Faculty of Information Technology

Cost-Efficient Collection and Delivery of Sensor Data using Mobile Devices

by

Prem Prakash Jayaraman

(Student No. [redacted])

Thesis

Submitted in fulfilment of the requirements for the degree of

Doctorate of Philosophy

Supervisors

Prof. Arkady Zaslavsky

Prof. Jerker Delsing

Prof. David Abramson

Date of Submission: 26th October 2010

Table of Contents

List of Figures

List of Tables

Thesis Outcomes

The research work has resulted in 2 Journal paper (1 under review), 7 peer-referred international conference papers and 1 peer-refereed international workshop paper. The paper titled *"Intelligent Processing of K-Nearest Neighbours Queries using Mobile Data Collectors in a Location Aware 3D Wireless Sensor Network"* was awarded the Best Paper Award at *The Twenty Third International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2010)*. The research has also contributed to 3 seminar presentations.

Journal Publication

1. Jayaraman, P.P., Zaslavsky, A., Delsing, J. (2010). "Intelligent Mobile Data Mules for Cost-Efficient Sensor Data Collection". In International Journal of Next-Generation Computing, Volume 1, Issue 1, July 2010. pp: 73-90
2. Jayaraman, P.P., Zaslavsky, A., Delsing, J. (2010)." Intelligent Processing of KNN queries in 3D Wireless Sensor Networks" In The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies (Submitted, Under Review)

Conference and Workshop Publications

1. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2007). "Sensor data collection using heterogeneous mobile devices". In F. Ozguner & B. Baykal (Eds.), Proceedings of the IEEE International Conference on Pervasive Services (ICPS 2007), Istanbul Turkey 15-20 July, pp. 161-164
2. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2008). "Cost efficient data collection of sensory originated data using context-aware mobile devices". In H.-V. Leong, W.-C. Lee, M. Hauswirth, B. Konig-Ries, W. Mansoor, et al. (Eds.), Proceedings of Ninth International Conference on Mobile Data Management Workshops (MDMW 2008), Beijing, China, 27 April, pp. 190-197
3. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2008). "Smart sensing and sensor data collection on the move for modelling intelligent environments". In Proceedings of

the 8th international conference, NEW2AN and 1st Russian Conference on Smart Spaces, ruSMART on Next Generation Teletraffic and Wired/Wireless Advanced Networking, St. Petersburg, Russia, Lecture Notes in Computer Science, vol. 5174/2008 pp. 306-317.

4. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2008). “Coverage area computation on the run for efficient sensor data collection”. In A. Aggarwal, M. Badra & F. Massacci (Eds.), Proceedings of the 2nd International Conference on New Technologies, Mobility and Security (NTMS 2008). Tangier, Morocco, 5-7 November, pp. 1-4

5. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2009). “Dynamic situation modelling and reasoning under uncertainty”. In Proceedings of the 2009 international Conference on Pervasive Services (ICPS 2009), London, United Kingdom, July 13 - 17, 2009, pp 113-122.

6. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2009), “On-the-Fly Situation Composition within Smart Spaces”. Lecture Notes in Computer Science. In Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces, St. Petersburg, Russia, 2009, pp. 52-65.

7. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2010), “Intelligent Processing of K-Nearest Neighbours Queries using Mobile Data Collectors in a Location Aware 3D Wireless Sensor Network”. In The Twenty Third International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2010), Cordoba, Spain, N. García-Pedrajas et al. (Eds.): IEA/AIE 2010, Part III, LNAI 6098, pp. 260–270, 2010 © Springer-Verlag Berlin Heidelberg **(BEST PAPER AWARD)**.

8. Jayaraman, P.P, Zaslavsky, A, Delsing, J. (2010), “Cost Efficient Data Collection Approach Using K-Nearest Neighbours in a 3D Sensor Network”, In Proceedings of the 2010 Eleventh International Conference on Mobile Data Management (MDM 2010), Kanas City, Missouri, USA, May 23 - 26, pp 183-188.

Seminar Presentations

1. Jayaraman, P.P, “Coverage area computation on the run for efficient sensor data collection”, at 2nd International Conference on New Technologies, Mobility and Security (NTMS 2008), Tangier, Morocco, November 2008
2. Jayaraman, P.P, “Data Collection in Sensor Networks: Mobility-based Techniques”, at Lulea University of Technology, Lulea, Sweden. October 2008
3. Jayaraman, P.P, “Sensor Data Collection using Intelligent Mobile Data Collectors”, at Caulfield School of Information Technology, Monash University, Melbourne, Australia. April 2007

Abstract

Wireless sensor networks represent an important component of distributed pervasive computing infrastructure supporting a range of applications including health, military, environmental monitoring, civil structure monitoring, smart homes, etc. The primary factor driving such pervasive real-world applications is availability of data from sensors distributed in the environment. Traditional way of collecting data is to transmit the data from sensors to a collection point using wireless radio communications. However, the traditional approach is expensive and not always efficient.

This thesis addresses a major challenge of cost-efficient collection of data from wireless sensor networks. Our data collection philosophy is to use mobile devices as sensor data collectors. The use of mobile devices as *mobile data mules* facilitates the formation of a mobile access network that can be used by sensors to connect to the external world.

We propose, investigate, develop and validate a sensor data collection framework called sGaRuDa which enables interoperable capabilities and takes advantage of existing communication and hardware capabilities of the mobile data mule platforms enabling them to collect sensor data on-the-run. The sGaRuDa framework incorporates intelligent mobile data mule allowing them to dynamically make data collection and delivery decisions. The sGaRuDa framework and the corresponding data collection algorithms are targeted at sensor networks that use short range radio communication technologies like Bluetooth. We have also proposed, implemented and validated a novel three dimensional *k*-Nearest Neighbour query-based sensor data collection approach called 3D-KNN to address broadcast-based sensor network communication architectures. The 3D-KNN facilitates multi-hop data collection from infrastructure-less wireless sensor networks (e.g. Zigbee).

We propose, develop, implement and validate a dynamic smart spaces modelling approach called Ranked-Context Spaces (R-CS). Our smart spaces modelling approach is driven by the notion of situation modelling and reasoning about context. Ranked-Context Spaces is capable of computing situation-based smart spaces model taking into account changing contextual information. R-CS is proposed as an extension to Context Spaces theory.

The thesis presents implementation and evaluation details of the proposed sGaRuDa framework and the 3D-KNN algorithm. We have demonstrated the feasibility and cost-efficiency of the sGaRuDa system framework in real-world environments by implementing a proof-of-concept prototype on a range of mobile device platforms, namely, Personal Digital Assistants and mobile robot. Extensive evaluation and experimentation have been performed to prove the extent of energy conservation using the proposed data collection framework and the 3D-KNN algorithm. Finally, we have implemented the R-CS system to demonstrate its reasoning ability under uncertainty. Experiments based on synthetic sensor data streams have been performed to evaluate the proposed Context Spaces extensions incorporated into R-CS.

During the course of the thesis work, 7 peer-refereed international conference papers, 1 peer-refereed workshop paper and 1 journal paper have been produced. One of the conference papers was awarded a *BEST PAPER AWARD*.

Declaration

I declare that this thesis contains no material that has been accepted for the award of any degree or diploma in any university or other institution and affirm that to the best of my knowledge the thesis contains no material previously published or written by any other person, except where due reference is made in the text of the thesis.

Prem Prakash Jayaraman

Acknowledgement

PhD is a great journey that has taught me a lot of things. This journey would never have been possible without Professor Arkady Zaslavsky, my main supervisor, who encouraged me to take up the PhD degree. His constant effort, motivation and advice have helped me face some of the tough challenges during the PhD years. I can never forget the days when he walked past my office making sure I am focused and working towards completion. His comments and guidance have been invaluable towards shaping the research outcome and writing of this thesis. Without his support, this thesis would have never been completed. I am very grateful to you Sir. You have and will always be my mentor and well wisher. I am very honoured and fortunate to have been associated with you over the years.

I thank Professor Jerker Delsing, my second supervisor, for his advice, supervision and crucial contributions. His insights, knowledge and comments have greatly helped during the writing of the thesis. He has always made time for discussions in-spite of his busy schedule and different time zones. The knowledge I imparted during our discussions will benefit me for a very long time to come. The opportunity to spend some of the PhD time in Lulea University would not have been possible without your support. I would also like to extend my sincere thanks to Dr. Jens Elliason from Lulea University of Technology for his support with Mulle programming.

Special thanks to Professor David Abramson for his support during later stages of the research project.

A very special thanks to Rob Gray for the outstanding job he did on proof reading the thesis. His comments, suggestions and corrections have helped significantly in improving the quality of the thesis. His support and encouragement during challenging times were invaluable.

I thank Ms. Julie Simon for providing me with timely teaching opportunities at Monash College. I thank all the administration and technical staff members at Caulfield School of Information Technology. Their support and interaction during different stages of the PhD was invaluable. Special thanks to Duke Fonias, Samedin Balla, Rafig Tjahjadi, See Ngieng, Mirek Szczap, Akamon Kunkongkapun, Allison Mitchell, Katherine Knight, Michelle Ketchen, Denyse Cove, Cornelia Lioulios, Diana Sussman and many others that I might have unintentionally left out.

On a personal front my heartfelt and deepest gratitude to my parents for instilling in me the confidence to purse the PhD degree. They have constantly motivated me and have stood by me during difficult times. Thanks for having believed in me and supported me all these years. *I would like to dedicate this thesis to them*, for having stood by me and given me everything I wanted at various stages of my life. But for your love and support my dear parents, I would not have been where I am today. Words cannot express my gratitude towards you. I would also thank my twin sister Saroj and our new member of the family, my brother-in-law Magesh. Her care and affection has helped me stay motivated. If not for her, I would have been foodless for days. Thanks Saroj, for making lunch and dinner for me when I used to work long hours in university. Besides them, I would also like to thank all my friends and cousins specially Gayathri who have been of great support to me over the years. A special thanks to Sailakshmi for her support and encouragement.

I thank all the anonymous reviewers of the published papers for providing valuable comments which greatly helped in fine tuning the research.

Finally, I would like to thank everyone who was supportive for the successful realisation of this thesis and apology for having missed to individually mention them.

1 Introduction

1.1 Preamble

Mark Weiser, widely considered the father of ubiquitous computing (Wikipedia, 2010), expressed a vision for 21st century computing devices (Weiser, 1999). Quoting Mark Weiser's vision from (Weiser, 1999) "*the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*" His vision reflected computing as an integral part of *everyday* human activity/life. Wireless Sensor Networks are one such technological advancement that has revolutionised the way of embedding computing devices in physical spaces. MIT's technological review (Huang, 2003) has identified Wireless sensor networks (WSN) as *"one of the top ten technologies that will change the world"*.

WSN have attracted a considerable amount of research interest in recent years (Shen et al., 2001, Culler et al., 2004) facilitated by advances in manufacturing of high density electronics. Wireless sensor node is a key component of WSN. The wireless sensor node is a tiny, battery powered computing device with sensing, processing, storage and communication capability. These aforementioned capabilities of WSN's have made them suitable for a number of application domains including domestic (Smart homes, home health care systems (Baker et al., 2007, Srivastava et al., 2001)), military (Akyildiz et al., 2002), environmental monitoring (Ramanathan et al., 2005) and scientific/industrial (Structural Monitoring (Xu et al., 2004)) applications. For example, Grape Networks(2010), a Wireless Sensor Networking company specialising in deploying and managing wireless sensor nodes at a commercial scale, have implemented WSN technology in vineyards to monitor environmental conditions.

The tiny low-powered (battery) sensor nodes deployed within an area have the capability to sense environmental parameters and collaborate with other sensors in the neighbourhood. Wireless sensor nodes distributed within an area work together to achieve single or multiple goals (Shen et al., 2001). The sensing property of sensor nodes allows them to gather information from the physical world (e.g. temperature). The sensor has on-board capability to process, store or transmit sensed information wirelessly, based on application requirements. The advent of WSN has enabled the acquisition of data from physical environments which previously was expensive, difficult or even impossible (Chu et al., 2006). WSN has paved the way in realising Mark Weiser's vision for 21st century computing.

The increased adaptation of WSN across various application domains has resulted in generation of massive amounts of data (Shen et al., 2001, Culler et al., 2004). Hence collecting data from sensors has been identified as one of the several challenges that need to be addressed for large-scale sensor network adoption. The process involving collection of the sensed data from the sensor nodes is called "*sensor data collection*". The resource constrained nature of sensor nodes make the data collection process more challenging, hence embodying the need for energy-efficient data collection techniques.

A key application area that uses sensor data as its foundation is smart computing applications i.e. context-aware systems. Context-aware systems have the capability to adapt their behaviour (operation) based on real-world contextual information in order to deliver the best service to the user (Hähner et al., 2004). Context-aware applications (Loke, 2006) rely on real-world information which is sourced from sensors (physical and software). The sensor data from wireless sensors are dynamic i.e. they change over time. Hence, the context-aware applications need to have the ability to adapt to changing contextual information.

