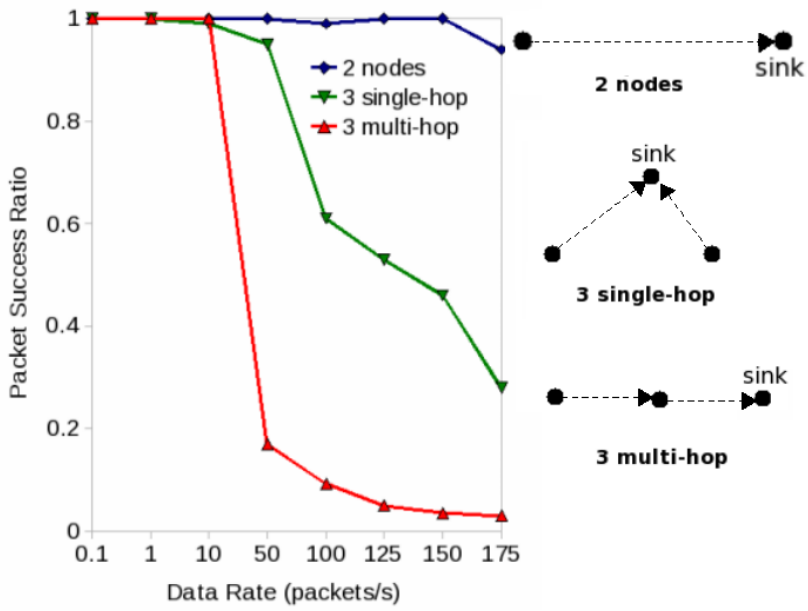

Figure 51: Comparison of Packet Success Ratio for 2 and 3 node topologies at various data rates.

First, consider a trivial network consisting of only two nodes: source and sink. We simulated continuous data delivery for this network in ns-2 using 802.15.4 standard in a non-beacon enabled experiment, with AODV routing, a transmission range of 15 m for the nodes, and a packet size of 70 bytes (or 70*8 = 560 bits). With the 2.4 GHz band 802.15.4 which supports 250 kbps, theoretical data rate of 446 packet/s (i.e. 250000 bps/560 bps)

can be handled by the channel. In reality, the available bandwidth is much lower. Figure 51 shows the packet success ratio achieved for the 2-node network at various data rates. We observe that the PSR remains equal to 1 up to around 160 packet/s, after which it begins to decrease.

None of the causes for packet drops that are discussed earlier are responsible for the decrease in PSR. The sink is within the range of the source with no physical interference in the simulated network. The source is the only node generating packets, there are no collisions-based losses. The route is a simple one-hop, which accounts for the lack of no errors due to unavailable routes. Moreover, since the sink has an infinite buffer, there are no congestion drops. Hence, the only reason for the PSR decrease is the fact that the channel capacity has been exceeded at this high data rate. Thus, the realistic bandwidth threshold available over a single wireless link for our experiments is 160 packet/s. After 170 packet/s, data drops become noticeable.

Compare this result with the PSR dynamics reported in Figure 49. We observe that even data rate of 1 packet/s is not handled well by the 10 and 100 node networks, while the channel can handle up to 160 packet/s. This is because of added topological complexity and resource constraints (e.g., limited buffer, difficulty in maintaining routes) in larger multi-hop networks, as we illustrate in the following example.

Consider two 3-node topologies: (1) with two source nodes delivering data to the sink node using a single hop, and (2) with one source node delivering data to the sink using an intermediate hop. Figure 51 reports on PSR for both topologies at various data rates. We observe that for the single hop topology the PSR starts decreasing at around 10 packet/s.

The distribution of causes for this drop is shown in Figure 52 (Left). As the data rates increase, the routes become unavailable and congestion losses increase. The link quality and collision losses seem to decrease at higher rates, since fewer packets are routed through the network because of the high unavailable route and congestion losses.

We also observe in Figure 51 that the decrease in the PSR for a 3-node single-hop topology is more gradual compared to the multi-hop topology, where PSR drops sharply after 20 packet/s. The distribution of losses in Figure 52 (Right) shows that the congestion losses were the significant cause of this drop (please refer to the scale on y-axis).

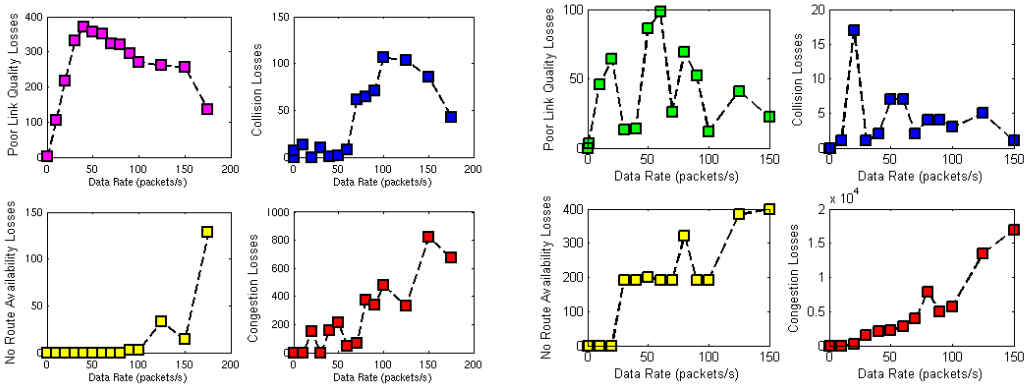

Figure 52: Left: Packet loss for a 3-node *single-hop* network at various data rates. Right: Packet loss for a 3-node *multi-hop* network at various data rates.

Thus, even a simple 3-node multi-hop network fails really fast under higher data load. This illustrates one of the major reasons why implementing data-intensive applications using a large multi-hop wireless sensor network is difficult: *Explicit optimization of the combined effect from different factors impacting the data losses is problematic.*

6.2.1 Bottleneck nodes and their impact on information delivery

A bottleneck is formed by one or several sensor nodes interfering with the rest of the network without contributing to successfully delivered data. Consider again the 3-node multi-hop topology from the previous section with the data delivery success decreasing sharply after 20 packets/s. In Figure 53, we show the per-second-reception-footprint of packets that were successfully delivered to the sink from each node of the network for the duration of the experiment. The top-left plot in Figure 6 shows us that at the data rate of 20 packets/s, some packets sent by node 2 could not reach the sink and were lost.

When we compare this footprint to the losses reported in Figure 52, we see that collisions and poor link quality are major reasons for the PSR decrease. Meanwhile, comparing the footprint for 20 packets/s with the footprint for 30 packets/s (Figure 53), we see that hardly any packet was received at the latter data rate. This also shows that as the drops due to

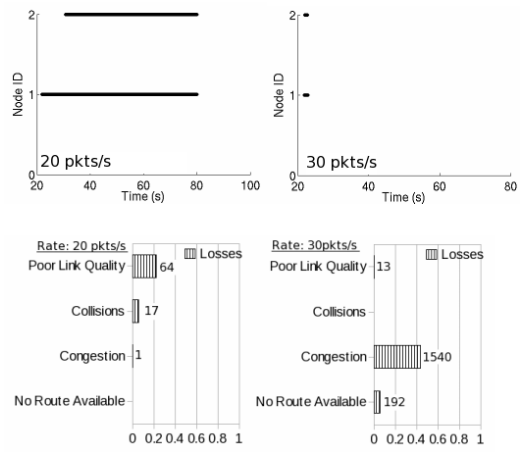

Figure 53: Left: Reception footprint and packet losses at 20 and 30 packets/s for 3-node multihop topology.

unavailability of route increase, the congestion losses suddenly become huge. It is likely that as the data rate increased, node 1 could not handle its own data along with the data coming from the node 2, and the network congestion jammed the route discovery. This, in turn, contributed to further losses due to network congestion. With respect to the terminology used in this chapter, both node 1 and node 2 are bottleneck candidates responsible for the performance degradation.

With a larger network of 21 nodes (Figure 54) we observe at data rate of 1 packet/s the PSR of this network is 0.19. The major losses are due to collisions, poor link quality and route unavailability. There is no congestion in the network at this data rate. The network footprint for all packets received by the sink is shown in Figure 54. The packets were sent regularly at 1 packet/s for 120s by all the 20 source nodes in the network. A careful observation of the plot shows that nodes 1, 2, 5 and 9 seem to have delivered most of their data. Nodes 4, 17, 18, 19 and 20 did not deliver any packets to the sink, while nodes 6, 7, 8, 10, 11, 12, 13, 14, 15, and 16 delivered some of their data.

In order to understand better what exactly is happening in this network, below we show a complete routing tree generated from the network trace (Figure 55). We observe that

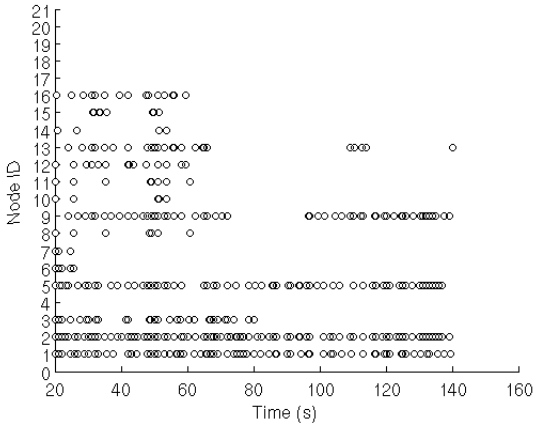

Figure 54: Receive packet footprint for 21-node topology at 1 packet/s.

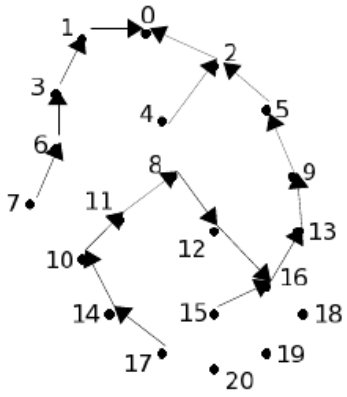

Figure 55: Routing tree for 21-node network. (Notice that the packets from node 8 were routed through node 12).

nodes 18, 19 and 20 could not find a next hop. For node 8 it seems natural to route data through node 4. However, the network conditions (e.g., node 4 is always occupied by data from node 5 that is coming at a high rate) must have forced node 8 to find route through node 12. We can deliberate that since node 8 diverted traffic to node 12, it also caused a lot of traffic through node 16. Meanwhile, since there was congestion in the network, the route

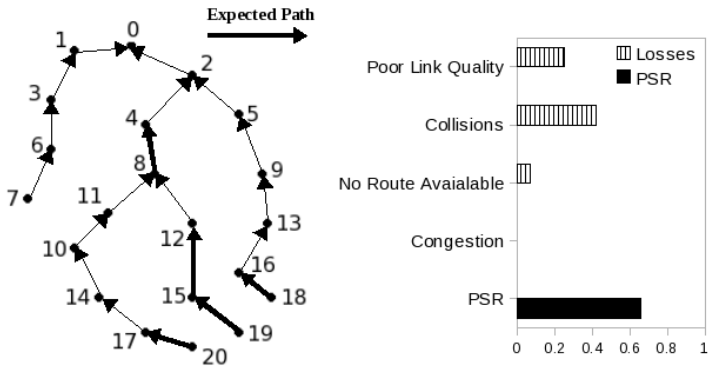

Figure 56: Preferred constructed routing tree for the 21-node network.

discovery for nodes 18, 19 and 20 failed to find a route to the sink.