In this research, we focus on the challenge of energy-efficient sensor data collection. In addressing the sensor data collection challenge, we note that communication is a major factor that affects sensor lifetime. The lifetime of a sensor node is the amount of time the sensor can perform its operations before its battery drains. Further, we extend our research by addressing the challenge of dynamically modelling smart spaces[1] using data collected from sensors that are embedded within the environment. In particular, we introduce the notion of situation-based smart spaces modelling where real-world contextual information is used to build a virtual model of the physical situation. The rest of the chapter is organised as follows: Section 1.2 presents the research motivation. Section 1.3 presents the research aims and contributions. Finally, section 1.4 presents a roadmap of the thesis structure.

1.2 Research Motivation

Currently there are around 4 billion mobile phones[2] (ITU, 2008) in use in the world. Many of these devices have enough spare energy to perform operations other than regular telephony and internet-based services. Moreover, current day mobile platforms come equipped with a plethora of communication technologies including GSM, Wi-Fi, Bluetooth, UMTS, etc, capable enough to form a ubiquitous mobile wireless access network. A survey conducted over three days at an info-security conference 2006 in London (Gostev, 2006) resulted in discovery of more than 2000 Bluetooth enabled devices in visible mode. The profuse use of mobile devices equipped with a host of communication technologies created the foundation of our vision to use *day-to-day* mobile computing devices as mobile sensor data collectors. We envision the use of mobile computing devices as mobile access points for sensor networks. The ubiquitous presence of mobile devices and increased number of sensor network deployments in smart environments, facilitates the creation of mobile wireless access networks with sufficient bandwidth (available from mobile device communication channels, e.g. GSM, UMTS) to support data originating from sensors.

The use of mobile devices as collectors and carriers of sensor data allows application developers to access sensor data from any location instantly which otherwise would require specially designed data-sink infrastructure. We take advantage of short-range communication technologies available on current day mobile devices. Our proposal

[1] *Smart space* is the term used to represent spaces embedded with computing infrastructure Satyanarayanan M., 2002. Pervasive computing: vision and challenges. *Personal Communications,* vol. 8, no. 4**,** pages: 10-17.

[2] We use the mobile phone as a classic example of widely available mobile device platforms in *everyday life*

explores the use of mobile devices as a shared access network that can act as data carriers for sensor nodes. This approach introduces a new sensor network paradigm facilitating a platform for easy adoption of sensor networks in applications which otherwise required laborious and expensive network infrastructure. An application scenario illustrating our motivation is presented in Figure 1.1. The illustration in Figure 1.1 depicts an industrial environment where sensors are distributed at different locations within a building. The mobile devices that act as data carriers in the scenario are mobile phones used by factory workers who move around within the sensor network environment.

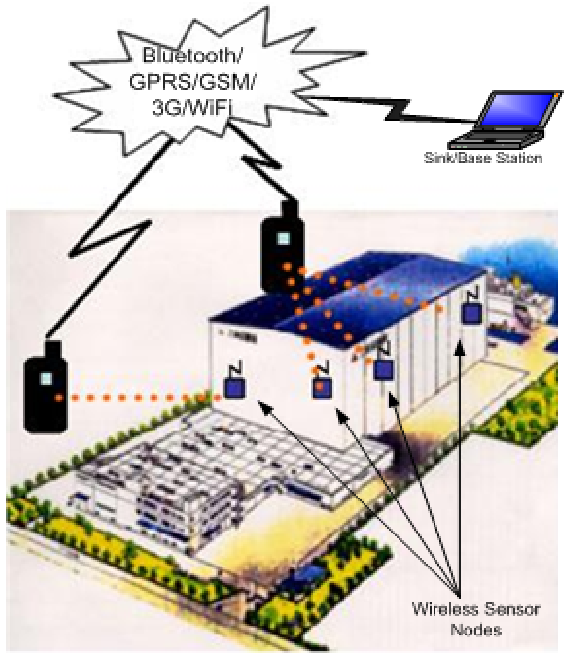


Figure 1.1: Motivation Scenario Example

Our motivation is further supported by the widespread acceptance of short-range communication technologies like Bluetooth. Recent research outcomes validate the feasibility of Bluetooth for low-powered sensor network operations (Leopold et al., 2003). The popularity and adoption of Bluetooth has already made it a ubiquitous technology helping us realise our vision of creating a heterogeneous network comprising Bluetooth enabled *day-to-day* mobile devices interacting with sensor nodes deployed within smart spaces. Moreover, our vision supports a variety of technologies, Zigbee being one such example. The adoption of 802.15.4 Zigbee (2009a) protocol for wireless sensor networks and development of Zigbee enabled mobile devices (ZigBee, 2007) has further strengthened our vision, extending our approach to future mobile devices. The use of *day-to-day* mobile devices as a vehicle to collect sensor data exploits a mobile device's spare capacity, utilising the mobile access network without the need for dedicated sensor data collection infrastructure.

1.3 Research Objectives and Contributions

The introduction of *day-to-day* mobile devices as sensor data collectors opens up a number of application possibilities. We use the term *day-to-day* to embody the class of mobile devices available in current *day-to-day* human activities. These mobile devices include smart mobile phones, personal device assistants (PDA), tablets, laptops, mobile robots, etc. We aim to take advantage of existing networking capabilities available on mobile devices to form a ubiquitous mobile device-based dynamic sensor data collection infrastructure. We term these mobile devices *mobile data mules*.

Our proposed approach creates the platform for tiny low-powered sensors to connect to an access network using mobile devices available within the environment. Recent research in the area of Bluetooth-based sensor networks (BSN) have spawned new application opportunities (Deventer et al., 2009, Eliasson et al., 2008, Eliasson et al., 2007) by which, individual sensors connected to the internet provide a range of services to the end-user. The wide acceptance of Bluetooth technology and its pervasiveness across current day mobile device platforms has helped in realising our vision presented in the previous section. Further, advances in the development of Zigbee-based mobile devices (ZigBee, 2007) and the large availability of Zigbee-based sensors (CrossbowTechnology, 2010a), introduce the need for data collection protocols to suit a wide range of sensor network platforms. Hence, in this thesis, we propose data collection algorithms targeted at a wide range of sensor network platforms, namely, sensor networks with broadcasting capability (Zigbee-based) and sensor networks without broadcasting capability (Bluetooth-based). Our approach takes advantage of ubiquitous availability of mobile devices within human environments. Hence, it is important and challenging to develop energy-efficient data collection protocols that can work on low-powered sensor nodes. Moreover, the data collection protocols need to have the capability to work on current day mobile devices without modifying the device platform.

Further, to validate the importance of the collected sensor data in smart environments, we also explore a dynamic smart spaces modelling approach. We use the term *smart spaces* to define pervasive environments embedded with computing device having the capability to react to environmental changes. Sensor networks play an important role in current world smart spaces as they are the source of valuable data. The thesis explores the possibility of a dynamic smart spaces model that adapts to changing sensor data (collected by mobile data mules).

Hence, the aim of this thesis can be summarised as follows: *to study, develop, implement and evaluate cost-efficient sensor data collection approaches that suit a wide range of sensor network platforms using day-to-day mobile devices as mobile data mules and to implement and develop a smart spaces modelling technique that adapts dynamically using sensor data collected by mobile data mules.*

The aim of this thesis is driven by both real-world sensor network applications that require cost-efficient ubiquitous data collection approaches and sensor network research which so far has not addressed sensor data collection using day-to-day mobile devices highlighted in section 2.3.6. This research can be applied to many application domains including health-care and intelligent road networks, to name a few, and has the capability to spawn new applications (e.g. district heating application presented in section 3.6) within the area of pervasive computing. We briefly highlight how the research presented in this thesis can benefit health-care. Bluetooth-based sensors have been identified to revolutionise health care devices (Wodajo, 2010). Wodajo et.al. (Wodajo, 2010) puts forward the use of disposable sensors embedded on health care devices that can provide real-time patient data to doctors. The use of disposable sensor nodes renders a constantly changing sensor network environment that requires new data collection approaches. This research addresses the sensor data collection challenge by taking advantage of existing mobile device infrastructure available in most urban areas (e.g. health care centre). The mobile devices collect the data generated by the health-care devices embedded with Bluetooth sensors. The approaches investigated in this thesis advances the research in sensor data collection by proposing an architecture that can bridge sensor nodes and ubiquitously available mobile devices. In section 3.6 we present detailed descriptions of two real-world applications that can take advantage of the sensor data collection technique proposed in this thesis. The real-world examples presented in section 3.6 may lead to a new class of sensor network applications which may have not been possible in the past.

To further simplify the overall aim of this thesis we breakdown the objective into three research questions. They are:

1) Is it possible to use *day-to-day* mobile devices as an energy-efficient alternative to collect data from a Bluetooth-based sensor network embedded within the environment?
2) Is it possible to extend the mobile device-based data collection algorithms to suit a wider range of sensor network communication architectures (broadcasting-based)?
3) Is it possible to develop a dynamic smart spaces model using newly generated sensor data?

To address these three research questions, the thesis proposes techniques that achieve cost-efficient sensor data collection from a class of sensor network platforms. Further, we propose a situation-based dynamic context modelling approach based on Context Spaces (Padovitz et al., 2004, Padovitz, 2006, Padovitz et al., 2005) that models smart spaces using sensor originated data. Our notion of smart spaces modelling is governed by situation modelling. The thesis makes the following contributions to address the three research questions.

1) A sensor data collection framework, namely, sGaRuDa (Jayaraman et al., 2008a, Jayaraman et al., 2008c, Jayaraman et al., 2008b, Jayaraman et al., 2007) that can be implemented on *day-to-day* mobile devices enabling them to discover, negotiate, collect and deliver sensor data. The proposed system framework incorporates
    a. Algorithms for energy-efficient discovery of Bluetooth-based sensors
    b. A sliding window inspired data collection algorithm to collect data from sensors in the presence of multiple independent mobile data mules

2) A dynamic sensor duty cycle adaptation (activation schedule) (Jayaraman et al., 2008a, Jayaraman et al., 2008c, Jayaraman et al., 2008b, Jayaraman et al., 2007) algorithm used to modify the sensor's duty cycle on-the-fly, using the mobile data mule with the objective to increase the sensor discovery ratio, hence reducing sensor energy consumption. The proposed sGaRuDa system framework facilitates on-the-fly exchange of the activation schedule.

3) A novel *k* nearest neighbour query based sensor data collection algorithm, namely, 3D-KNN (Jayaraman et al., 2010a, Jayaraman et al., 2010c, Jayaraman et al., 2010b) targeting sensor nodes with broadcasting capabilities (e.g. Zigbee). The proposed 3D-KNN algorithm has the ability to work in a three dimensionally distributed sensor network. The 3D-KNN algorithm takes into consideration sensor location and communication channel quality in a 3D space to identify the best set of energy-efficient sensors neighbouring the mobile data mule. The 3D-KNN algorithm uses multi-hop sensor data collection. The proposed 3D-KNN algorithm is a pioneering work in using mobile data mules to employ *kNN* queries for sensor data collection in a 3D sensor network.

4) A dynamic situation modelling approach, namely, R-CS (Jayaraman et al., 2009b, Jayaraman et al., 2009a, Jayaraman et al., 2008c), proposed as an extension to Context Spaces (Padovitz, 2006, Padovitz et al., 2004, Padovitz et al., 2005) that allows dynamic composition of situations based on collected sensor data.

Further, to verify, validate and evaluate the proposed data collection algorithm and the smart spaces modelling approach, the thesis makes the following contributions:

1) Prototype implementation of the sGaRuDa framework and its data collection algorithms on real-world mobile devices validating the function feasibility of the system.

2) Demonstrates the cost-efficiency of sGaRuDa by extensive real-world experimentations.

3) Demonstrates the cost-efficiency of the proposed 3D-KNN algorithm by extensive evaluations and experimentation over large-scale wireless sensor network environments.

4) Implements, validates and evaluates the dynamic situation modelling extensions incorporated to Context Spaces (Padovitz, 2006).

Our research methodology is driven by proposing, developing, implementing and testing the proposed algorithms and protocols to address the three research questions. Our proposed approaches are supported by extensive evaluation outcomes in real-world and simulator environments.

1.4 Thesis Structure

The thesis is organised into 7 chapters excluding introduction. An illustration of the thesis structure is presented in Figure 1.2.

Chapter 2 presents a literature background on sensor data collection techniques. We present a broad background on current state-of-the art wireless sensor networks. Further, we present an in-depth literature review of mobile and non-mobile based data collection approaches, identifying their shortfalls and building an argument for our proposed approaches. Finally, we present a literature review of situation-based context modelling approaches identifying the need for a dynamic smart situation modelling approach.

Chapter 3 presents in-depth discussion of the proposed sensor discovery, data collection and sensor management algorithms. This chapter is partly based on the published papers (Jayaraman et al., 2008a, Jayaraman et al., 2008c, Jayaraman et al., 2007, Jayaraman et al., 2010b). We propose our system framework, namely, sGaRuDa. The proposed system framework can be implemented on current generation mobile devices. The system framework primarily targets Bluetooth-based sensors nodes while the algorithms proposed for data collection can be extended with relative ease to non-Bluetooth based networks.

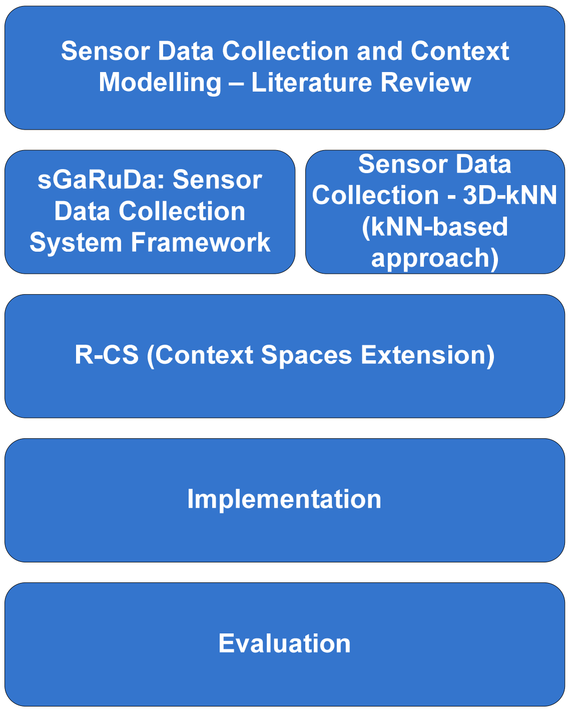

Sensor Data Collection and Context Modelling – Literature Review
sGaRuDa: Sensor Data Collection System Framework
Sensor Data Collection - 3D-kNN (kNN-based approach)
R-CS (Context Spaces Extension)
Implementation
Evaluation

Figure 1.2: Thesis Structure

Chapter 4 extends the proposed system framework by introducing *k* nearest neighbour-based three dimensional sensor data collection (*3D-kNN*) algorithm using mobile data mules. This chapter is based on published papers (Jayaraman et al., 2010a, Jayaraman et al., 2010c, Jayaraman et al., 2010b, Jayaraman et al., 2008b). The proposed *3D-kNN* algorithm aims at cost-efficient discovery and data collection from broadcast-based sensor nodes. We present a detailed discussion of the sensor discovery and multi-hop data collection algorithms.

Chapter 5 presents the proposed Context Spaces extensions that are implemented in our R-CS reasoning system. This chapter is based on the published papers (Jayaraman et al., 2009b, Jayaraman et al., 2009a, Jayaraman et al., 2008c). The proposed extension allows R-CS to compose situations on-the-fly enabling increased reasoning accuracy. We also discuss introduction of additional metrics into R-CS to increase the reasoning confidence under uncertainty.

Chapter 6 presents implementations of the proposed system framework (sGaRuDa), the *3D-kNN* algorithm and R-CS (Context Spaces extensions). We present a detailed discussion of sGaRuDa (the proposed data collection framework) implementation on real-world devices (personal digital assistant) and sensor node (Mulle). The *3D-kNN* algorithm has been implemented within a simulator environment. We present a detailed discussion of the simulator environment and the *3D-kNN* implementation. Finally, we present the details of JAVA implementation of R-CS.

Chapter 7 presents extensive evaluations and experimental results of the systems presented in Chapter 3, Chapter 4 and Chapter 5 respectively. We perform real-world experimentations to prove the feasibility of sGaRuDa supported by real-world evaluation outcomes to validate the efficiency of the proposed algorithms. Similarly, we discuss the results of extensive evaluation of the *3D-kNN* algorithm in our simulator environment simulated over large-scale sensor networks. The outcome of the *3D-kNN* algorithm is compared with a non-mobile *kNN*-based sensor data collection algorithm to validate its cost-efficiency. Finally, we present evaluation results of R-CS comparing the experimental outcomes with Context Spaces validating the proposed extensions.

Chapter 8 concludes the thesis with pointers to possible future works.

2

Sensor Data Collection Approaches and Context Modelling - A Literature Review

2.1 Introduction

A major focus of this dissertation is to address key challenges in sensor data collection. The advent of sensor networks, primarily driven by advances in manufacturing of high density electronics, is hindered by slow developments in battery technologies. The energy (battery power) constrained nature of sensors is a major consideration in the development of data collection protocols. This chapter presents a survey on sensor data collection techniques available in the literature. In section 2.2, we present an overview on wireless sensors network (WSN) operations identifying key challenges. Sections 2.3 and 2.4 present current literature on sensor data collection approaches. Finally, sections 2.5 and 2.5.2 discuss current literature on situation-based context modelling and reasoning approaches identifying the importance of dynamic situation modelling.

2.2 Wireless Sensor Networks

2.2.1 Overview

A *wireless sensor network* (WSN) is a collection of components that work together to achieve single or multiple goals (Vieira et al., 2003, Burke et al., 2006). A major component of WSN is distributed, autonomous, tiny devices called wireless sensor nodes. Wireless sensor nodes are equipped with on-board sensing, communication and data processing capabilities (Akyildiz et al., 2002). The sensor essentially is an *atomic unit* (Kumar, 2003) comprising the capabilities mentioned earlier. Kumar (Kumar, 2003) uses the term "*Atomic Computing Particle*" to describe sensors. Sensor networks help forming a *fully connected information space* (Kumar, 2005, Kumar, 2003). The research in the area of sensor networks was pioneered by UC Berkeley researchers (Culler et al., 2004) who were instrumental in developing the *MICA mote* sensor platform (CrossbowTechnology, 2010a) . The sensor node has the capability to work with a variety of sensors ranging from mechanical, thermal, biological, chemical, optical and magnetic sensors (Yick et al., 2008). In most cases, the primary power source of a sensor node is a battery though recent research initiatives have explored secondary power sources like solar panels (Alippi et al., 2008, Kansal et al., 2007). Going by Moore's Law (Pinto, 2002) - *number of transistors on a chip doubles every 18 months* and Gilder's Law (Pinto, 2002) – *communication bandwidth triples every year* – a theory for battery life prediction is unavailable. This limitation in power always remains the primary challenge in sensor design and operation.

Early research in *wireless integrated network sensor* (sensor networks) (Pottie et al., 2000, Asada et al., 1998, Bult et al., 1996) was motivated by pervasive computing application requirements to bridge the physical and virtual worlds. Sensors are an excellent technological advancement that made this requirement a reality. They facilitated monitoring and collecting physical world parameters that were fed into virtual world pervasive applications. The Smart kindergarten (Srivastava et al., 2001) project is one such example of a pervasive application dependent on sensor inputs. A simple wireless sensor network is illustrated in Figure 2.1. Though sensor networks share a number of ad-hoc network characteristics (Frodigh et al., 2000), algorithms employed in traditional ad-hoc networks cannot be straight-forwardly adapted to sensor network requirements.

A typical WSN has little or no network infrastructure information (Yick et al., 2008). In most applications WSN's are deployed within a dense area in an ad-hoc manner. Based on the type of network deployment, they can be classified into structured and un-structured. A deployment depicted in Figure 2.1 is classified as un-structured while planned deployment in specific locations is classified structured. The sink/gateway/base station is another important component of WSN. It is a centralized component to which sensed data is transmitted for further processing/decision making. In this thesis, we use the terms *sink* and *base station* interchangeably.

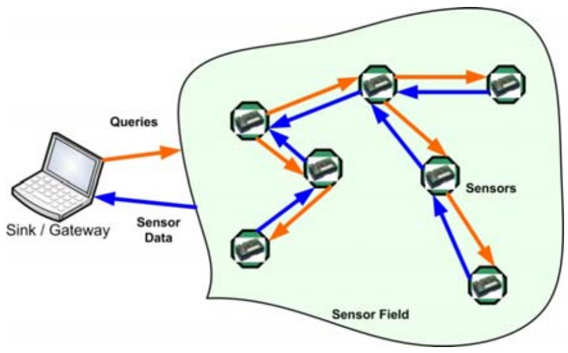


Figure 2.1: A Simple Wireless Sensor Network Deployment

2.2.2 Sensor Node Components/Operations

A sensor node is made of four main components, namely, physical sensor, battery, communication radio and processor each associated with a key operation:

Sensing: The process of measuring the physical phenomenon is sensing. A wireless sensor can be equipped with different types of sensors based on applications requirement. The sensing unit in a sensor may have more than one type of sensor installed. The sensing unit is responsible in managing various sensors converting electrical signals generated by changes in physical environment into raw sensed data. This raw data can be saved to an on-board memory or transmitted to a central processing station wirelessly. For example, a temperature sensor has the ability to measure temperature within the deployed environment. Sampling performed by the sensor i.e. sensing the change in the parameter measured can be classified into *Periodic* and *On-Request*. A *periodic* sampling of data is performed over a time cycle (e.g. sample temperature every 2 minutes). This involves waking-up the sensing device, measuring the parameter and storing the raw data. *On-Request* sampling of data is controlled by the application. This is on-demand sampling where the sensor performs sampling after receiving a request from the application.

Power Management: Power Management manages available power sources by performing energy-efficient sensor operations. Sensors are energy-constrained devices that depend on efficient battery usage for extended lifetimes. Power management schemes employed can use battery voltage information to extend the life of the sensor (Park et al., 2002). The most common energy source for sensor nodes is battery while concurrent research into harvesting energy from the environment is being explored. Amirtharajah et al. (2004) and Roundy et al. (2004) explore harvesting energy from environmental vibrations while (Kansal et al., 2007, Kansal et al., 2003, Alippi et al., 2008) investigates solar cells. Table 2.1 provides a summary of power density available from various energy sources. The data presented in Table 2.1 is a summary of experimental results obtained from (Rabaey et al., 2000, Vieira et al., 2003).

Energy Source	Power Density	Energy Density
Battery (rechargeable lithium)		300 mWh/cm3 (3 - 4 V)
Solar (outdoors)	15 mW/cm2 (direct sun) 0.15mW/cm2 (cloudy day)	
Solar (indoors)	0.006 mW/cm2 (standard office desk) 0.57 mW/cm2 (< 60W desk lamp)	
Vibrations	0.01 - 0.1 mW/cm3	

Table 2.1: Comparing Power from Various Energy Sources

Power-efficient sensor operation is by far the most important design constraint and requires every protocol, function of the sensor to abide with this design consideration. A number of Dynamic Power Management (DPM) approaches are presented in (Lin et al., 2006, Passos et al., 2005, Sinha et al., 2001, Wang et al., 2001). The principle behind DPM is to dynamically shutdown sensor components when no events occur. An event is defined as a variation in the sensed phenomenon. A key issue is when to turn-off/on the sensor components. For example, while performing a periodic sensing operation, turning off the processor during non-sensing period would save energy. On the other hand returning the sensor components from sleep consumes energy and time. Hence, there is always a trade-off between energy and time spent in turning-off and turning-on the sensor components (Lin et al., 2006).

Communication: The sensor node is equipped with a radio that allows it to exchange data with neighbouring sensors and the base station (sink). The data exchange between sensors or a sensor and base station takes place in three possible ways: 1) unicast: one-to-one - one sender communicates with one receiver, 2) multi-cast: one-to-group - one sender communicates with a selected group of receivers and 3) broadcast: one-to-all - one sender communicates with all receivers that are within the communication range. We argue, communication is the major energy consuming component in a sensor node compared to the other functional components (Raghunathan et al., 2002). Simulation results presented in (Shnayder et al., 2004) validate our argument. Hence energy-efficient use of the radio is the primary criterion for any communication protocol design. The sensor radio component determines its communication characteristics. Though radio frequency (RF) has been the widely used transmission media (Pottie et al., 2000, CrossbowTechnology, 2010a) optical medium has also been employed by SmartDust (Kahn et al., 1999) sensor platform. The antenna size used in RF communication needs to be a fraction of the wavelength for efficient operation and coverage (Warneke et al., 2002). The miniature design of sensor nodes allows only small antennas, thus reducing the size of the antenna results in low antenna gain. This design outcome reduces the communication range of sensor nodes. For example, a sensor equipped with a Bluetooth (2010c) radio like the Mulle (Eliasson et al., 2008) has a communication range of 10 to 100m. A number of communication standards have been proposed for sensor networks focusing on sensor design requirement for low power consumption. These standards, to name a few, include IEEE 802.15.4 (Howitt et al., 2003), Zigbee (2009b), Wibree (BluetoothSIG, 2010b), IEEE 802.15.3 (IEEE, 2003.). We have presented a summary of radio hardware characteristics in Table 2.2 currently available on most sensor platforms. The source of data is a combination of datasheets (TexasInstruments, 2010, TexasInstruments, 2007) and experiments (Shnayder et al., 2004, Cordeiro et al., 2006, EISLab, 2010, Polastre et al., 2005). Few radio hardware technologies presented in Table 2.2 have the capability to dynamically change the receiver sensitivity to improve communication reliability.