One way to resolve bottlenecks is to provide better routing. Figure 56 shows an improved routing tree that was constructed manually. Here, we reduce load on node 16, and thus resolve the bottleneck node 8. The dark arrows show the preferred expected new routes that should be discovered to resolve the bottlenecks. We experimented with this network, and we found that the PSR indeed increased from 0.19 to 0.66 (Figure 56). We notice that the losses in Figure 54, where PSR is low (0.19), represent a much larger fraction (i.e., 81%) of packets that were successfully delivered compared to the losses in Figure 56 where PSR is 0.66, and only 44% of packets are dropped. We also observe that losses due to routes being unavailable decreased due to successful bottleneck resolution.

To sum up, after resolving the bottlenecks, PSR improves dramatically. However, resolving bottlenecks *manually*, as illustrated above, is not efficient for large networks. Further in this chapter, we will elaborate on how the network performance can be improved by using our adaptive strategies that are light-weight, can be easily built into the DISN and automatically detect bottlenecks and resolves them.

In the next section, first we discuss the works related to solving the data delivery problem in wireless networks and WSNs, in general and with rate adaptation techniques.

6.3 RELATED WORK

Rate adaptation is a proven technique used in high capacity wire-line modems [45]. Researchers have attempted to solve the low data reliability problem in wireless networks using 802.11 standard, by implementing rate adaptation techniques. It has been an active area of research [69], [44], [39], [46], [33] and [60]. Most rate adaptation mechanisms using 802.11 standard maintain previous packet success and failure reports to design strategies to dynamically adapt the physical layer transmission rate to optimize the throughput. ARF [46] uses a heuristic to predict the channel quality based on past transmission success and failure records. RBAR [33] uses RTS/CTS to get immediate feedback from the destination to evaluate the quality of the channel and determine the appropriate transmission rate. OAR [60] opportunistically transmits multiple packets back-to-back at a high data rate during clear channel periods. Robust Rate Adaptation [69] uses a two-state Markov channel. However, not many rate adaptation mechanisms have been proposed for Wireless Sensor Networks using the IEEE 802.15.4 standard. In [40], adaptive energy reduction schemes have been proposed based on rate adaptation. Pushback [43], a MAC layer solution, considers the time-varying nature of the wireless sensor network and uses a Hidden Markov Model-based scheme in order to predict the future channel conditions.

We propose a decision-centric dynamic rate adaptation mechanism. Rate adaptation in our case is a two-stage local process for each sensor node, comprised of local network state identification and local rate adaptation. These components in our research bear a similarity to RRAA [69] for 802.11. In RRAA, a lost-frames based metric was used to estimate loss in the network. In our research, we use a simple, accurate, localized metric, local-Packet Success Ratio, which gives us a direct evaluation of network behavior in the node's neighborhood. Our dynamic rate adaptation approach is a step towards designing high-capacity wireless sensor networks. A high-capacity network should be able to make every transmission count, increase parallelism in data delivery and reduce errors [11].

Our technique uses a local information-based logic to generate optimal Markov Decision Process (MDP) based policies at each node to transmit packets when it is more likely for them to deliver the data. Our approach rewards the actions (rate changes) that maintain a

good state for the node. The sent-packet footprints also show that rate adaptation introduces some asynchronous behavior, which may generate opportunities for concurrent data delivery. Furthermore, learning the state-transition probabilities of the network may map the Signal-to-Noise ratio in the network. At high noise levels we would expect more transitions from a good state to a bad state. Conversely, the transition probabilities would reflect the noise level and generate another optimal policy.

6.4 HANDLING UNCERTAINTY IN WSNS

As we explained in section 6.2, explicit optimization of the combined effect from the different factors impacting the data losses (e.g., link quality degradation, collisions, congestion and route unavailability etc.) is problematic. The initial study in Section 6.2 tells us that there are few characteristics of the wireless sensor networks that we can exploit to decide on a reinforcement learning [57] based solution. For example, congestion and route unavailability problems, to some extent, can be handled by rate adaptation; and link quality and collision problems can be handled by setting transmission power and/or generating collision-free schedules. We also know that in order to resolve congestion, if the data rates are too high, we may need to decrease them; and in case of link quality degradation we may want to increase the transmission power. So, even though we do not have the perfect mapping to optimal solutions, at least we have a little bit of supervised learning. There is a recursive nature to the wireless sensor network problems e.g., at each step of data delivery a node experiences the same questions: what data rate should be used to send data?, and what should be the transmission power? These questions remain the same even if the step-sizes change. We found that MDPs with their associated state-action rewards are likely superior in such cases, because at each stage they may learn to answer the question better.

MDP aims to influence the behavior of a dynamic probabilistic system. An MDP model defines system states, actions, state transition probabilities and expected rewards for actions performed in specific states. The decisions on choosing an optimal action in a current state are made at time points called decision epochs. Choosing action a in state s results in

a reward *r(s, a)*. System state s_i at the next decision epoch depends on the transition probability distribution $p(s_i|s, a)$. A decision policy prescribes action selection at each state at all decision epochs. Depending on whether the number of decision epochs is finite or infinite, the model is finite horizon or infinite horizon correspondingly. For the infinite horizon model it is common to consider stationary policies that apply the same decision rule to choose actions at each epoch.

Commonly MDP assumes that an immediate reward *r(s, a)* is more valuable than future reward resulted from expected sequence of state transitions. Hence, any future rewards accumulated by a decision maker, is discounted by a factor. A core problem in MPD is to find an optimal policy that maximizes the expected discounted reward and there is a standard set of algorithms [57] to calculate such policy (i.e., to solve MDP model).

MDP has been used to decide on actions in uncertain environments, where the systems operating in that environment do not have total effective control and a central control if often not robust. For example, MDP has been used to fly autonomous helicopter i.e., without a human pilot [53]. A helicopter equipped with sensors that record its position and orientation say 10 times/s can input this to a reinforcement learning algorithm, which then outputs a signal to the control-stick for proper movement in order to maintain a good flying state of the helicopter. Every time the control stick action leads helicopter to be in an unstable state the reinforcement learning algorithm gets a negative reward and corrects its action. A system may be in an uncertain environment for a very long time and may require a sequence of actions to be taken throughout its life as it transits through various states. With reinforcement learning, over time, such an algorithm learns to maintain a stable flying condition by taking a sequence of actions that maximize the accumulated rewards.

We use the packet success ratio as the metric that decides on the system states. *Packet success ratio* represents the percentage of transmitted data packets that reach the sink node successfully. Since, this ratio ranges from 0 to 1, the number of states that a node can be in is intuitive and relatively small when compared to the time horizon. As we adapt the data rate, we use a percent increase or decrease in data rate as the actions for the MDP. Hence, the number of actions are also finite, intuitive, and relatively small. In general, MDP is not a good choice when the number of states and actions are large because it becomes

computationally very difficult to solve the MDP. However, in our case with a small set of states and actions, a solution based on solving MDP suits well.

We believe that such a rewards-based reinforcement learning will help the nodes to make transmissions that they believe would count, minimize the congestion and route-unavailability errors. When used along with transmission power adaptation and on top of a collision-aware mechanism it can lead to an effective data delivery mechanism in wireless sensor networks.

6.5 FORMALIZING DATA DELIVERY IN WSN AS AN MDP

The main idea of our approach is to have sensor nodes adaptively tune up their information delivery according to their contribution to successfully delivered data. As a result we expect that sensors will transmit only if they have some confidence that the data will be delivered successfully. Tuning the data rate at each sensor in this way can detect and mitigate the bottlenecks manifested by the combined effect of the factors impacting the data losses (Section 6.2).

Our localized rate adaptation mechanism is performed at each sensor node in two steps:

1. State estimation: each sensor estimates the state of the network based on some local information readily available at the sensor.
2. Rate adaptation: based on the locally estimated state, sensor decides when and how to update data transmission rate.

Below we consider each of these steps in detail.

6.5.1 State Estimation

Each node maintains a locally estimable metric that reflects the behavior of that node. A node keeps track of the number of packets it sent, and the number of packets that were actually delivered to its one-hope neighbors in a short-term window called a cycle. The ratio of the locally delivered packets to the packets sent by a node within a cycle is called its *local-*

PSR (l-PSR). Thus, l-PSR gives sensors an estimate of losses in each node's neighborhood within its cycle.

The expectation is that a sufficiently low (or high) l-PSR would indicate bad (or good) network PSR. Localized data rate adaptation maintaining reasonable l-PSR should improve the overall information delivery performance in DISNs. Here, we have to address two issues: (1) how can we define the state space reflecting notable changes in l-PSR, and (2) what should be the cycle length.

It is obvious that l-PSR of 1 indicates that node is in a *good* state, while l-PSR of 0 indicates a *bad* state for the node. For effective rate adaptation, we add more granularity to the state space. The l-PSR state thresholds should be chosen to minimize the possibility of data loss in the next transmission during the next cycle. We experimented with different l-PSR ranges as candidate system states. Table 3 shows an example of one of the state spaces that we also maintained in our experiments.

Table 3: States of a sensor node based on its l-PSR.

PSR	**State**
>0.8	State 1: highly functional
>0.5 and <=0.8	State 2: moderately functional
>0.3 and <=0.5	State 3: hardly functional
<=0.3	State 4: non-functional

The cycle length is a critical parameter. It controls how often localized rate adaptation should take place. It depends on our expectation of when we want the adaptation to begin, and the current data rate of the system. The length of the cycle can be specified in terms of number of packets transmitted by a node, after which it will decide on adapting transmission rate for the next packet. If we expect an early adaptation, then a small cycle length (say 1 packet) would help the network to adapt better. Meanwhile, if the data rate is high (10 packets/s), then it would not make sense to adapt the rate for each packet. In this case the cycle length may be set longer (e.g., 100 packets).

6.5.2 Rate Adaptation

1. **Adaptation Actions**

   At the beginning of each cycle, sensor node estimates its state by checking its l-PSR in the previous cycle. Depending on the estimated state the node makes a decision on whether to decrease or increase the interval between the packets, which corresponds to a respective increase or decrease in the data rate. Thus, each sensor performs adaptation actions depending on its estimated state. Table 4 lists one possible set of the adaptation actions. The actual value of the data rate decrease or increase is a tuning parameter for a particular adaptation action. We experimented with different set of actions and the results are reported in Section 6.7.

Table 4: Five possible adaptation actions for a node.

Actions	**Change in Interval**
=	Prev. Interval * 1
↑1%:	Prev. Interval * 1.01
↑5%:	Prev. Interval * 1.05
↑20%:	Prev. Interval * 1.2
↑30%:	Prev. Interval * 1.3

Assuming that sensor is in one of the four states listed in Table 3, it needs to choose a proper action from Table 2. The choice of an action for a particular state should be based on its potential impact on the successfully delivered data. For example, if the node is highly functional (state 1), then, action 1 (i.e., '=') should probably be preferred over ↑1% or ↑5%. Meanwhile, sensors in a non-functional state (state 4) would better perform more conservative action like ↓20%, or even ↓30%. Specific choice of actions would depend on the probability of that action to change the node state from a less functional to more functional state, and vice-versa.