Processing: The processing unit is the brain of the sensor node performing the computation and operations of controlling and coordinating the various system components. The primary function of the processing unit is to process raw data[3] obtained from the sensing unit and transmit it over the communication channel to the network base station (sink) for further processing.

Characteristics / Radio Device	Carrier Frequency	Receiver Sensitivity	Data Rate (kbps)	Range (n)	Current Consumption (RX)/(TX) (mW)
TR1000 (MICA Mote)	916.5 MHz	-97dBm	40	100	12 /36 @0dBm
CC1000 (MICA 2 / BTnode rev3)	300 to 900 MHz	-98dBm	38.4	100	29 / 42 @0dBm
CC2420 (MICA 2/ MICAz / Telos/ Sun SPOT/ TMote Sky)	2.4 GHz IEEE 802.15.4	-95dBm	250	50 /30 (indoor) 125 /100 (outdoor)	38 / 35 @ 0dBm
Mitsumi WML-C46 AHR (Mulle)	2402 - 2480MHz (Bluetooth v1.2)	-80dBm	721	10	47.9

Table 2.2: Radio Hardware Characteristics

The energy consumed by the processor to perform operations like process raw sensor data, prepare the data for transmission, control the sensor components is relatively low compared to the energy spent in communication (Raghunathan et al., 2002). For example, the MICA mote consumes about *720nJ/bit* to transmit data and *110nJ/bit* to receive data while it consumes only *4nJ* to perform a single operation

[3] Raw data refers to the analog data sensed by the sensing unit which is converted to digital data using Analog to Digital convertor (ADC)

(Srivastava, 2002). As processing is less expensive compared to communication, it is appropriate to process, validate and compress raw data before transmitting it to the base station (sink). A number of power management approaches have been employed to dynamically change the processor's speed for energy-efficient operation. Dynamic voltage scaling is one such technique employed to dynamically change the processor's voltage resulting in lower clock frequencies depending on computational loads (Gutnik et al., 1997). A performance evaluation of energy consumed against dynamic clock frequencies is presented in (Chandrakasan et al., 2002). Table 2.3 presents taxonomy of microprocessor characteristic used in current sensor node platforms. The CPU on most sensor platforms has access to onboard memory (RAM-Random Access Memory) used to store sensor data.

Characteristics / Micro-processor	Bits	Clock Frequency	Operating Voltage	Power (Idle Mode) mA	Power (Active Mode) mA	Flash (Bytes)
ATmega128L (MICA2 / MICAz)	8	8 MHz	2.7 - 5.5V	2.5mA @ 3V, 4MHz 11mA @ 5V, 8MHz -	5.5mA@ 3V, 4MHz - 19mA @ 5V, 8MHz	128K
TI MSP430F16X (Telos / Tmote Sky)	16	8MHz	1.8 - 3.6 V	95µA @ 3V, 1MHz 22µA @ 3V, 0MHz	600µA @ 3V, 1MHz 20µA @ 3V, 4,096Hz	48K
Renesas M16C / 62C (Mulle)	16	24MHz (Mulle @ 10MHz)	2.7 to 5.5 V	1.8 µA @ 3V, 32kHz (wait mode), 0.7 µA @ 3V (stop mode)	14mA @ 5V, 24MHz 8mA @ 3V, 10MHz	256 K
Atmel AT91RM9200 (Sun SPOT)	32	180 MHz	1.65V - 3.6 V	13.8mA @ 3.3V, 180MHz	24.4mA @ 3.3V, 180MHz	128K

Table 2.3: Characteristics of microprocessors used in current sensor platforms

The data Table 2.3 has been tabulated from different datasheets (SunMicrosystems, 2007, Atmel, 2009a, Atmel, 2009b) and experimental outcomes (EISLab, 2010, Polastre et al., 2005, Shnayder et al., 2004). The characteristics of microprocessors widely used in sensor platforms presented in Table 2.3 validate Moore's law i.e. Sun SPOT (SunMicrosystems, 2007) uses a 180MHz, 32 bit processor as compared to first-generation MICA motes (8 MHz).

2.2.3 Sensor Platforms

The previous section provided an introduction to sensors identifying the key functional units and their corresponding operations. A number of sensor node hardware has been developed in recent years taking into consideration the key design criteria: 1) energy-efficient hardware, 2) energy-efficient sensor operation and 3) energy-efficient sensor software (operating system/ algorithms). Table 2.4 is an in-depth analysis of current state-of-the-art sensor platforms. The summarised data presented in Table 2.4 is from Crossbow (CrossbowTechnology, 2010a, CrossbowTechnology, 2010b), ETH Zurich (ETH-Zurich, 2007), EISLAB (EISLab, 2010) and Moteiv (Moteiv, 2006).

2.2.4 Connection-Oriented vs. Connection-Less Sensor Network

In this subsection, we classify sensor network platforms based on the communication architecture employed. We use the term "*connection-oriented*" and "*connection-less*" to classify sensor network platforms. The connection-less approaches use radio that relies on fixed communication frequency (channel). This communication channel is shared by a number of nodes within the communication radius. The connection-oriented approach consists of sensors that use Bluetooth (BluetoothSIG, 2010a) radio for communication, where each node in the network uses a separate channel to transfer data. Figure 2.2 and Figure 2.3 illustrate the connection-less and connection-oriented approach.

The connection-less approach enables data broadcast without establishing a prior connection while the connection-oriented approach requires a connection to be set up before any data transfer. The connection-oriented approach came into being with the wide acceptance of Bluetooth technology. Bluetooth specification (Erricson, 2010) was first developed in 1994 by Ericsson. Bluetooth was initially proposed as a cable replacement technology allowing devices to communicate wirelessly.

Sensor Platform	CPU	Radio	Available Sensors	Memory	Operating System
MICA 2 2001 (CrossbowTechnology, 2010a, Cordeiro et al., 2006)	Atmel Atmega 128L	433 MHz or 868/916 MHz	Accelerometer Ambient Light Pressure & Temperature GPS Microphone Photo-resistor Humidity & Temperature Thermostat	4K RAM, 128 Flash	Tiny OS
MICAz 2004 (CrossbowTechnology, 2010a, Cordeiro et al., 2006)	Atmel Atmega 128L	2.4 GHz IEEE 802.15.4		4K RAM, 128 Flash	Tiny OS
Telos 2004 (ETH-Zurich, 2010, CrossbowTechnology, 2010b)	TI MSP430F1611	2.4 GHz IEEE 802.15.4	Humidity Temperature	10K RAM, 48K Flash	Tiny OS
Moteiv Tmote Sky (Moteiv, 2006)	TI MSP430F1611	2.4 GHz IEEE 802.15.4	Humidity Temperature Light	10K RAM, 48K Flash	Tiny OS
BTNode rev3(ETH-Zurich, 2007)	Atmel ATmega128L	433-915 MHz & Zeevo ZV4002 Bluetooth		64K RAM, 128K Flash	BTnut System Software and TinyOS
Mulle (EISLab, 2010)	Renesas M16C/62	Mitsumi Bluetooth 2.0 OR 802.15.4 - ZigBee	Infra Red (IR), Temperature, Acceleration/ Seismic GPS (Lassen iQ) Pulse (Polar)	20K RAM, 256 K Flash	Proprietary Software. TinyOS and Contiki (Dunkels, 2010) support

Table 2.4: Sensor Node Platforms

The Bluetooth Special Interest Group (SIG) (2010c) created in 1998 allowed a large number of companies to collaborate to improve and standardise the Bluetooth specifications. Bluetooth specification allows connectivity between devices from different vendors, hence, delivering interoperability across a variety of devices. The

Bluetooth specifications are constantly reviewed and updated by the Bluetooth SIG (BluetoothSIG, 2010c) with current version being 3.0 + HS. A new version 4.0 of the specification is to be released by the end of 2010 (BluetoothSIG, 2010c). Bluetooth uses frequency-hopping spread spectrum (BluetoothSIG, 2010a) with time division multiplexing access (TDMA). It operates in the 2.4-2.4835 GHz band. The Bluetooth radio operates by hopping between 79 frequency channels of 1 MHz each, 1600 times, every second. Each time slot has duration of 0.625ms. The Bluetooth specification available from Bluetooth.com (BluetoothSIG, 2010a, BluetoothSIG, 2010c) provides detailed functioning of Bluetooth.

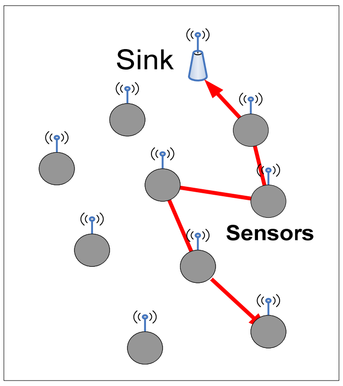


Figure 2.2: Connection-less multi-hop sensor network

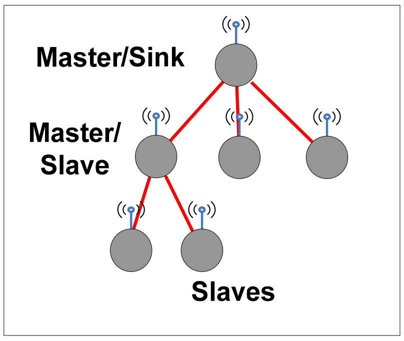


Figure 2.3: Connection-oriented Bluetooth-based sensor network

2.2.5 Sensor Network Applications

Sensor networks have certain unique characteristics that open the door for a wide range of applications. These characteristics are:

Low-powered: This characteristic of sensor network is both a feature and a design requirement. As elaborated in previous section, the sensor components (primarily hardware) need to be designed with the focus of low-powered operation. This allows the sensor to survive on battery for longer time periods. This is useful, for example in traffic monitoring applications (iRoad, 2010, Coleri et al., 2004) that require extended periods of operations without frequent recharging.

Self Organising / Autonomous: This characteristic of sensor networks makes them an excellent choice for applications where manual network configuration is not possible. The self organizing capability allows the sensor to adapt itself into the existing infrastructure or in new infrastructures. This is useful for example in environmental monitoring applications (Steere et al., 2000, Ramanathan et al., 2005, Rytter, 2003, Werner-Allen et al., 2006, Xu et al., 2004) that require sensors deployed in remote locations (forests) to autonomous form a network.

Wireless Communication: The wireless communication available on the sensor allows applications to connect to the sensor without the need to be physically present within the monitored environment. This has created opportunities for new applications including environmental monitoring, health-care and traffic monitoring that depend on data from different locations, something that was previously not feasible.

Distributed Sensing and Processing: We characterise sensor networks as a distributed data sensing and processing platform due to their sensing and processing characteristics.

The above characteristics of sensor networks enable acquisition of data from distributed locations which previously were expensive, difficult or even impossible (Chu et al., 2006). To summarise sensor network applications we have created taxonomy of applications presented in Figure 2.4. We classify sensor network applications based on their operating characteristics, namely:

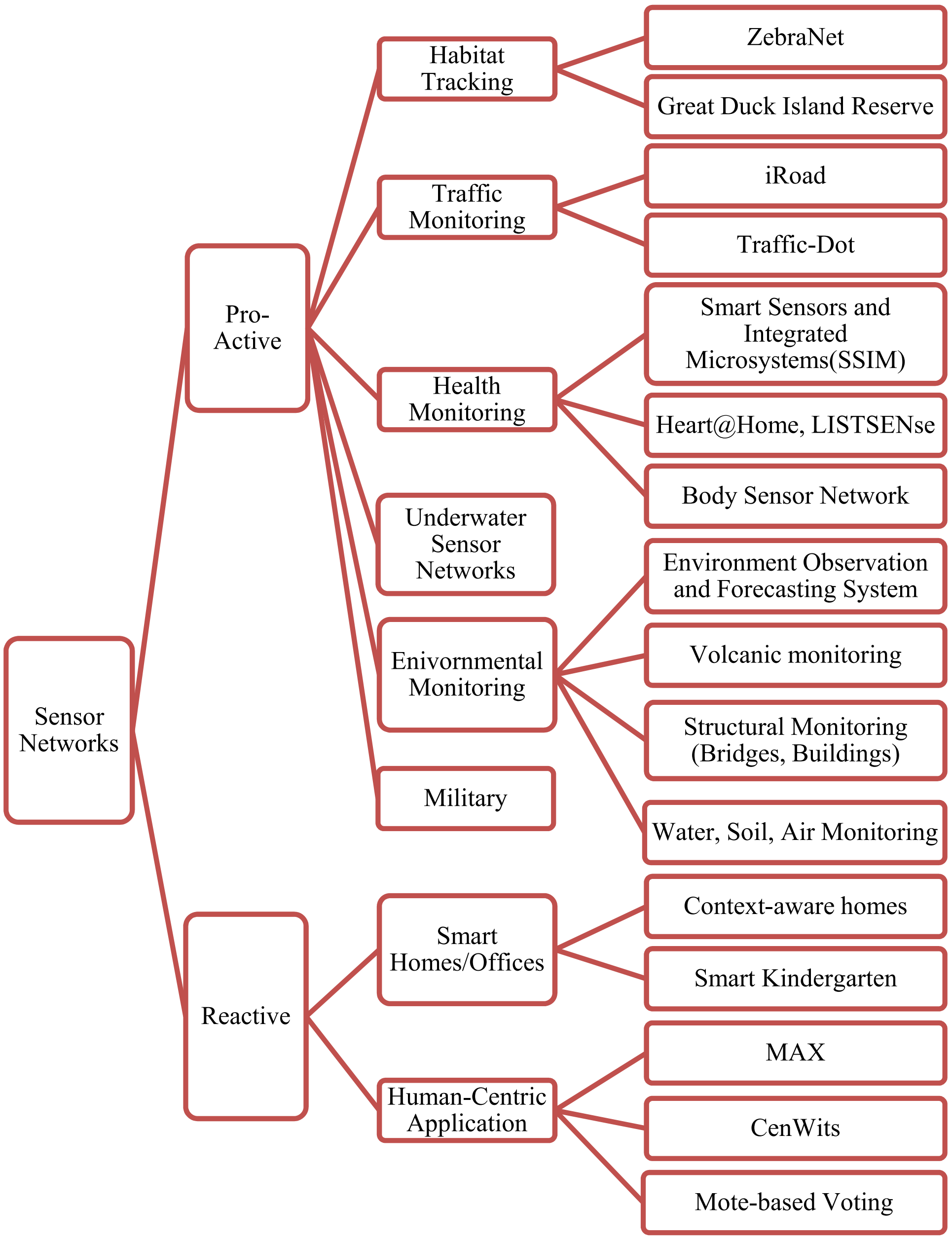


Figure 2.4: Taxonomy of Sensor Network Applications

***Pro-Active*:** The sensor wakes up at periodic intervals, collects data and transmits it to the base station. The sensor also reacts to events i.e. sudden change in the value of the sensed attribute.

***Reactive*:** In addition to the pro-active functions, the sensor employs actuators to react to any sudden change in the parameters monitored. For example, in case of fire in a building, the pro-active mode reports the fire while the reactive mode activates the sprinklers within the area of the fire. This mode refers to wireless sensor and actor networks (Akyildiz et al., 2004) where the actors perform certain system operations based on sensed inputs.

The list of applications presented in Figure 2.4 range from Habitat monitoring (Zhang et al., 2004, Mainwaring et al., 2002), Military Applications (Li et al., 2002, He et al., 2004), Traffic Monitoring (iRoad, 2010, Coleri et al., 2004) including Vehicular Accident Monitoring and Notification (Acharya et al., 2008), Health Monitoring (Baker et al., 2007, Yang, 2010), Environmental and Structural Monitoring (Steere et al., 2000, Ramanathan et al., 2005, Rytter, 2003, Werner-Allen et al., 2006, Xu et al., 2004), Underwater Sensors (Vasilescu et al., 2005) to Smart Home/ Office applications (Meyer et al., 2003, Srivastava et al., 2001, Rabaey et al., 2000) and Human-Centric applications (Yap et al., 2005, Huang et al., 2005). The examples provided for each application domain highlight some important projects in the area while a plethora of sensor network applications surveyed is available in the literature (Arampatzis et al., 2005, Gharavi et al., 2003, Shen et al., 2001, Xu, 2009).

2.2.6 Sensor Networks Challenges: A Roadmap

The previous sections presented a roadmap on sensor operations, current state-of-the-art sensor platforms and applications. The early adoption of WSN's across a varied range of applications has introduced a series of challenges that need to be addressed for long-term adoption of sensor networks. We classify the challenge faced by sensor networks into hardware and software. A taxonomical representation of our classification is presented in Figure 2.5. Research in the past decade has addressed major challenges facing sensor networks including self-organization, data-centric routing, low-power hardware design, efficient power management, security, etc. A major focus of this dissertation is sensor data collection. A detailed discussion on sensor data collection is presented in sections 2.3 and 2.4.

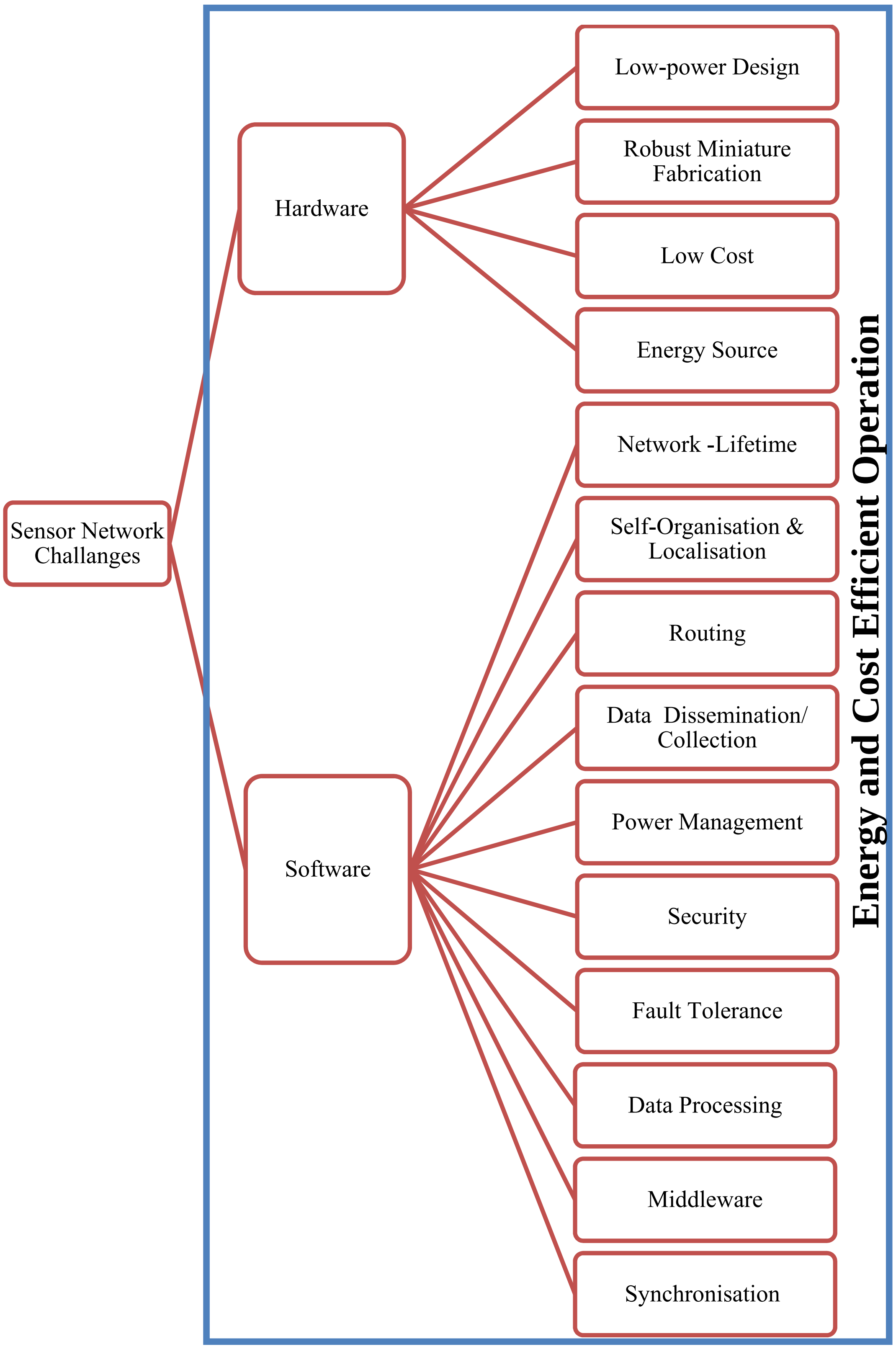


Figure 2.5: Sensor Network Challenges- A Roadmap

2.3 Data Collection in Wireless Sensor Networks

Section 2.2 presented a background on sensor network development and operations identifying the wide range of sensor applications. We identified the key requirement of sensor networks is energy-efficient operation which influences both hardware and software design/development. The advent of pervasive computing (Weiser, 1999) and pervasive computing applications (Burke et al., 2006) has attracted enormous research interest in sensor networks.

The word pervasive which means to pervade[4] is defined as *"to become diffused throughout; every part of"*. This definition suits well within the sensor network paradigm as deployed sensors fuse themselves within the environment. The data generated by sensor nodes is highly valuable. Hence, sensor networks are referred as data-centric networks (Al-Karaki et al., 2004) i.e. data is obtained from the sensor network based on application requirements. For example, if the requested information is *area where humidity > 80%,* sensors that have humidity reading greater than 80% need to respond. These operations of sensor networks require energy-efficient protocols for data collection and delivery. We classify sensor data collection approaches into two categories presented in Figure 2.6.

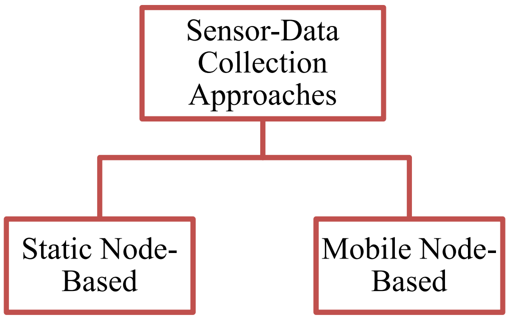


Figure 2.6: Broad classification of data collection approaches

Static Node-Based: We use the term *static node-based* data collection to classify the data collection approaches that employ static sensors/ sinks for data collection.

[4] http://www.merriam-webster.com/dictionary/pervade

Mobile Node-Based: We classify the data collection approaches that employ *mobile data collectors* to collect data from sensors as *mobile node-based*.

2.3.1 Static Node-Based Sensor-Data Collection Approaches

The static node-based data collection approaches adapt the operation of ad-hoc network routing protocols. The basic principle is to propagate a query into the sensor network from the sink and wait for the network to respond with the appropriate data. The sensed data can then use two approaches for data delivery, namely, end-to-end and aggregated. The end-to-end and data aggregation approaches are illustrated in Figure 2.7.

The end-to-end approach, as depicted in the Figure 2.7, delivers the un-modified data from the source to the sink using a multi-hop strategy, i.e. all the sensed data is routed to the sink. The aggregated approach, as depicted in Figure 2.7, employs data aggregation i.e. data is combined by intermediate sensors nodes eliminating redundant transmissions.

As illustrated in Figure 2.7, the end-to-end approach has higher communication overheads compared to the in-network aggregation approach which fuses incoming data packets from different nodes into one single outgoing data packet. The in-network aggregation techniques have proved to be more energy-efficient (Krishnamachari et al., 2002) than the end-to-end data collection techniques. The in-network aggregation allows sensors to inspect relayed packets instead of acting as mere forwarding agents. The data collection approaches that employ in-network aggregation techniques can be classified based on the type of network infrastructure. Figure 2.8 presents taxonomy of these data collection approaches.