Our localized rate adaptation can be represented as an infinite horizon MDP model with sample states and actions from Tables 3 and 4. An MDP model can be solved to find a

state-action policy that accumulates highest reward. Choosing appropriate rewards for states and actions, we can tune the policy selection process to achieve desired performance targets. Solving the MDP model using a standard value iteration or policy iteration algorithms [57] we receive an optimal state-action policy. Each sensor node consults the optimal MDP policy when making decision on rate adaptation in its current state.

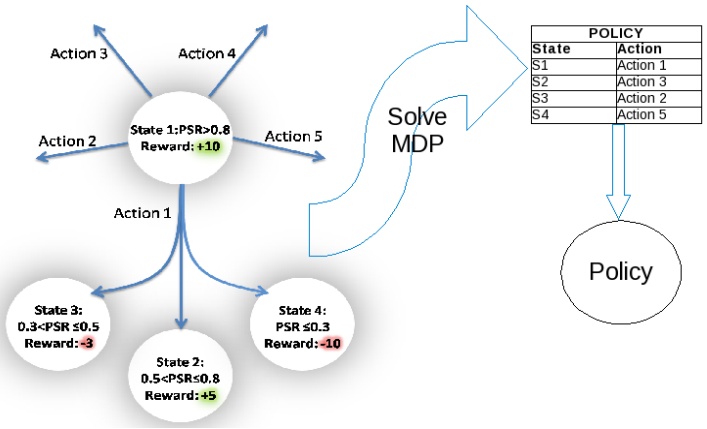

Figure 57: Part of l-PSR-based state-action diagram with a decision policy.

Figure 57 presents a part of state action diagram of an MDP model together with an example of a decision policy. Intuitively, the policy suggests to increase the packet interval for the next transmission if the l-PSR is relatively bad (i.e., less than 50% packets were successfully delivered). However, the magnitude of increase in the interval varies. For a very good l-PSR of 0.8 or better, the interval is either kept same or increased by 1%, whereas for the l-PSR below 0.5, there is a 20% to 30% increase in the length of the interval for the next packet. Hence, the data rate changes adaptively when the network in the local sensor neighborhood performs bad. If a node has low l-PSR for certain number of consecutive cycles, the adaptive interval increases. Hence, if a node(s) performs consistently bad (and thus is a candidate for being the network bottleneck), it has lower chances to transmit the next packet.

2. **Transition probabilities and rewards** In order to complete the model specification we also define probabilities of sensor state transitions under given actions. For our

experiments, we form the transition probability matrices by observing the state changes from the network traces. Meanwhile, such probabilities can also be assigned by expert for different generic network configurations and application requirements. We defined a reward matrix that assigns a reward value to a node for being in a particular state and performing certain action. In general, the reward may depend only on the state of the sensor node, or both states and action, or state transition based on the action taken. Finally, we have to assign a factor to discount future rewards accumulated by a sensor. A discount factor of 0.9 would imply that a reward now will be worth 10% less the next time when the node accumulates it.

We call this MDP based system as MDP-DRA (MDP based Dynamic Rate Adaptation)

6.6 IMPLEMENTATION NOTES

The MDP based Dynamic Rate Adaptation (MDP-DRA) mechanism is implemented on each node. When a packet arrives at the beginning of a new cycle, the node uses MDP-DRA logic to perform adaptation. Since l-PSR measures only one-hop data success, it uses the information that is readily available at the node in form of packet reception acknowledgments (ACK). Thus, it does not add any substantial network overhead. In our NS2 implementation, before a wireless sensor node transmits data to its neighboring nodes, it decides on a data rate. This decision is usually fixed in the configuration file at the beginning of the simulation for a node or the network. We extended NS-2 so that each node performs MDP-DRA logic. Each node has access to packet success information from its network trace history. The nodes use only their real-time trace history (i.e., trace beginning from the time when the previous packet was delivered by this node).

Table 5: Algorithm: MDP-DRA

Phase 1:

Input: A network $\boldsymbol{N}$, recent trace Tr, States $\boldsymbol{S}$, Actions $\boldsymbol{A}$,
Transition Probability matrix $\boldsymbol{T}$, Rewards matrix $\boldsymbol{R}$, A discount factor $\boldsymbol{g}$

Output: A policy $\boldsymbol{P}$

1. Initialize S, A, R, g and derive T from Tr
2. Solve MDP to get P

Phase 2:

Input: A node $\boldsymbol{N_i}$ with data to be delivered, recent node trace $\boldsymbol{T_i}$,
State-action policy for the node $\boldsymbol{P_i}$

Output: Data transmission rate for the next packet delivery i.e., action for the node

1. Initialize i=1
2. For a node N_i with data to deliver, calculate the local packet success ratio i.e., $l - PSR_i$ for the node N_i
3. Decide on node state S_i
4. Return an action A_i to the node N_i i.e., the data transmission rate for the next packet delivery
5. Go to i=i+1

6.7 EXPERIMENTS AND ANALYSIS

6.7.1 Set-Up

For this study, we implemented MDP-DRA on ns-2 with CMU wireless extension [3]. The MAC layer followed IEEE 802.15.4 standard [2] incorporated into ns-2 by Zheng & Lee [77]. All sensor nodes were Fully Functional Devices (FFDs) [2].

Two-ray ground path loss model was chosen for radio propagation. The wireless channel data rate was 250 Kbps. Each flow in the network was executed according to constant bit rate (CBR) traffic. AODV protocol was used for routing. Each sensor had a transmission range of 15m. The packet size was 70 bytes, and the queue length was set to 50 packets. The complete experiment length was 140s. The nodes set themselves up during first 20s and deliver data for next 60s. The sink stops receiving any data being propagated in the network after 140s. Each sensor had an initial energy of 10J. Transmission and receiving power were 0.0225 W and 0.03 W, respectively. We report experimental results for 10 and 100 node networks. The 10 node topology was in a 50mX50m area. The 100 node topology was in a 100mX100m area. All nodes deliver packets to one sink in multi-hop fashion with data load accumulation. For example, in case of 1 packet/s data generation rate, leaf node n1 transmits 1 packet to its next hop neighbor n2, and n2 transmits 2 packets which include its own sensing data and data from the previous leaf node n1 to the next hop and so on.

6.7.2 Resolving bottlenecks

First, we applied MDP-DRA on the 21 nodes network discussed in context of bottlenecks, earlier in 6.2.1. Using MDP-DRA, the network could automatically discover a route through node 8, and a route for the nodes 18 and 20. MDP-DRA helped to find and *maintain* alternate routes e.g., route from $8-4$ and $8-12$; $10-11$ and $10-7$; $11-8$ and $11-10$; $13-9$ and $13-16$; hence, giving more opportunities for data delivery. One of the routing trees obtained by using MDP-DRA is shown in Figure 58c. This route is very similar to our constructed, preferred route Figure 58b, although, it still could not find a route for packets from node 19 where no coordinator could be associated with the node. Resolving this would need more fine-tuned adaptation.

The PSR dynamics and contributing loss factors are shown in Figure 59. The PSR increased from 0.19 for CSMA to 0.73 for MDP-DRA (Figure 59), which is even more than the PSR achieved (0.66) by using our constructed route. MDP-DRA was able to decrease the drops due to route unavailability by more than 80%, but some congestion losses appeared due to higher traffic at nodes 8, 10 and 16.

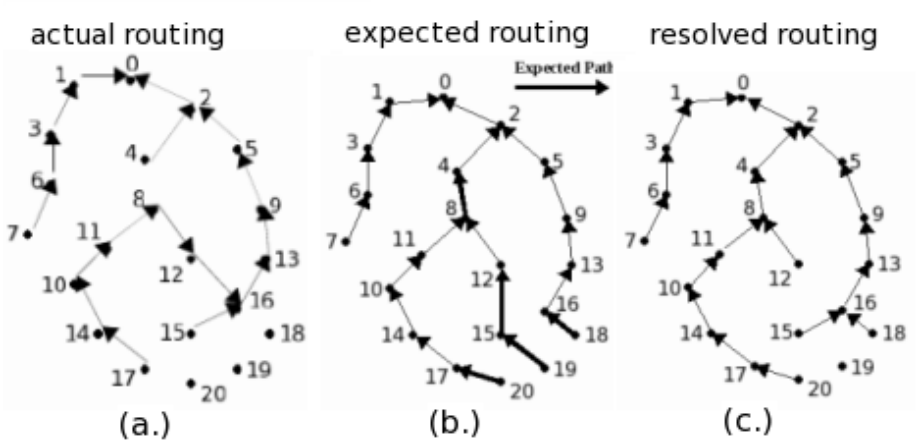

Figure 58: Effect of MDP-DRA on routing.

With MDP-DRA various nodes delayed their packets, and this facilitated route discovery and data delivery for other nodes. For example, nodes 5 and 9 (after 30s) delayed their packet delivery and that enabled nodes 6, 7, 8, 10, 11, 12, 13, 14 to deliver more data, and 17, 18 and 20 to discover new routes for themselves. Nodes 5 and 9, although they were delivering data before, yet had hampered route discoveries like $8 - 4 - 2$. Node 5 kept on delaying its packets, whereas node 9 started to deliver data normally after 60s, which shows strength of our rewards based adaptive actions policy.

We have observed similar bottleneck resolution performance in networks with 10 and 100 nodes. For the 100 node topology, we can see the received packet footprints at 0.1 and 1 packet/s in Figure 61. The left graph in each pair is the received packet footprint without using MDP-DRA, and the right graph is packet reception with MDP-DRA. We can see that more packets were received as a result of dynamic rate adaptation. The darker patches in the plot show that larger number of packets were received within small intervals of time. It implies that MDP-DRA could maintain good local PSR, and hence, could use higher data rates to deliver data.

Similarly, data reception at the sink was also better at the data rate of 1 packet/s (Figure 61.). The longer trails of the packet reception footprints indicate that MDP-DRA helped the network to maintain better reception for longer time compared to CSMA with exponential

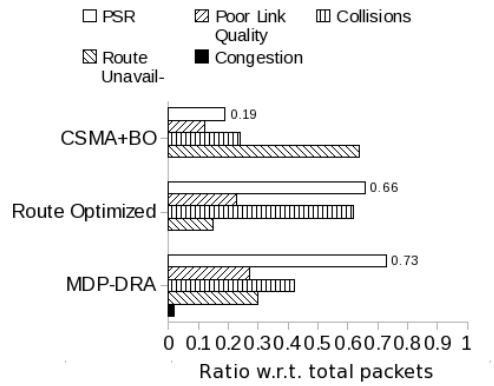

Figure 59: Effect of MDP-DRA on PSR.

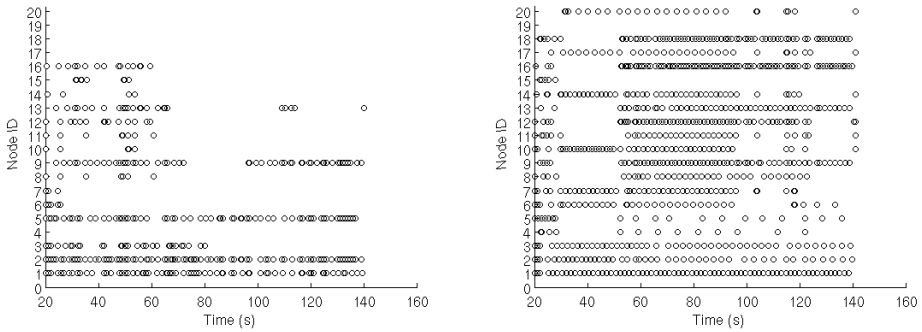

Figure 60: Data reception footprint for 21-node network (a.) before and (b.) after using MDP-DRA.

back-off (CSMA+BO).

Later, we will show (Figure 65) that at a higher rate of 10 packets/s, MDP-DRA was able to maintain the PSR of 0.3, which is more than 800% higher compared to the CSMA+BO packet success ratio of 0.03. At the same time MDP-DRA also managed to deliver more data. Figure 62 (d.) shows that while many of nodes did not deliver any data, MDP-DRA improved data delivery for the nodes that were able to route data. MDP-DRA also delayed data transmission from the bottleneck nodes (compare Figure 62(a.) to Figure 62 (b.)) by continuously increasing the interval for those packets that would most likely drop. Thus,

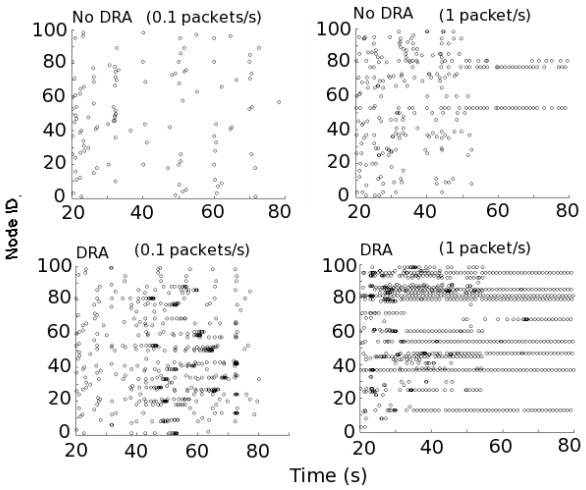

Figure 61: Pairs of data reception footprint for MDP-DRA at 0.1 packet/s (top pair) and 1 packet/s (bottom pair) for 100-node topology

we observe that MDP-DRA is able to resolve bottlenecks efficiently even for a large, dense network with a high initial data rate of 10 packets/s per node.

6.7.3 Packet Success Ratio

Next, we report on packet success ratio for basic CSMA with exponential back-off (CSMA+BO), a random rate change strategy (RandRate), where the rate is changed randomly, and our localized MDP-DRA approach (MDP-DRA) where rate change is based on an MDP policy. The packet success ratio (PSR) represents the percentage of transmitted data packets that reach the sink node successfully. The higher packet success ratio implies a better and more reliable network. In other words, the network is less susceptible to dropped packets caused by packet interference, congestion or route unavailability if the packet success ratio is high. The experiments take into account the real-time history of the trace in order to determine PSR for each node. Now, we compare results for CSMA with exponential back-off, Random Rate Change and MDP-DRA.

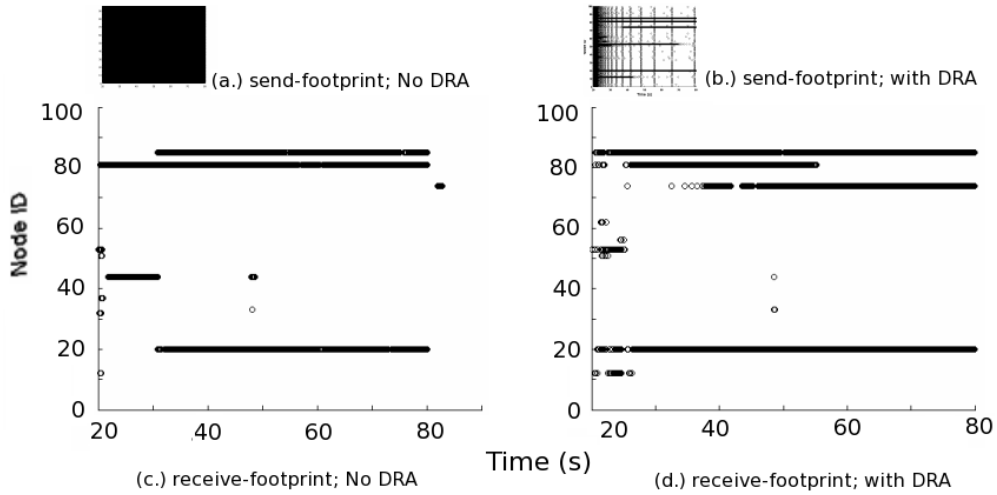

Figure 62: Send and receive footprint for 100-node network without and with MDP-DRA, at 10 pacekts/s.

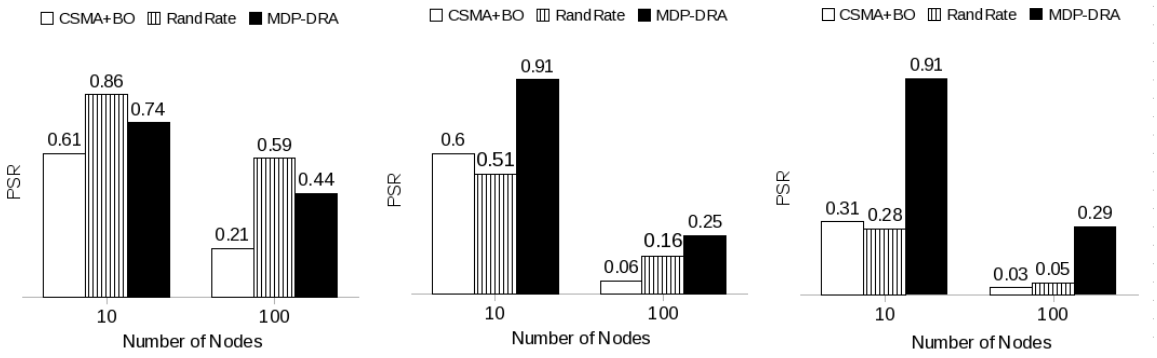

Figure 63: PSR for 10 and 100-node networks at 0.1 (left), 1 (middle) and 10 (right) packet/s

Figure 63 shows that for these set of experiments, MDP-DRA wins over CSMA+BO every time. MDP-DRA loses to RandRate at the lower data rate of 0.1 packet/s. This implies that there exists a better set of actions, which we can choose for rate adaptation. Hence, as an update over our MDP-DRA we could improve the set of actions that are used. In that case, MDP would find the optimal state-action policy corresponding to those set

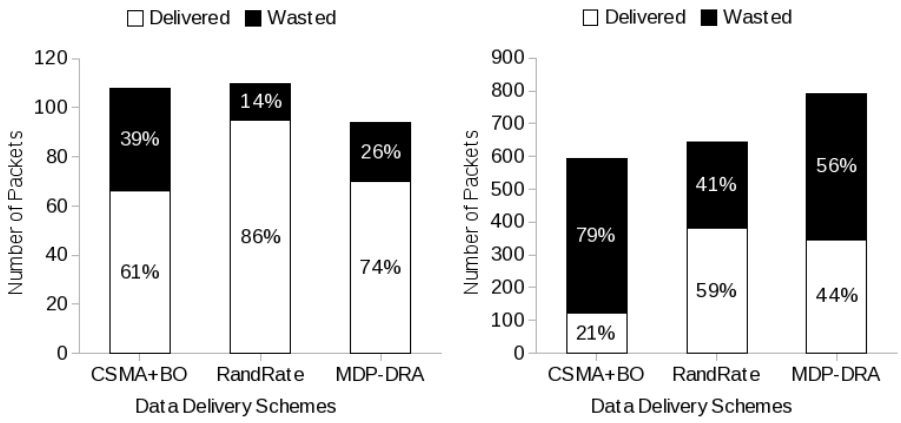

Figure 64: % data del. & wasted for 10 (left) and 100-node (right) network at *0.1 packet/s*

of actions. It should be noted that RandRate loses even to the CSMA+BO at higher data rates of 1 packet/s and 10 packet/s, whereas MDP-DRA wins over CSMA+BO every time. Hence, a systematic rate adaptation scheme is a better way to improve efficiency of data delivery. When CSMA+BO is used for simulations without any rate adaptation, we observe (Figure 16) that at 0.1 packet/s, the PSR decreases sharply from 0.61 to 0.21 for 10 and 100 nodes network respectively. At data rate of 1 packet/s, PSR is 0.6 and 0.06 for 10 and 100 nodes networks respectively. At 10 packets/s data rate, PSR is 0.31 and 0.03 for 10 and 100 nodes respectively. Hence, we observe that for CSMA+BO, the PSR decreases sharply with increase in the size of the network. With respect to the change in data rates within the same topology, the PSR follows similar trend, although the magnitude of decrease is smaller.

One of the design goals for a higher capacity wireless sensor networks is to make every transmission count [11]. In order to do so, the idea is to minimize the number of transmissions that don't lead to a successful packet transmission. Figures 64, 65 and 66 show that MDP-DRA is much better at keeping up with this goal as compared to CSMA and RandRate. We can see from Figure 59, that at 0.1 packet/s MDP-DRA delivers more and wastes lesser data than CSMA.

As the data rates increase (Figures 65 and 66), MDP-DRA wastes orders of less data than other approaches. At 1 packet/s, for 10 nodes, CSMA delivered 646 packets compared to

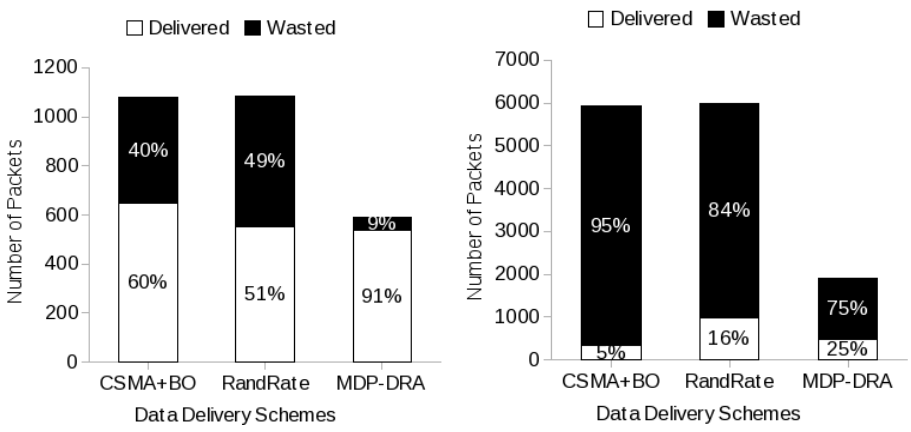

Figure 65: % data del. & wasted for 10 (left) and 100-node (right) network at *1 packet/s*

536 by MDP-DRA, however, CSMA wasted 434 packets whereas MDP-DRA wasted only 53 packets. More wasted packets also translate into more energy being used, which is precious in wireless sensor networks.