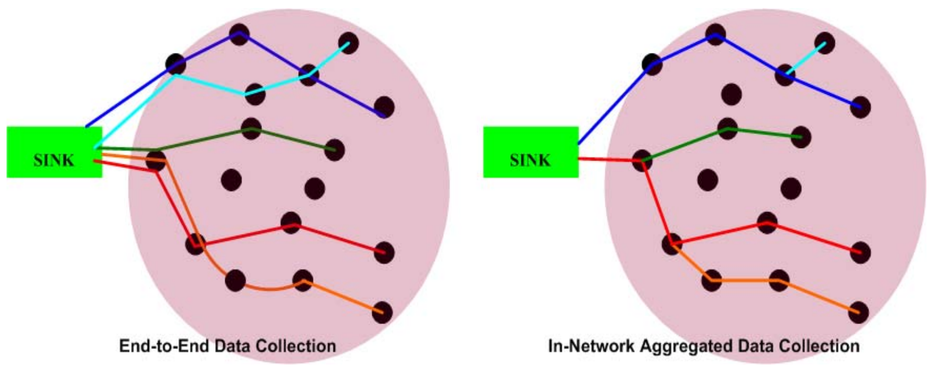

SINK
End-to-End Data Collection
SINK
In-Network Aggregated Data Collection

Figure 2.7: End-to-End vs. In-Network Aggregation Approaches

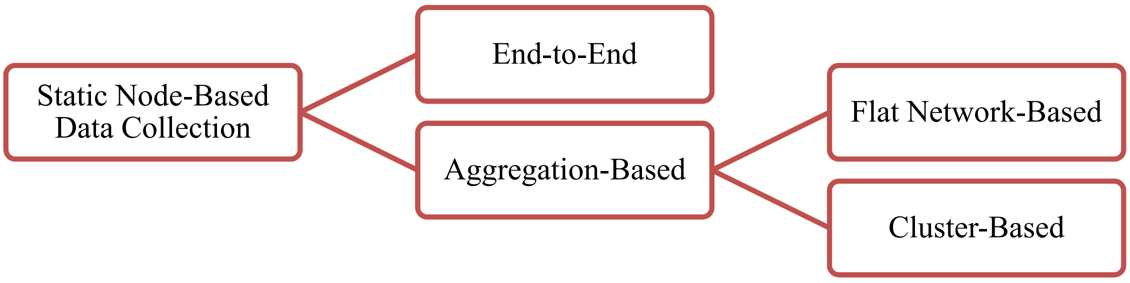


Figure 2.8: Taxonomy of Static Node-Based Data Collection Approaches

2.3.1.1 End-to-End Data Collection Approaches

The end-to-end data collection approaches are adaptation of ad-hoc techniques to suit sensor network requirements (Orecchia et al., 2004, Williams et al., 2002, Hedetniemi et al., 1988). These approaches are initial research efforts in the area of sensor data collection and focus more on data routing which we classify under data dissemination and collection i.e., propagating a request into the network and creating a route to deliver the collected data. Gossiping and flooding are the classic sensor data relaying techniques (Orecchia et al., 2004, Hedetniemi et al., 1988).

Flooding (Williams et al., 2002) is a simple but expensive technique for sensor data collection. It uses a one-to-all (broadcasting) approach. The sink creates a broadcast packet that is sent to the entire network. Sensors that are part of the request respond to the broadcast message. Sensors which do not belong to the request forward it to neighbouring nodes. This approach as stated earlier is very expensive as it requires the entire network to be listening to broadcast packets. It also faces the following drawbacks (Heinzelman et al., 1999):

Implosion: Nodes receiving multiple copies of the broadcast messages and sensed data.

Overlap: Nodes covering same region results in data redundancy

Resource Blindness: The approach is not adaptive and resource aware, hence depleting sensor energy rapidly.

Gossiping (Haas et al., 2002, Hedetniemi et al., 1988) is an enhancement proposed to solve the drawbacks of flooding approach. It solves the implosion problem by using random forwarding of a broadcast message to a selective list of neighbours. This random selection of neighbours reduces the number of broadcast messages but increases response latency. Though gossiping reduces the number of messages exchanged within the sensor network, it only manages to reduce the rate of energy depletion. Moreover, data overlap problem is not addressed by gossiping.

Orecchia et al. (2004) handles the drawbacks of gossiping and flooding by proposing an *Irrigator* protocol that pre-computes one-hop neighbours to form a virtual topology of the network. Then a gossip based broadcasting protocol, namely, *Fireworks* is used to propagate broadcast messages within the virtual network topology. Irrigator approach employs a controlled broadcast, i.e. messages are disseminated to a subset of nodes rather than the entire sensor network. The Fireworks protocol decides to broadcast a message by tossing a coin. A message is broadcasted to all neighbours with a probability p while it is broadcast only to a set of neighbours with the probability 1-p. Orecchi et al. (2004) approach to broadcast packets within a restricted set of neighbours reduces message transmission but does not reduce energy consumption rather delays the energy decay process.

2.3.1.2 Data Aggregation-Based Collection Approaches

Data aggregation is a widely accepted paradigm for data-centric sensor networks (Krishnamachari et al., 2002, Solis et al., 2006) as opposed to the end-to-end approach discussed earlier. Data aggregation combines data from various sources aiming to reduce redundancy and hence reducing the number of transmissions required to deliver the data to the sink. Data aggregation has been employed in sensor networks by processing data within the network, namely, employing the technique called *in-network aggregation*. The three major aggregation techniques (Kulik et al., 2008) are: 1) *Packet Merging (Concatenation)*, 2) *Partial Aggregation (Aggregator) and* 3) *Suppression*.

The *packet merging (concatenation)* technique going by its name combines data from different sensors into a single data packet. The *partial aggregation (Aggregator)* is used when data from different sources needs to be combined to compute one output. For example, combining humidity readings from various sensors in the surrounding and transmitting only the average value. The s*uppression* technique employs suppression operation by discarding redundant data. For example, if a sensor reading does not change from previous measurements, the data is not transmitted.

2.3.1.2.1 Flat Network Structure-Based Approaches

The flat network structure is a single tier sensor network. The flat network structure is also termed a tree-based approach (Nath et al., 2004, Kulik et al., 2008). In tree-based approaches, the root node is responsible for sending a query into the network. Leaf nodes i.e. child nodes perform data sensing and aggregation while relaying data back to the root node. Hence, data from one leaf node propagates to the root via other intermediate leaf nodes.

Sensor Protocol for Information via Negotiation (SPIN) (Kulik et al., 2002, Heinzelman et al., 1999) is one of the early approaches for sensor data collection. The SPIN approach tries to overcome the shortfalls faced by flooding and gossiping approaches by introducing sensor negotiations. The sensor negotiation allows sensors to advertise availability of data with interested sensors subscribing to receive data. The SPIN protocol works in the following way. Each source sensor node advertises data availability to neighbouring nodes. Nodes that are interested in the data send a request to receive data. On receiving a request, the source transmits actual data to interested nodes. The data advertisement contains meta-data descriptors that describe the available data. To achieve this function, SPIN uses three messages, ADV message to advertise meta-data, REQ message to place a request and DATA message to send the actual data. The energy consumption of SPIN is due to frequent propagation of ADV, REQ and DATA messages among interested neighbours. The advertising mechanism of SPIN makes it unsuitable for many applications as sensor data is made available only to sensors that are interested. For example, if the nodes interested in the data and the source node providing the data are separated by intermediate sensors that are not interested in the source data, the data may never reach the sensors that are interested. Moreover, frequent exchange of negotiation messages increases communication overheads.

Intanagonwiwat et al. (2000) propose *Directed Diffusion*, a well-known paradigm for sensor data dissemination and collection. Directed Diffusion is a data-centric protocol that diffuses data through the sensor network using naming schemes. The protocol functions as follows. An *interest* message is propagated by the sink into the network through neighbouring nodes. The interest message is a query containing parameters like location, attribute-value pair, alive-duration, etc. Every receiving node forwards the interest packet to its neighbours setting up a gradient path. A gradient path is a reply-link path identifying the source of the interest packet. The gradient path is then used to determine the route between the sensor data source and the sink. In directed diffusion, sensor nodes within the network have the ability to perform data aggregation. Directed Diffusion approach enables the network to compute multiple routes to data source from the sink, dynamically avoiding overheads to maintain permanent routes. The communication involved in Directed Diffusion is only between neighbouring nodes and hence does not require a single node to have a map of the entire network infrastructure. The energy cost of Directed Diffusion is from computing the gradients dynamically. Directed Diffusion differs from SPIN in terms of data collection approach. Directed Diffusion uses queries propagation while SPIN requires the sensor to advertise data availability.

Tiny AGgregation (TAG) (Madden et al., 2002) is another popular aggregation scheme for sensor data collection. TAG functions in two phases *Distribution* and *Collection*. The distribution phase is used to distribute the query within the network and the collection phase handles collecting aggregated data. To collect data, the sink appoints itself as root of the tree. It then broadcasts its level and identifier to the surrounding sensors. Each sensor with unassigned level assigns itself a level as an increment to the one in the broadcast message. This message is further broadcast with new identifier and level. This process continues until the entire network is reached. The parent node sets a receive time interval during which it listens to the channel for responses from child nodes. Each child node uses the parent's time interval for synchronisation. In the data collection phase, data from the child nodes are gradually propagated towards the sink during each parent's receive interval. Aggregation operation is performed in TAG at every level. The primary energy consumption in TAG is the requirement for parent nodes to constantly listen to the channel. To further optimize TAG, Madden et al. propose semantic routing tree (SRT) (Madden et al., 2005, Madden et al., 2003) algorithm. SRT further optimizes TAG by selectively propagating the broadcast message to a set of sensors which fall within the scope of the query. SRT optimizes the *distribution* phase of TAG by reducing the number of messages broadcast. To achieve this, each parent that receives the query forwards it to its children only if it satisfies a given predicate. The SRT approach is energy-efficient with assumption that parent's nodes are aware of the entire child node's attributes (sensed values). Maintaining this network infrastructure is expensive due to the overheads involved in frequent message exchange.

2.3.1.2.2 Cluster-Based Approaches

Cluster-based network structure employs physical clustering techniques i.e. sensors are grouped into physical clusters where cluster heads are elected to handle data processing, aggregation and additional communication operations. A number of clustering techniques employed in sensor networks are presented in the literature (Abbasi et al., 2007, Younis, 2004, Bandyopadhyay et al., 2003, Younis et al., 2002, Ghiasi et al., 2002, Lindsey et al., Heinzelman et al., 2000, Moussaoui et al., 2005). Each cluster encloses child nodes that transmit sensed data to the cluster head for further processing and delivery. A typical cluster with cluster heads and child nodes is illustrated in Figure 2.9. The data from the child nodes are transmitted to the cluster head (indicated by the lines connecting cluster head and child nodes). The cluster head delivers the data collected from the child nodes to the sink. Clustering techniques are similar to tree-based techniques but have specific clustering protocols to group sensors into physical clusters. The cluster establishment is based on number of parameters e.g. residual energy, geographic location, sensor type, etc. The cluster-based approaches also employ a different communication strategy than tree-based approaches. In tree-based approaches, each sensor in the network transmits the data to the base station (sink) using direct or multi-hop technique. In a cluster-based approach, cluster members transmit data to the cluster head which transmits the data to the base station directly or via other cluster heads (multi-hop clusters).

Chien-Chung et al (2001) present Sensor Information Network Architecture *(SINA)*, a sensor data collection approach based on querying. In SINA, the sensor network is conceptually viewed as a collection of massively distributed objects. SINA employs hierarchical clustering in order to improve network lifetime and performance. The cluster heads perform information filtering, fusion and data aggregation. SINA uses a spreadsheet paradigm for querying and monitoring. In the spreadsheet paradigm, each sensor node maintains a logical datasheets consisting of cells. Each cell is uniquely named and represents an attribute of the sensor node (e.g. remaining battery power). The value stored in a cell can be queried by other nodes in the network using a sensor query and tasking language (SQTL), a procedural scripting language designed to be flexible and compact. SINA plays the role of a middleware responsible for fetching information from the sensor network based on the user queries.

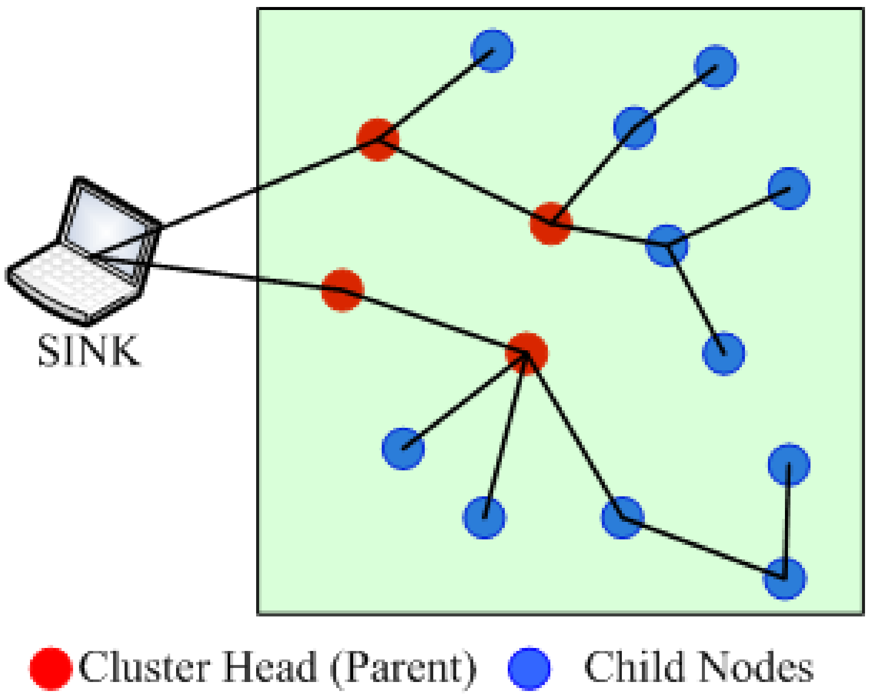


Figure 2.9: Cluster-Based Sensor Network Structure

Low-Energy Adaptive Clustering Hierarchy (LEACH) (Heinzelman et al., 2000, Heinzelman, 2000) was an initial effort towards an adaptive clustering scheme in sensor networks. LEACH employs random cluster head rotation to efficiently balance the sensor network energy. LEACH operates in multiple rounds with each round performing the operation of cluster formation and data transmission. In cluster formation, a node becomes a cluster head based on certain probability. The cluster is then computed taking minimum communication energy factor into consideration i.e. a node decides to participate in a cluster if the communication energy to communicate with the cluster head is least compared to other cluster heads. During data transmission, all nodes transmit data to the cluster head at specific time intervals. The cluster head performs data fusion (aggregation) before forwarding it to the sink (base station). A variant of LEACH, LEACH-Centralised (LEACH-C) is proposed in (Heinzelman et al., 2002), that employs a centralised clustering algorithm to reduce the energy spent by non-cluster heads to transmit data to cluster heads. To achieve this, sensors transmit location and energy information to the base station. The base station then determines average energy threshold above which a sensor can become a cluster head. This information is then propagated back into the network. LEACH approach is energy-efficient but does not guarantee equal distribution of cluster heads. LEACH-C performs better than LEACH (Heinzelman et al., 2002), nevertheless a centralised approach may not be feasible for many applications.