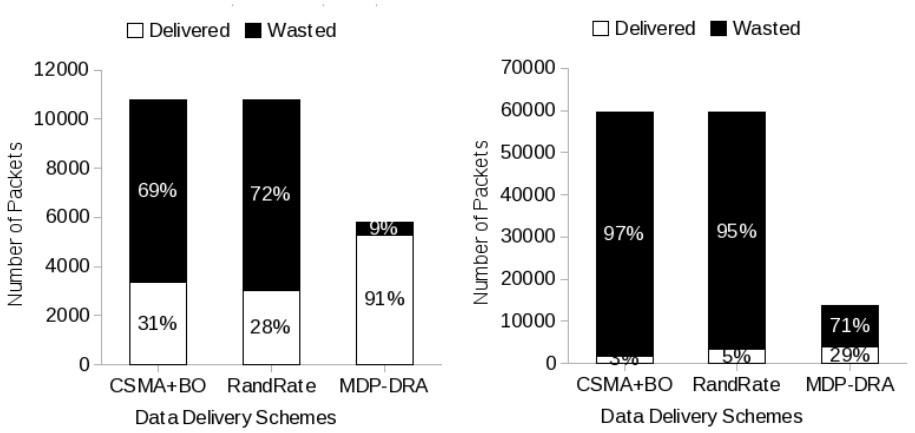

Figure 66: % data del. & wasted for 10 (left) and 100-node (right) network at *10 packet/s*

At 10 packet/s, for the same 10 node topology, CSMA delivered 3349 packets, RandRate delivered 3027 packets and MDP-DRA delivered 5288 packets. Similarly, for the 100 node topology for the same 10 packet/s data rate, CSMA, RandRate and MDP-DRA delivered 1805, 3241 and 3952 packets respectively, whereas the packets wasted were 57694, 56210

and 9747 respectively - an order of magnitude less for MDP-DRA. At times, when other approaches win over MDP-DRA in terms of the amount of data delivered, the difference between the data delivered by other approaches and MDP-DRA is not in orders of magnitude whereas the packets wasted are at least an order higher. Also, note the state definition for this MDP-DRA is PSR-based, which is a ratio of successful data delivered to the total data sent. A different state definition can be devised and would motivate better amount of data delivery if that would be the performance target.

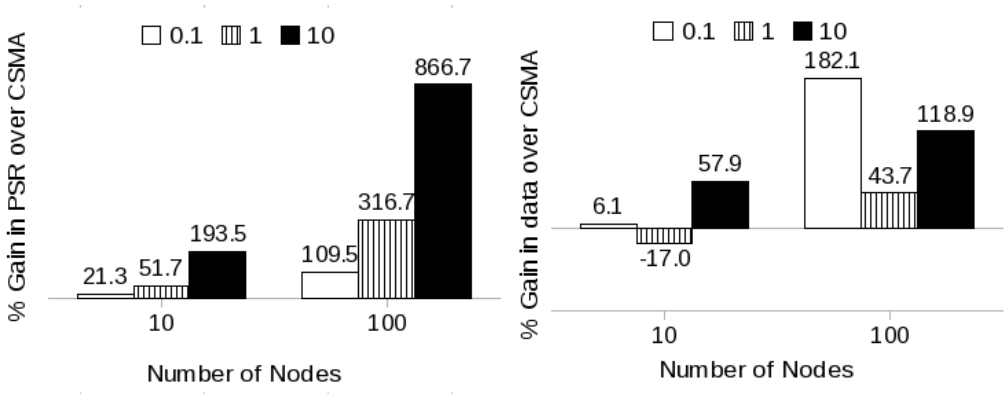

Figure 67: Gain in PSR and data delivered using MDP-DRA over CSMA+BO for 10 and 100-node networks at 0.1, 1 and 10 packet/s respectively

In contrast to the PSR obtained using CSMA+BO, our localized rate adaptations scheme (MDP-DRA) achieves substantial improvement in the PSR performance. Figure 67 provides the breakdown of % PSR and % data gain at each data rate. With respect to PSR, we infer that MDP-DRA performs better and becomes more valuable when the network size becomes larger and data rates are higher. The magnitude of the amount data gain, as discussed earlier, varies. In general, MDP-DRA considerably outperforms CSMA, by minimizing critical losses from those factors that contribute the most to the information delivery degradation.

In Figure 68, we have plotted the distribution of absolute numbers of packet drops for all the networks considered in this study. We observe that losses due to congestion almost disappear in larger networks at higher data rates when MDP-DRA is used. The link quality and collision losses increase at times for the 10-node network. These can be reduced

using appropriate transmission power adjustment. A fine-tuned MDP-based solution can be developed where a state may be defined using a combination of all these losses in the system. Then, for high link quality losses and collisions, we could tune the transmission power as well. So, we can see that this MDP-based adaptation approach is flexible and enhanceable. However, the implementation of the combined state based MDP is reserved for future work. Figure 68 also shows that the drops due to route unavailability are high for CSMA+BO. By using MDP-DRA, these drops have been substantially decreased. Note that these drops indicate that no route is available for a packet and the route discovery is adversely affected, which is often a case in larger and higher data rate networks.

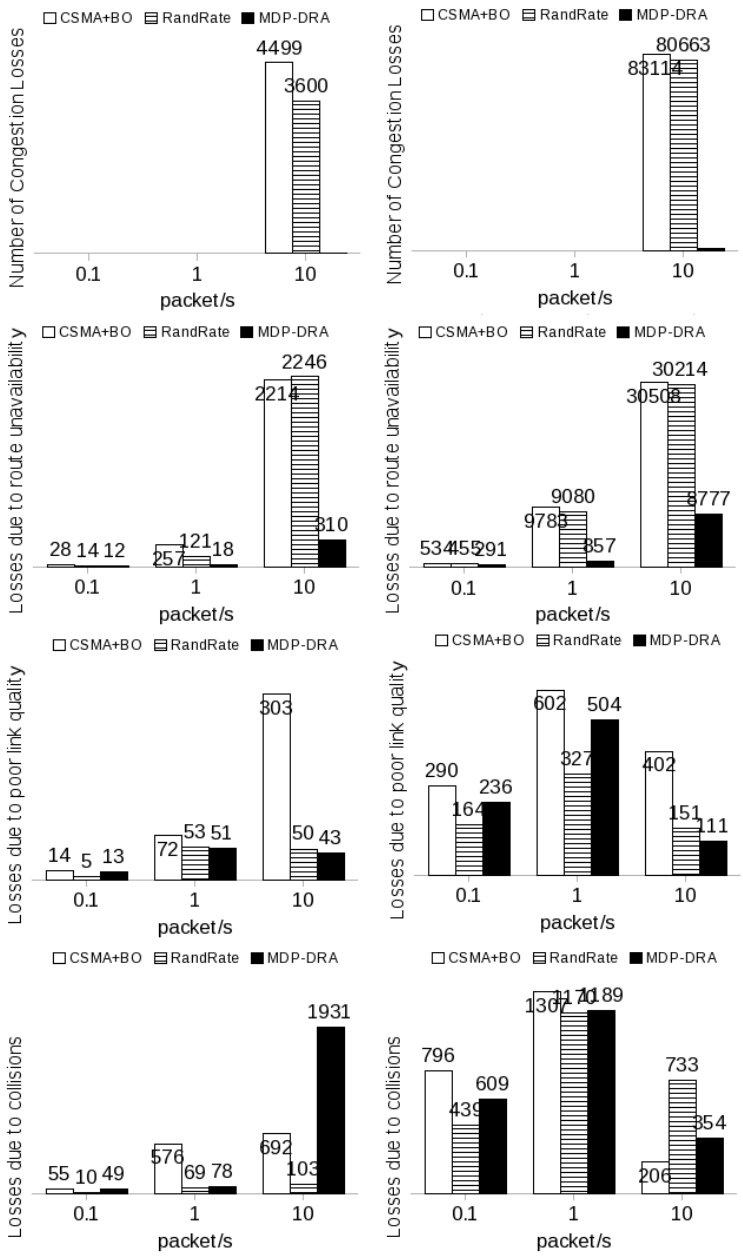

Figure 68: Comparison of packet losses for 10 and 100-node networks at 0.1, 1 and 10 packet/s

6.8 IMPROVED ALGORITHM

We modify the MDP-DRA algorithm by adding an initialization phase to it. Here, the network decides on the optimal transmit power setting for the nodes and an optimal set of actions that would be then used to generate a state-action policy for the network (Table 6).

Table 6: MDP-based Dynamic Adaptation: MDP-DA

Phase 0 (a.):

Input: Initial transmit power setting, P_{ti} and a range of transmit power settings.

Output: An optimal power setting $optP_t$

1. $optP_t = P_{ti}$; PSR_i is the PSR for the transmit power setting P_{ti}
2. while(PSR benefit >> 0), set P_t = a random power setting
3. For a number of short trials, record the PSR and calculate averagePSR.

if (averagePSR > PSR_i), $optP_t = P_t$, else $optP_t = P_{ti}$

4. go to 2.
5. Return $optP_t$

Phase 0 (b.):

Input: A range of possible actions

Output: An optimal set of actions

The action-set selection algorithm is similar to power setting selection

Start MDP-DRA

The initialization phase comprises of an iterative improvement algorithm to select power settings and/or an action-set. Short trial experiments are conducted over the network using a random-start power setting from the available set of power settings. Average PSR is calculated for the trials. If the average PSR is better than the PSR using initial power setting, then the new transmit power is set to the trial transmit power. The power settings are tweaked for each iteration depending on an improvement or decrease in the PSR. When no tangible benefit over average PSR can be obtained by any more changes in power set-

tings, the final transmit power is set. Similarly, an action-set initialization phase can help choose a better set of actions. This step also uses iterative improvement. Since, the actions we consider are percent change in data rate, we tweak the data rates iteratively to obtain network PSR for short trials, and improve the action-set.

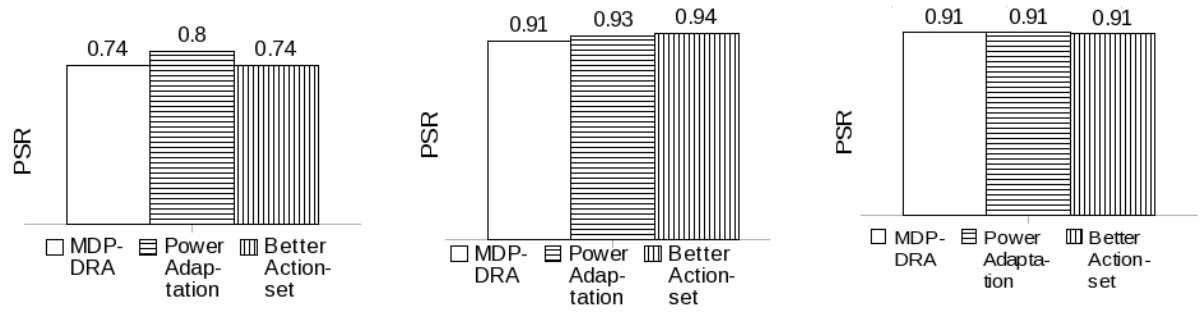

Figure 69: Comparison of packet losses for 10-node network at 0.1, 1 and 10 packet/s with power adaptation and a better action-set

Figure 69 shows the effect of power adaptation and the possibility of choosing a better action-set on the 10 nodes network with 0.1 packet/s, 1 packet/s and 10 packet/s initial data rates. The PSR increased for power adaptation for 0.1 packet/s, and for power adaptation and better actions-set for 1 packet/s. For 10 packet/s, we found that we had already chosen a better actions-set and power setting, hence, we don't see any change in the PSR for the three approaches.