Power-Efficient Gathering in Sensor Information Systems (PEGASIS) (Lindsey et al., 2005) is a chain-based clustering protocol proposed as an improvement to LEACH. PEGASIS employs a greedy algorithm to organise nodes into a chain. In each round, exactly one node transmits data to its neighbouring node. This is done using a simple token mechanism. Each node receiving data from the neighbour fuses it with its own data into a single packet and forwards it to the base station. Nodes in the chain take turns to transmit the data to the base station hence spreading the energy consumption uniformly across the entire network. PEGASIS like LEACH requires global network knowledge to compute the chain.

Hybrid Energy-Efficient Distributed Clustering (HEED) (Younis, 2004) is proposed as an improvement to LEACH by focusing on efficient cluster formation. HEED is a multi-hop clustering algorithm i.e. multiple cluster heads are used to relay the data to the base station rather than a cluster head - base station communication. HEED aims to reduce overheads of cluster head election hence, extending the overall network lifetime. It computes clusters based on sensor residual energy and intra-cluster communication cost. The sensor residual energy is used to elect cluster heads while the intra-cluster communication cost is used by members to determine the cluster to join. HEED evaluation results are favourable compared to LEACH and its variants, nevertheless the approach still needs to re-elect cluster heads periodically which proves energy inefficient.

A number of research approaches similar to LEACH, HEED and PEGASIS have been proposed extending single level clustering to multi-level clustering. Bandyopadhyay et al. (2003) present a *hierarchal clustering algorithm* that computes multi-level clusters enabling multi-hop clustering. Moussaoui et al. (2005) proposes a *distributed energy-efficient clustering hierarchy protocol (DECHP)* that uses ideas similar to (Bandyopadhyay et al., 2003), employing multi-level cluster heads to relay data from cluster members to the base station (sink). The experimental observations of both approaches show improved energy-efficiency with increasing clustering levels. On contrary, the multi-level clustering approaches introduce additional overheads in constant cluster maintenance.

2.3.1.3 Summary of Static Data Collection Approaches

This section presented a summary of research in static node-based sensor data collection. The use of data aggregation in static node-based data collection has evolved into an energy-efficient sensor data collection paradigm. Cluster-based approaches have efficiently employed data aggregation to further improve the energy efficiency in data collection over tree-based approaches. The major drawback of static-node based approaches is the overheads involved in collecting data using a fixed base station. Cluster-based approaches introduce more overheads caused by cluster maintenance, hence increasing energy consumption. A comparison of the static node-based approaches is presented in Table 2.5.

Technique / Characteristics	Flooding / Gossiping	Flat Network-Based Approach	Cluster Based Approach
Data Aggregation	NO	YES	YES
Query Based	NO	YES (Certain Approaches)	YES
Energy Consumption	HIGH	HIGH	Better than Flat Network-Based but consumes energy in cluster maintenance
Global Network Knowledge	NO	NO	YES (Most methods require global information to compute efficient clusters)
Features	Simple protocol extended from ad-hoc networking	Robust operation for node failures. Multi-route capability. Does not require global network knowledge. Single/ Multi-Hop data delivery	Clusters adaptation based on sensor residual energy. Multi-Hop clustering capabilities Single/ Multi-Hop cluster to base station communication
Limitations	Leads to broadcast storms (Tseng et al., 2002). Does not scale well for large sensor deployments.	Energy inefficient for changing network conditions. Some approaches result in broadcast storms (Tseng et al., 2002).	Requires Global Network Knowledge. Energy consumption during cluster head rotation.

Table 2.5: Summary of Static-Node based Data Collection Approaches

2.3.2 Mobile Node-Based Data Collection Approach

The static node-based approaches discussed in the previous section introduced energy-efficient data collection approaches for sensor networks with a fixed sink. Recently, mobility-based approaches have been explored as an alternative to static node-based approaches. The mobility-based techniques aim to further improve energy-efficiency of the data collection process. We use the term *"mobile node"* to represent the class of mobility enabled platforms/devices that are employed for sensor data collection. Introducing mobility to collect sensor data has certain advantages namely:

Coverage/Connectivity: Use of mobile nodes facilitates data collection from disconnected sensor networks. This is applicable in both *sparse* and *dense* sensor network deployments. In *sparse networks* two nodes may not be in communication range hence requiring sensors to be equipped with long range radio communication hardware (Anastasi et al., 2009c, Venkitasubramaniam et al., 2004, Jenkins et al., 2007, Shah et al., 2003, Jain et al., 2006). This requirement solves coverage problems but imposes high energy requirements. In *dense networks* the problem of coverage arises when relay sensors fail or have different duty cycles isolating specific parts of the network. One solution to this approach is periodic synchronisation between the sensors. Though this is a viable option, it is also high energy consuming (Dini et al., 2008).

Sensor Lifetime: The single-hop/multi-hop strategies employed by static data collection approaches depend on intermediate nodes to deliver the data to the sink. These approaches were acceptable as early research in sensor networks assumed a sensor model with fixed sensor nodes and static sinks, as depicted in Figure 2.9. The static node-based multi-hop data collection strategies deplete the energy of relay sensor nodes transmitting the data. More specifically, sensor nodes near the sink deplete energy at a higher rate than other nodes in the network (Kansal et al., 2004). Employing mobility-based data collection leverages the idea of mobile nodes moving closer to sensor nodes, reducing the number of hops involved in transmissions. Reducing multi-hop transmissions reduces the number of packets exchanged within the sensor network, hence reducing the overall energy consumption. As stated previously communication is a major energy consuming operation hence, reducing the overall packet transmission plays a substantial role in increasing sensor lifetimes.

Deploy/ Calibrate/ Recharge Sensors: The introduction of mobility can help to assist in a number of sensor operations which previously were impractical. LaMarca et al.

(2002a, 2002b) present a novel way of using mobile robots to deploy, calibrate, recharge and maintain WSN. Rahimi et al. (2003) present the use of mobility to harvest energy. The technique employs mobile sensors with capability to discover rechargeable energy sources delivering energy to static energy-depleted sensors. (Eliasson et al., 2006, Elson et al., 2003) propose different approaches for time synchronization in sensors.

The range of applications that can take advantage of mobile nodes in sensor networks is vast many of which are not practical using static sink-based techniques. A typical mobile node-based data collection scenario is presented in Figure 2.10. The mobile node moves along a path around the sensor network area collecting data from nearby nodes delivering it to the sink.

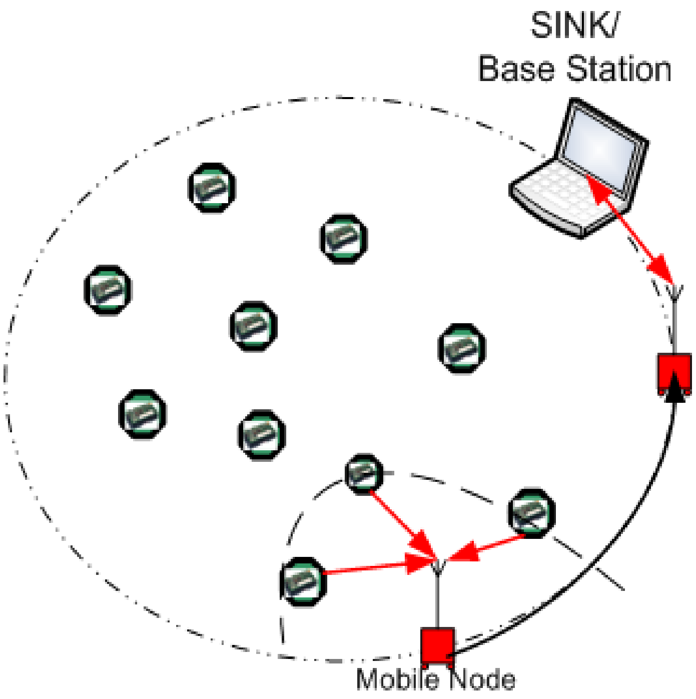


Figure 2.10: Mobile Node-Based Data Collection

We classify mobility in sensor networks into the following three types:

Mobile Sensors: Mobile sensors represent a class of sensors that have the capability to move within the sensor network autonomously. The mobile sensors are part of the network deployment infrastructure performing operations including mobile sensing, data collection from non-mobile sensor nodes, etc. Mobile sensors inherit the advantages of mobility but have specialised requirements for practical realisation including specialised hardware for mobility, energy resources to sustain mobility, increased costs in design and development and protocols for autonomous operations.

Mobile Sinks: Mobile sink approach exploits mobility by introducing sink/base station mobility. This allows the base station (sink) to move around the sensor network performing data dissemination and collection. Though making base stations mobile is a viable option, it involves overheads in constantly updating the entire network with the base station's location/movements.

Mobile Data Collectors: Mobile data collectors exploit the existence of mobile nodes within the environment to collect sensor data. For example, introducing a specially designed mobile robot for data collection (Kansal et al., 2004) or using existing mobility enabled objects (e.g. bus, cars) as data collectors (Shah et al., 2003). The mobile data collector acts as a relay node to deliver the data to the sink. It does not have sink capabilities.

Mobile node-based data collection approaches are further classified on the basis of mobility pattern employed. The sensor mobility pattern can be broadly classified into Random Mobility, Controlled Mobility and Predicted Mobility (Schindelhauer, 2006). Our classification is presented in Figure 2.11.

In random mobility (Shah et al., 2003, Jain et al., 2006) mobile nodes with random movement patterns collect sensor data e.g. humans, cars, animals, etc. In predicted mobility the mobile node's movement pattern is fixed (pre-known), for example, a data collector mounted on shuttle buses within the campus (Chakrabarti et al., 2003). In controlled mobility the movement of the mobile node is controlled by the application (Basagni et al., 2007, Jea et al., 2005, Kansal et al., 2004) adding flexibility to the data collection process.

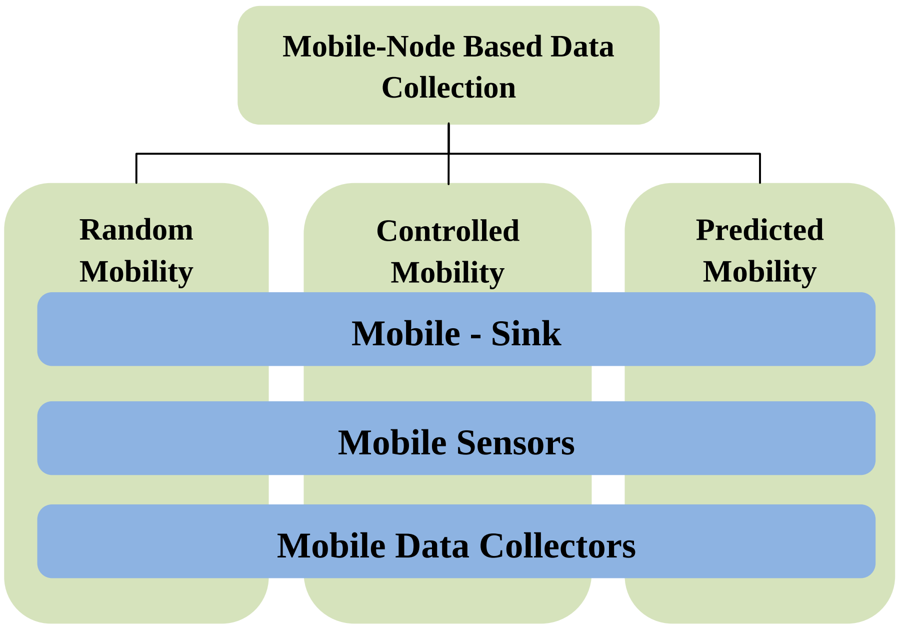


Figure 2.11: Classification of Mobile-Node based Data Collection Approaches

2.3.3 Mobile Sensor-Based Data Collection

Mobile sensor networks comprise a distributed collection of sensor nodes with locomotion capabilities. These sensors have the same functionality as the static sensors with added capabilities for localisation, navigation, path planning, etc.