6.8.1 Performance target based tuning

Different applications have different performance expectations from a DISN. In this section, we explore the possibility of tuning MDP-DRA to achieve different performance targets. Here, we also explore the possibility of increasing data rates depending on the applciation demands from the network. The tuning is based on modifying rewards associated with the states and actions.

We consider three different reward strategies and their impact on DISN performance

targets. For example, if each node were to increase the packet interval by 30% (↑ 30%) for each state of the node, the network would live longer while transmitting a low amount of data fairly successfully. We can reflect this choice of conservative actions ↑ 30% action for most states with a set of rewards that lead the MDP model to generate such a *conservative policy*. The corresponding reward function is shown in Table 7. Here, we can see that rewards create a bias to choose ↑ 30% action by assigning a relativeyly higher reward in each state. Optimal policy obtained from solving MDP with discount factor 0.9 is [↑ 30%, ↑ 5%, ↑ 30%, ↑ 30%].

Table 7: Reward matrix for a conservative policy. Optimal policy generated = [↑ 30%, ↑ 5%, ↑ 30%, ↑ 30%]

	=	↓ 30%	↓ 5%	↑ 5%	↑ 30%
Highly Functional	-0.1	-1	-0.8	0.5	0.7
Moderately Functional	-0.1	-1	-0.8	0.7	0.8
Hardly Functional	-0.5	-1	-0.9	0.8	0.9
Non-Functional	-0.8	-1	-0.9	0.8	1

We can obtain an *aggressive* target policy where aggressive actions (e.g., decrease in interval by 30% or ↓ 30%) are chosen for each state. Similarly, a *moderate* policy that chooses a mix of actions can also be targeted. Table 8 and 9 show an example of moderate and aggressive policy rewards with corresponding optimal policies.

We performed adaptation experiments for these three policies. Figure 70(a.) shows that the data delivered by the conservative policy is low, and it is high for the aggressive policy. For the moderate policy, which is the same as the adaptive policy that we have used in our experiments, we can see that in terms of data delivered, it performs as good as the aggressive policy. But, with respect to PSR it gives a much better performance than both conservative and aggressive policies (Figure 70(b.).

Table 8: Reward matrix for a moderate policy. Optimal policy generated = [↓ 30%, ↓ 5%, ↑ 5%, ↑ 30%]

	=	↓ 30%	↓ 5%	↑ 5%	↑ 30%
Highly Functional	0.7	1	0.8	-0.7	-1
Moderately Functional	0.9	-0.1	1	0.5	0.1
Hardly Functional	-0.2	-1	-0.5	1	0.5
Non-Functional	-0.5	-1	-0.7	0.5	1

6.9 MDP-DRA ON TOP OF DTA AND DRAND

In this section, we show the benefit of using MDP-DRA on top of DTA and DRAND [58]. DRAND is a distributed implementation of RAND [59], a randomized time slot scheduling algorithm. The algorithm does not require any time synchronization and is shown to be effective in adapting to local topology changes without incurring global overhead in the scheduling. DTA and DRAND are efficient in handling collisions and maximizing concurrent transmissions. However, they are not good at congestion handling and do not allow for another route discovery in case of congestion. In order to eliminate these deficiencies of DTA

Table 9: Reward matrix for a aggressive policy. Optimal policy generated = [↓ 30%, ↓ 30%, ↓ 30%, ↓ 30%]

	=	↓ 30%	↓ 5%	↑ 5%	↑ 30%
Highly Functional	0.7	1	0.8	-0.8	-1
Moderately Functional	0.9	1	0.8	-0.5	-0.7
Hardly Functional	0.8	0.9	0.7	-0.3	-0.4
Non-Functional	0.4	0.8	0.5	-0.2	-0.3

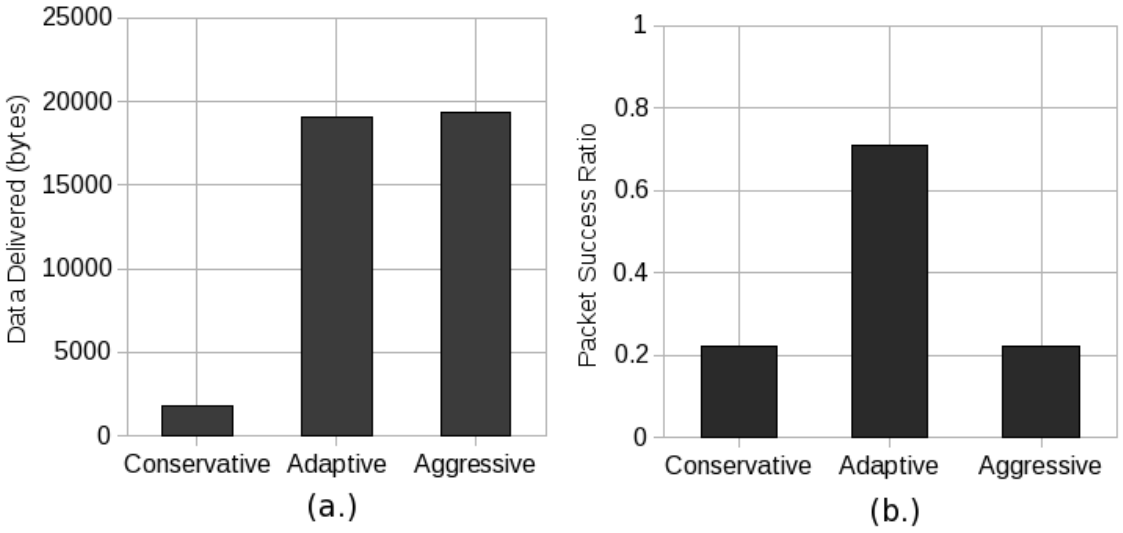

Figure 70: (a.) Data delivered and (b.) PSR, for different policies.

and DRAND, and to show the efficacy of MDP-DRA when used with other mechanisms we deployed MDP-DRA on top of DTA and DRAND.

To apply MDP-DRA with in DTA and DRAND's context, first, we consider fixed routes and time slots. We can see from Figure 71 that PSR with DTA/DRAND alone is low. When we apply MDP-DRA on top of DTA/DRAND using DTA/DRAND's original route only, the PSR benefits only a little, especially at higher data rates when congestion losses need to be compensated more. DTA and DRAND generate fixed schedules, so it may not be wise to use them in inherently uncertain wireless environment. In order to provide more flexibility to DTA/DRAND we allow for new route discovery if bottlenecks occur in the network. We call this approach *route scrambling*. DTA/DRAND already generate a collision-free schedule, which is better than seeding the scrambler with a random schedule. As we can see from Figure 71, the scrambled DTA/DRAND perform very well. At higher data rates, the new route updates may be hampered by congestion in the network, hence, adversely affecting the scrambling. MDP-DRA adapts the data transmission rates and resolves network congestion while allowing more route updates. Hence, as shown in figure 71, scrambling along with MDP-DRA achieve higher PSR. Since, DTA and DRAND are designed for collision-handling,

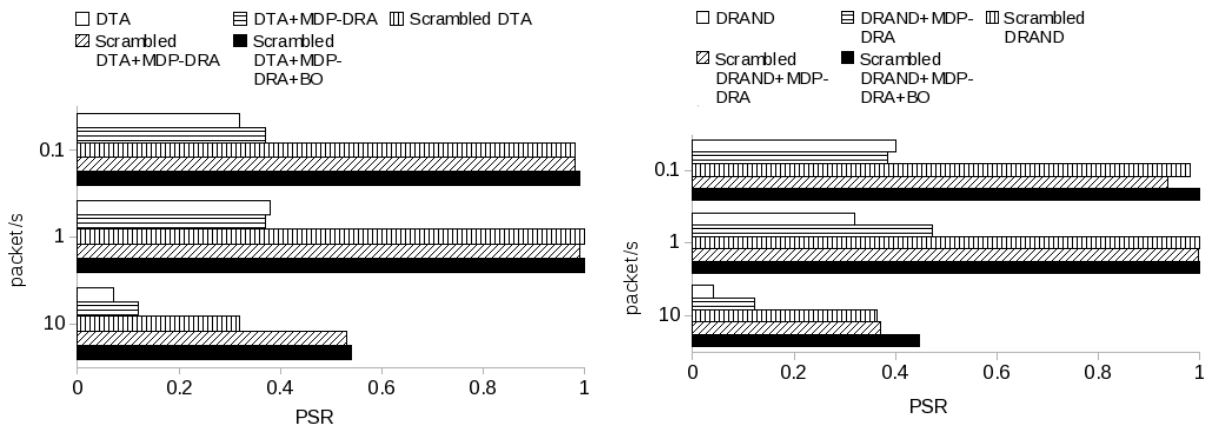

Figure 71: PSR for DTA, DRAND and improvements using MDP-DRA and route scrambling.

initially, we did not allow packet retransmissions. However, when we use scrambling and MDP-DRA along with exponential backoffs, the PSR increases even more.

Hence, we see that MDP-DRA can be used in conjunction with other data delivery techniques and it adds to the performance of network, especially, as the data rates increase.

6.10 DYNAMIC NETWORK MONITORING

To monitor the present state and the state to which the wireless sensor network may eventually converge, we developed a tool that can be used while the network is delivering data using a state based adaptive strategy like MDP-DRA. In order to do so, we capture snapshots of the network performance at various times. After getting the snapshot, the state-transition probabilities are computed for that snapshot. Then, assuming the network as a Markov system, we use an initial distribution vector to calculate the steady state vector. For the matrices that are not regular (i.e., a matrix such that all the entries of some power of the matrix are positive) we assume a transition in the highly dysfunctional State 4, to convert it into a regular matrix.

6.10.1 Background

A Markov system can be in one of several (discrete) states, and can pass from one state to another each time step (snapshot time) according to fixed probabilities. A state-transition probability matrix represents the probability (transition probability) that the system will move from one state to another, given that it is currently in one state. The sum of transition probabilities out of one state is, by definition 1. The set of probabilities is stored as a transition matrix P, where an entry *(i,j)* is the transition probability from state i to state j. A distribution vector v, is a row vector with one non-negative entry for each state in the system. If v is an initial distribution vector and P is the transition matrix of a Markov system, then the distribution vector after 1 step is the product of v and P: vP. The distribution after one step is obtained by again multiplying by P i.e., $(vP)P = vP^2$. Similarly, the distribution after n steps is obtained by multiplying v on the right by P n times i.e. vP^n If P is a transition matrix for a Markov system, and if v is a distribution vector with the property that $vP = v$, then v is called a *steady state distribution vector*, which is also the principle eigen vector of the matrix P. To find a steady-state distribution, we solve the system of equations given by

$$u + v + w + ... = 1$$
$$[u\ v\ w...]P = [u\ v\ w...]$$

where we use as many unknowns as there are states in the Markov system. A steady state probability vector is then given by

$$v_\infty = [u\ v\ w...]$$

where the vP approaches a fixed matrix v_∞. This matrix gives the long-term probability that the system will be in each state.

6.10.2 Experiments

Figure 72 shows the steady states for 30 second interval snapshots for CSMA and MDP-DRA approach after 30s, 60s, 90s and 120s for a 100-node network at 1 packet/s. The pie-charts show percent of transitions to a particular state. Transitions to state 1 (represented by

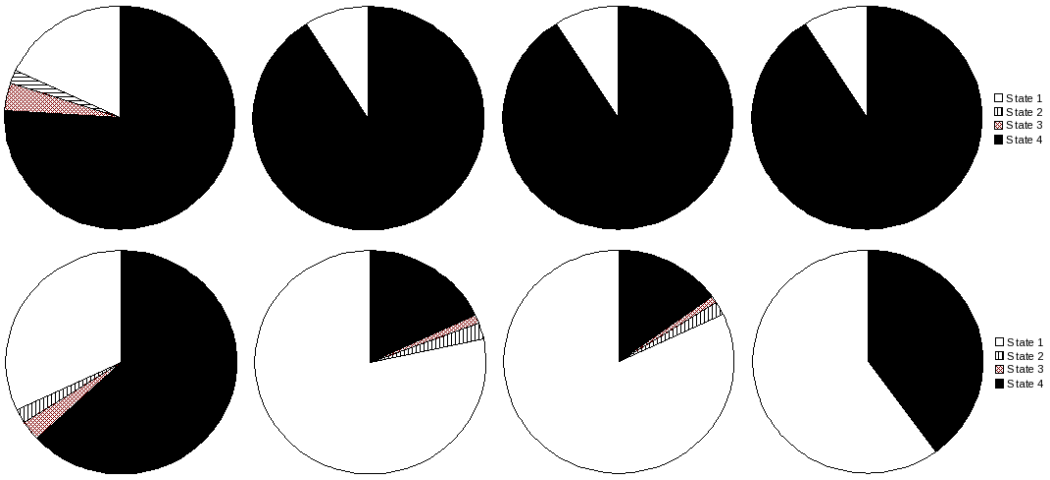

Figure 72: Steady states for 30s intervals for CSMA+BO(top) and MDP-DRA(bottom) after 30s, 60s, 90s and 120s into the experiment.

white piece of pie) are considered good and suggest that the network is converging towards a healthy state. We can see from the top row of Figure 72 that for CSMA+BO as the time progresses the network converges towards a bad state (represented by black piece of pie), as the size of the black pie increases. Meanwhile, for MDP-DRA (bottom row) as the network delivers data, the adaptive data delivery maintains a healthy network state and largely, improves it. The steady state achieved in the final snapshot decreases, however, as the network adapts from first snapshot's bad state to second and third snapshots better states, if we would let the network run for more time the reinforcement learning logic would again adapt the network back to a favorable state. A smaller snapshot (e.g., here less than 30s snapshot) can also help in reducing the amount of network time spent in a bad state.

A detailed look at the state-transition matrices after 30s tells us about the network behavior during the snapshot time. For CSMA+BO about 17% of all the transmissions entering the bad state (State 4) get out of it. In case of MDP-DRA 18% do the same. About 31% transmissions out of all the transmissions that transit from the best state (State 1) transit to State 1 itself; whereas in case of adaptive MDP-DRA 59.4% transit to State 1. Here, we already see the signs of adaptation. The steady state distribution for this snapshot

for CSMA tells that in long-term if things continue the same way for the network, 76% of total transitions would result in the bad state (State 4), and only 18% will result in good state (State 1).

Comparatively, for MDP-DRA snapshot 63% of total transitions would result in the bad state and 32% in good state. As the network delivers data in these conditions, for CSMA+BO the long term transitions in good state drop from 18% to 9% and bad state transitions go up from 76% to 91%, whereas for MDP-DRA the good state would result in 30-80% of transitions.

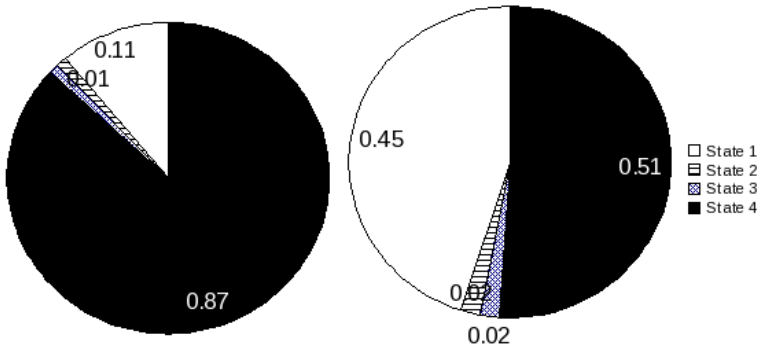

Figure 73: State achieved by complete experiment.

The steady state achieved for the complete experiment (Figure 73) shows that 87% of time CSMA transitions resulted into a transition into the bad state and only 11% into the best state where as for MDP-DRA the transitions resulting in good state were 45% and 51% resulted in the bad state. Out of all the transitions that entered the bad state (State 4), around 10% left it in case of CSMA, whereas 14% left it in case of MDP-DRA. Out of all the transitions that transited from the best state (State 1) 21% maintained the good state, whereas 81% maintained the good state with MDP-DRA. Hence, using this mechanism we could monitor the network at various check-points and predict the network behavior. Also, if needed, we could modify the actions after each snapshot to improve and converge the network to a healthy state. Next, we discuss this.

Table 10: Snapshot of steady state vectors for MDP-DRA and Multistage MDP-DRA

	MDP-DRA	**Multistage MDP-DRA**
After Stage 1	[0.4 0.05 0.05 0.5]	[0.4 0.05 0.05 0.5]
After Stage 2	[0.57 0.02 0.01 0.4]	[0.6 0.04 0.02 0.4]
After Stage 3	[0.85 0.01 0.02 0.12]	[0.83 0.02 0.02 0.13]
After Stage 4	[0.84 0.04 0.02 0.1]	[0.88 0.01 0.01 0.11]

6.10.3 Multi-stage MDP-DRA using Dynamic Network Monitoring

Using dynamic network monitoring, we can influence the behavior of the network by modifying the actions while the network is running. After we get the first snapshot and compute its convergence, the steady state vector gives us an idea to be conservative or aggressive in our approach. For example, in Table 10 after stage 1 the Multi-stage MDP-DRA chooses another optimal policy based on the new transition probabilities. Based on that we can see that at stage 2, the steady-state vector converges to a better state-distribution compared to MDP-DRA. We maintain the same policy for Stage 3, however as the network performance becomes worse than MDP-DRA, we again learn new transition probabilities and force the network to converge to a better state.

6.10.4 Conclusion

In this chapter, we discussed a decentralized design of the data delivery in wireless sensor networks. We described an MDP-based data delivery mechanism that minimizes the number of transmissions that do not lead to a useful packet transmission and resolves various network bottlenecks. We compared and used it along with other approaches, and showed its efficacy. We showed that this approach could be improved using power and action-set optimization and also used it to do performance target based data delivery. We also proposed a steady-state vector based network monitoring model, and used it to improve our MDP-DRA approach.

7.0 CONCLUSION, DISCUSSION AND FUTURE WORK

In this thesis, we proposed approaches for efficient data delivery in DISNs. First, we devised a novel algebraic framework (DTA) for collision-aware non-conflicting concurrent data delivery. We showed that this framework could optimize latency, energy, data and extended it to optimize the level of trust associated with the schedule. We improved DTA's scalability with a sectored rotating interrogation strategy called Whirlpool. As the network size grows, generating optimal DTA schedules becomes challenging. Also, congestion problems in larger DISNs jam data delivery. In order to design high capacity larger scale DISNs, then, we proposed relaxed optimization strategy that dependeds largely on local knowledge available at each node. From our experiments with random rate changes, we realized that a non-linear rate assignment helped generate more interleaved data delivery pattern, which in turn led to fewer bottlenecks in the network. We found that a Markov Decision Process (MDP) could be used to generate a more diverse rate assignment, where the change in rate assignment is motivated by the state of the node in the network. Even a static MDP i.e., with a single policy for the life of WSN performed better than other approaches. In each experiment, MDP-DRA approach on an average performed much better than the achievable rate $\Theta(1/n)$ (protocol model) [50] for arbitrary networks, or $\mathcal{O}(W/\sqrt{nlogn})$ [26] for random ad-hoc networks for smaller networks, and better than average capacity, though not necessarily always fair, for larger networks. MDP-based approach also benefits by alleviating the need for back-propagating the congestion message on a congested route. It is a good idea to let the nodes in a route know that there is congestion in the network ahead. But, in our approach, whenever a node gets congested, its local-PSR decreases and leads to a decrease in its rate. While this node sets into problem resolution mode, the nodes waiting to deliver to this node drop their packets and modify their data rate. This automatic con-

gestion detection and resolution process continues to backtrack till the nodes are able to maintain a stable state again. Hence, this MDP-based process does not need to explicitly back-propagate a congestion alarm and starts to self-heal at the moment it faces a problem. However, a static MDP assumes that the state-transition probabilities in the network do not change, and transition probabilities as well as associated rewards assignment do not have any uncertainty. Since, this is far from reality [18], an MDP would eventually suffer. Hence, we extended MDP-based rate adaptation with a multi-stage MDP where the state-transition probabilities are updated for various network snapshots and new optimal policies are generated at each snapshot. We also demonstrate that choosing a specific power assignment for the network has effect on data delivery. We proposed an improvement over our MDP-based dynamic rate adaptation by adding an initial phase that searches for an optimum transmission power level for the network. From our experiments, we have realized that an optimum power assignment at node level would be more beneficial. In order to do so, our future work would include local collisions and link quality along with PSR to devise a new state for each node. A reinforced learning algorithm would help us assign power and rate for each node. In another improvement over our MDP-based dynamic rate adaptation, we proposed to improve the choice of action-set for the state-action policy. Another future work would be to let each node in the network decide on a preferred action-set, which in our view may lead to even more interleaved transmissions and more opportunities to resolve congestion, generating new routes.

8.0 BIBLIOGRAPHY

[1] Arity prolog. *This prolog is discontinued now. Other prolog e.g., SWI Prolog is available.*

[2] *IEEE P802.15.4/D18, Draft Standard: Low Rate Wireless Personal Area Networks.*

[3] The network simulator ns-2 with cmu extension.

[4] Device database systems. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, page 194, Washington, DC, USA, 2000. IEEE Computer Society.

[5] Zigbee aliiance, 2001.

[6] Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pages 1 –305, 2006.

[7] Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 11: Wireless medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages C1 –1184, 2007.

[8] H. Aberbanal. *Analysis of Observed Chaotic Data*. Springer-Verlag, 1996.

[9] F.N. Ali, P.K. Appani, J.L. Hammond, V.V. Mehta, D.L. Noneaker, and H.B. Russell. Distributed and adaptive tdma algorithms for multiple-hop mobile networks. volume 1, pages 546 – 551 vol.1, oct. 2002.

[10] M.H. Ammar and D.S. Stevens. A distributed tdma rescheduling procedure for mobile packet radio networks. pages 1609 –1613 vol.3, jun. 1991.

[11] Hari Balakrishnan. High capacity network design. http://nms.csail.mit.edu/6.829-f05.

[12] Jonathan Beaver, Mohamed A. Sharaf, Alexandros Labrinidis, Ros Labrinidis, and Panos K. Chrysanthis. Location-aware routing for data aggregation in sensor networks. In *In Proc. of Geo Sensor Networks Workshop*, 2003.

[13] Ugur Cetintemel, Andrew Flinders, and Ye Sun. Power-efficient data dissemination in wireless sensor networks. In *MobiDe '03: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 1–8, New York, NY, USA, 2003. ACM.

[14] S. Chatterjea, L.F.W. van Hoesel, and P.J.M. Havinga. Ai-lmac: an adaptive, information-centric and lightweight mac protocol for wireless sensor networks. pages 381 – 388, dec. 2004.

[15] Krishna Kant Chintalapudi. A wireless sensor network for structural health monitoring: Performance and experience. In *in Proceedings of the Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, 2005.

[16] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. Approximate aggregation techniques for sensor databases. In *In ICDE*, pages 449–460, 2004.

[17] B. P. Crow, I. Widjaja, J. G. Kim, and P. Sakai. Investigation of the ieee 802.11 medium access control (mac) sublayer functions. In *Proceedings IEEE INFOCOM '97*, pages 126–133, 1997.

[18] Erick Delage and Shie Mannor. Percentile optimization in uncertain markov decision process with application to efficient exploration. In *ICML '07: Proceedings of the 24th Internation Conference on Machine Learning*, Corvallis, OR, USA, 2007.

[19] Gianluca Dini and Ida Maria Savino. An efficient key revocation protocol for wireless

sensor networks. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 450–452, Washington, DC, USA, 2006. IEEE Computer Society.

[20] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM.

[21] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. pages 263–270, 1999.

[22] Charles R. Farrar, Scott W. Doebling, and David A. Nix. Vibration-based structural damage identification. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359(1778):pp. 131–149, 2001.

[23] Samir Goel and Tomasz Imielinski. Prediction-based monitoring in sensor networks: Taking lessons from mpeg. *ACM Computer Communication Review*, 31:2001, 2001.

[24] S.S. Gorshe and T. Wilson. Transparent generic framing procedure (gfp): a protocol for efficient transport of block-coded data through sonet/sdh networks. *Communications Magazine, IEEE*, 40(5):88 –95, may. 2002.

[25] John Grant and Vladimir Zadorozhny. Logical approach to capability-based rewriting in a mediator for websources. *J. Intell. Inf. Syst.*, 17(1):47–70, 2001.

[26] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transaction on Information Theory*, 46(2):388–404, 2000.

[27] B. Mahr H. Ehrig. *Fundamentals of Algebraic Specification. Equations and Initial Semantics*. Springer-Verlag, 1990.

[28] A.M. Hegland, E. Winjum, S.F. Mjolsnes, C. Rong, O. Kure, and P. Spilling. A survey of key management in ad hoc networks. *Communications Surveys Tutorials, IEEE*, 8(3):48 –66, 2006.

[29] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming, 2001.

[30] Wendi Rabiner Heinzelman, Anantha Ch, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. pages 3005–3014, 2000.

[31] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. pages 174–185, 1999.

[32] L. F. W. Van Hoesel and P. J. M. Havinga. A lightweight medium access protocol for wireless sensor networks. Technical report, 2004.

[33] Vaidya N. Holland, G. and P. Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. In *7th Annual international Conference on Mobile Computing and Networking*, pages 236–251, 2001.

[34] Y. E. Ioannidis and Younkyung Kang. Randomized algorithms for optimizing large join queries. *SIGMOD Rec.*, 19(2):312–321, 1990.

[35] Philippe Bonnet Johannes, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. pages 3–14, 2001.

[36] E. A. Johnson, H. F. Lam, L. S. Katafygiotis, and J. L. Beck. Phase i iasc-asce structural health monitoring benchmark problem using simulated data. *Journal of Engineering Mechanics*, 130(1):3–15, 2004.

[37] Audun Josang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99). The Internet Society*, 1999.

[38] Audun Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3):279–311, 2001.

[39] A. Kamerman and L. Monteban. Wavelan ii: A high-performance wireless lan for the unlicensed band. In *Bell Labs Technical Journal*, 1997.

[40] Mehta A Pister K.S.J Lanzisera, S. Reducing average power in wireless sensor networks through data rate adaptation. In *IEEE International Conference on Communications*, 2009.

[41] CK Lin. A research review of reduced power mode for power-saving in wireless ad hoc networks. Technical report, School of Information Sciences, University of Pittsburgh, 2005.

[42] Ming Liu and M.T. Liu. A power-saving scheduling for ieee 802.11 mobile ad hoc network. pages 238 – 245, oct. 2003.

[43] Srivastava R. Koksal C. E. Liu, S. and P. Sinha. Pushback: A hidden markov model based scheme for energy efficient data transmission in sensor networks. In *Ad Hoc Netw.*, pages 973–986, 2009.

[44] F. Valois M. Dohler, T. Watteyne and J. Lu. Kumars, zipfs and other laws: How to structure a large-scale wireless network? In *Annals of Telecommunications*, pages 146–157, 2008.

[45] P. Dong M. Eyugoglu, C. Forney and G. Long. Advanced modulation techniques for v.fast. In *Eur. Trans. Telecommun.*, pages 243–256, 1993.

[46] M. H. Manshaei M. Lacage and T. Turletti. Ieee 802.11 rate adaptation: A practical approach. In *ACM MSWiM*, 2004.

[47] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *In ACM SIGMOD*, pages 491–502. ACM Press, 2002.

[48] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *IN OSDI*, 2002.

[49] JH Miller and S Page. *Complex Adaptive Systems*. Princeton University Press, 2007.

[50] Thomas Moscriboda. The worst-case capacity of wireless sensor networks. In *IPSN 07: Proceedings of Information Processing in Sensor Networks*, Cambridge, MA, USA, 2007.

[51] Surendra Nahar, Sartaj Sahni, and Eugene Shragowitz. Simulated annealing and combinatorial optimization. In *DAC '86: Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 293–299, Piscataway, NJ, USA, 1986. IEEE Press.

[52] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary Anderson. Synopsis diffusion for robust aggregation in sensor networks. *ACM Trans. Sen. Netw.*, 4:7:1–7:40, April 2008.

[53] Andrew Ng. Reinforcement learning lecture. http://www.youtube.com Search: Andrew Ng Machine Learning.

[54] J. M. Nichols, M. D. Todd, M. Seaver, and L. N. Virgin. Use of chaotic excitation and attractor property analysis in structural health monitoring. *Phys. Rev. E*, 67(1):016209, Jan 2003.

[55] Charles E. Perkings and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[56] The CMU Monarch Project. The cmu monarch project's wireless and mobility extensions to ns, 1998.

[57] M.L. Puternam. *Markov Decision Process*. Wiley, 1994.

[58] Injong Rhee, Ajit Warrier, Jeongki Min, and Lisong Xu. Drand: distributed randomized tdma scheduling for wireless ad-hoc networks. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*.

[59] Injong Rhee, Ajit C. Warrier, and Lisong Xu. Randomized dining philosophers to tdma scheduling in wireless sensor networks. Technical report, North Carolina State University, 2004.

[60] Kanodia V. Sabharwal A. Sadeghi, B. and E. Knightly. Oar: an opportunistic auto-rate media access protocol for ad hoc networks. In *Wirel. Netw.*, pages 39–53, 2005.

[61] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Topology management for sensor networks: Exploiting latency and density. pages 135–145, 2002.

[62] Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, Ros Labrinidis, and Panos K. Chrysanthis. Tina: A scheme for temporal coherency-aware in-network aggregation. In *In MobiDE*, 2003.

[63] Divyasheel Sharma, Vladimir I. Zadorozhny, and Panos K. Chrysanthis. Timely data delivery in sensor networks using whirlpool. In *DMSN '05: Proceedings of the 2nd international workshop on Data management for sensor networks*, pages 53–60, New York, NY, USA, 2005. ACM.

[64] Krishna M. Sivalingam, Jyh cheng Chen, Prathima Agrawal, and Mani B. Srivastava. Design and analysis of low-power access protocols for wireless and mobile atm networks, 1998.

[65] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, 43(12):112 – 119, dec. 2005.

[66] Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke, and Rajmohan Rajaraman. Wave scheduling and routing in sensor networks. *ACM Trans. Sen. Netw.*, 3(1):2, 2007.

[67] R. van Nee, G. Awater, M. Morikura, H. Takanashi, M. Webster, and K.W. Halford. New high-rate wireless lan standards. *Communications Magazine, IEEE*, 37(12):82 –88, dec. 1999.

[68] G.P. Williams. *Chaos Theory Tamed*. Joseph Henry Press, 1997.

[69] Yang H. Lu S. Wong, S. H. and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *12th Annual international Conference on Mobile Computing and Networking*, pages 146–157, 2006.

[70] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31:2002, 2002.

[71] Wei Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. volume 3, pages 1567 – 1576 vol.3, 2002.

[72] M. Younis, M. Youssef, and K. Arisha. Energy-aware routing in cluster-based sensor networks. pages 129 – 136, 2002.

[73] Vladimir Zadorozhny, Panos K. Chrysanthis, and Alexandros Labrinidis. Algebraic optimization of data delivery patterns in mobile sensor networks. In *DEXA '04: Proceedings of the Database and Expert Systems Applications, 15th International Workshop*, pages 668–672, Washington, DC, USA, 2004. IEEE Computer Society.

[74] Vladimir Zadorozhny, Louiqa Raschid, Maria Esther Vidal, Tolga Urhan, and Laura Bright. Efficient evaluation of queries in a mediator for websources. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 2002. ACM.

[75] Vladimir Zadorozhny, Divyasheel Sharma, Prashant Krishnmurthy, and Alexandros Labrinidis. Tuning query performance in mobile sensor databases. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pages 247–251, New York, NY, USA, 2005. ACM.

[76] Vladimir I. Zadorozhny, Panos K. Chrysanthis, and Prashant Krishnamurthy. A framework for extending the synergy between mac layer and query optimization in sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 68–77, New York, NY, USA, 2004. ACM.

[77] J. Zheng and M. J. Lee. *A comprehensive performance study of IEEE 802.15.4*. IEEE Press Book, 2004.

[78] Jianliang Zheng and M.J. Lee. Will ieee 802.15.4 make ubiquitous networking a reality?: a discussion on a potential low power, low bit rate standard. *Communications Magazine, IEEE*, 42(6):140 – 146, jun. 2004.