Zebranet (Zhang et al., 2004, Juang et al., 2002) project is a classic example of a mobile sensor network. *Zebranet* system focuses on wildlife tracking, with real life implementation monitoring of Zebras. In *Zebranet*, each animal is equipped with a special collar that includes a global positioning system (GPS), dual radio (a short range and long range radio), solar panels and rechargeable battery. Data collected from each zebra is delivered when the zebra comes within communication proximity of the fixed base station (sink). The system also facilitates inter-zebra communication allowing multi-hop data collection/delivery. The communication protocol employed is simple flooding which is energy consuming. Hence, to save energy, a history-based data collection scheme is proposed that employs mobile node hierarchies. The hierarchy determines each mobile node's (Zebras) data delivery rate. A higher value indicates higher communication contact with the base station and vice-versa. Mobile nodes within communication range request for hierarchy level information of peer nodes. Data is offloaded to the mobile node with the highest hierarchy level value. This facilitates better data delivery rates with lesser energy consumption. The simulation outcomes presented in the paper validate the energy-efficient performance of the history-based data collection protocol over the flooding approach. A similar approach is extended in (Small et al., 2003) employing whales as mobile sensors. Small et al (2003) propose S*hared Wireless Infostation Model* (*SWIM*), a networked architecture that facilitates information propagation by means of mobile sources. To this end, the *SWIM* approach has been studied within the scope of biological information acquisition, targeting whale monitoring. *SWIM* employs mobile base stations (termed *SWIM* stations) that float within the area of whale movement. When the whale arrives within close contact of the *SWIM* station, it offloads data collected to the *SWIM* station.

More recent research with focus on mobility enabled sensor platforms can be found in the literature (Paredis et al., 2002, McMickell et al., 2003, Bergbreiter et al., 2003, Dantu et al., 2005). Most of these approaches introduce mobility by means of specially designed hardware rather than the *Zebranet* and *SWIM* approaches discussed earlier. The *MetroSense* project at Dartmouth (Campbell et al., 2006) is one approach that focuses on collecting data from mobile sensors. The mobile sensors in this case are high powered devices e.g. personal digital assistant (PDA) equipped with sensors. The project focuses on people-centric sensing, a term used to represent collaborative data sensed from people. *CarTel* (Hull et al., 2006) is another example of a distributed mobile sensor system. *CarTel* is designed to collect process and deliver data obtained from mobile sensors to a central store for further processing and visualisation. The system itself is equipped with sensors with imaging and location abilities installed on commuter cars. As these commuter cars move randomly, the sensors collect local information (e.g. pictures of streets/roads) delivering it to a central sink using delay-tolerant opportunistic networks (e.g. WiFi available within the commuter car’s range). The *CarTel* project aims to augment content provided to user with rich valuable information, e.g. a photo of a house rather than information about the house’s location. As the *CarTel* hardware is installed on cars, energy-efficient operation is not a key concern for such type of mobile sensors. This allows them to be equipped with more powerful communication hardware e.g. WiFi cards. Adding mobility to sensors introduces additional challenges of controlling sensor movement autonomously. Use of random mobility eases this problem, but in applications where mobile sensors are introduced, the area of mobile sensor research maps closely with robotics (Howard et al., 2002).

2.3.4 Mobile Sink-Based Data Collection

Mobile Sink approaches discussed in the literature employ mobility to solve the problem of non-uniform energy depletion in the sensor network i.e. reduce the burden on sensor nodes placed around the sink. These sensors placed around the sink drain energy faster than other sensor nodes in the network. A SEnsor Network with Mobile Access points (SENMA) is proposed in (Venkitasubramaniam et al., 2004). SENMA employs powerful mobile access points equipped with long-range radio communication hardware to communicate and collect data from low-powered sensor nodes. An example of one such power access point is an unmanned aircraft flying over the sensor network terrain. Data collection is done by the mobile access point by sending a wake-up beacon to sensors within the terrain. Each sensor activated by the mobile AP starts sending a packet with a probability *p*. SENMA considers single hop data collection using mobile sinks. This work is further extended in (Mergen et al., 2006) by optimizing network parameters, namely, coverage area, flying altitude and mobile access point trajectory aiming to improve data collection efficiency. Some interesting insights highlight the importance of channel quality (signal-to-noise ratio) to maximize network lifetime. SENMA approach falls within controlled mobility as the case study employing unmanned aircraft presented in the paper can be controlled to optimize the data collection process.

Liang et al. (2005) explores sink mobility in dense sensor networks as against SENMA which primarily focuses on sparsely deployed sensor network. They propose multiple nodes transmission scheduling algorithm (MTSA-MSSN) that handles transmission from multiple nodes that use a single channel for communication. The mobile sink modelled in this proposal moves with specific velocity and direction collecting data at pre-defined time intervals (predicted mobility). MTSA-MSSSN requires nodes to have certain signal-processing capabilities to participate in the scheduling process. SENMA focuses more on random channel access while MTSA-MSSSN focus on time synchronized multiple channel access with higher efficiency.

Some mobile sink-based data collection approaches formulate the data collection problem as a Linear Programming problem. Gandham et al. (2003) explores mobile sink placements using an integer linear programming (ILP) approach in a multi-hop sensor network. The data collection operation is divided into rounds and at the beginning of each round, the best location for mobile sink placement is determined. The ILP formulation aims at minimizing the maximum energy spent by the sensor node during each round to communicate with the sink. Gandham et al. (2003) achieves energy-efficient data collection by reducing the maximum energy spent at each node which helps as an indicator towards the overall network lifetime. Alternatively, Wang et al. (2005) presents a similar mobile sink solution that directly formulates the ILP on the overall network lifetime. In this approach, sensors are modelled in a two-dimensional square grid. The mobile sink moves within the grid spending a fixed time interval collecting sensor data. The objective of the ILP is to determine which node needs to be visited next by the mobile sink and the optimal sojourn time at each point. The results of evaluations from (Wang et al., 2005, Gandham et al., 2003) indicate the performance improvements using mobile sink over static sink approach. Specifically, Wang et al. (2005) claims 500% improvement in network lifetime over static sink based approach. The proposed ILP approaches mostly follow random mobility patterns with the assumption that mobile sink appears equally at every node location (same frequency of visits). Finally, Chatzigiannakis et al. (2006) present a mobile sink-based data collection approach with focus on studying the impact of mobility patterns and data collection strategies. The mobility patterns simulated in the paper include random and predicted mobility while data collection strategies explore passive, limited and complete multi-hop routing.

Kansal et al. (2004) propose the use of mobile robots to achieve sink mobility. To this end, the authors propose a prototype implementation of a mobile base station with the ability to move within the sensor network along a specific path. The mobile base station used (also termed mobile router) is a rugged multi-terrain unmanned ground vehicle, namely, Packbot. The robot's movement is controlled using Simple Interface for Robots (SIR), a generic development platform. The robot is equipped with a Stargate node processing platform running Linux and a mica mote interface hardware. The interface hardware is a mica-mote that can communicate with sensors in the surrounding area. The mobile base station's movement can be controlled to increase system performance. The prototype implementation uses a mica mote (CrossbowTechnology, 2010a) sensor node. Further, Kansal et al. (2004) explores the following challenges involved in data collection when the mobile base station is on-the-move: 1) mobile base station's movement speed influencing data collection and 2) data Collection protocol to enable multi-hop data collection. While previously discussed approaches require the base station to stay at the node's location for a constant time, Kansal et al. (2004) employs speed control algorithms that can vary the mobile base station's velocity based on sensor nodes in the environment. The algorithm works as follows. An initial exploration round is used to determine and collect state information from the set of node within the vicinity of the mobile base station. The state information is further used to partition the sensors into two sets based on data delivery success percentage. The mobile base station employs a stop-and-collect approach for sensor nodes in the first set and a slow-down-and-collect approach for sensor nodes in the second set. Finally, a data collection (communication) protocol based on directed diffusion (Intanagonwiwat et al., 2000) is proposed to enable multi-hop data collection. Nodes that are not in direct contact with the mobile base station (sink) offload their data to a node that is in direct contact with the mobile base station. The communication protocol employs an acknowledgement scheme to notify nodes of successful data delivery. Nodes failing to receive an acknowledgement will re-transmit their data. Unlike previously discussed sink-based mobile data collection approaches, Kansal et al. (2004) present results obtained from real-world experiments using the proposed system prototype. Experimental results validate the advantages of employing mobility by achieving better data success rate at lesser energy using the adaptive speed control algorithm and the modified, directed diffusion approach.

Jea et al.(2005) extends Kansal et al. (2004) work by introducing multiple mobile base stations to alleviate latency in data collection process. Two issues discussed are choice of number of mobile base stations and handling shared nodes (nodes within the range of both mobile base stations). They propose a load balancing algorithm that uniformly distributes the load of the data collection process among available mobile base stations. The load balancing process is divided into five parts namely: initialization, leader election, load balancing, assignment and data collection. The end result of the load balancing operation is a node list that each mobile base station needs to service (data collection operation). This list avoids conflicts from sharable nodes as each shared node is assigned specifically to only one mobile base station. The data collection involves the mobile base station traversing the network collecting data from its assigned node list. The mobile base station sends acknowledgement to nodes it services, which registers the base station information. This avoids responding to new broadcast beacons from other mobile base stations.

Somasundara et al. (2006) further explore controlled mobile base station approach by proposing a cluster-based communication protocol to collect data from sensor nodes that are not in direct contact with the mobile base station. The clustering protocol proposed involves a set of network layer algorithms that are responsible for establishing clusters during initial exploration of the sensor network by the mobile base station. The clusters formed are rooted at the mobile base station. Sensors within the cluster offload their data to the cluster heads. The cluster head then employs a round robin delivery approach to transfer the data to the mobile base station. The adaptive speed control algorithm previously introduced is further extended to be adaptive based on information collected from the network. For example, the mobile base station spends more time collecting data from the congested part of the network while spending relatively less time within uncongested parts of the network. The analytical energy comparison of the mobile base station approach validates the use of mobile base station as a viable option for data collection in dense sensor networks.

Ren et al. (2006) present a data collection approach using mobile sinks in a hybrid mobile wireless sensor network model. A hybrid wireless sensor network comprises low-level sensors and high-level mobile devices that perform data sink operations. They investigate the influence of number of mobile sinks, velocity, communication radius and data collection delay. The delay during the end-to-end data collection is dominated by the time the senor node waits for mobile sink to come within communication range. The simulation studies three performance metrics, namely, average data delivery delay, data success rate and lifetime of the network. The sensor model is assumed to be both sparse and densely populated. The simulation also explores data collection using multi-hop strategy. The outcomes show significant savings in energy using mobile sink employing single or multi-hop data collection strategy over static approaches. The mobile data sink approach proposed does not investigate any system architectures or data collection protocols that can be used in real-world situations. Further, no discussion on the specifics of the multi-hop communication strategy employed is presented.

The main drawback with the sink-based approaches is the need for the entire network to be aware of the sink's changing positions. The ILP-based approach presented in (Luo et al., 2005) explores the use of optimal data routing to handle changing sink locations with the assumptions that sensors are aware of sink trajectory and the entire network topology. Wang et al. (2005) assumes sensors are deployed in a square grid while (Gandham et al., 2003) requires network flow (traffic) information at each node to decide the best route to the mobile sink. Moreover, the ILP-based approaches require sensor network topology information to find an optimal solution for the linear programming problem. Further, the sensor nodes need to have the capability to solve the ILP in finite time to obtain the best result if not the optimal result. The controlled mobility-based (Somasundara et al., 2006, Kansal et al., 2004, Jea et al., 2005) data collection approaches provide a good understanding of the challenges involved in designing and implementing a real-world mobile base station (sink). This approach requires introduction of specially designed mobile hardware with capability to continuously traverse the network collecting sensor data. Also, the entire model is based on the assumption of the availability of a dedicated mobile base station within the sensor network. This requirement may not be feasible in many applications. The basic and extended speed control algorithms incorporating network parameters to control the mobile base station's velocity require a training phase for efficient functioning. Moreover, multi-hop data collection is efficiently achieved using clustering which introduces the drawbacks of cluster-based approaches. In addition clustering introduces a training phase to group sensors into different clusters.

2.3.5 Mobile Data Collector-Based Data Collection

Mobile data collector synonymously referred to as mobile relays (Anastasi et al., 2008) or data mules (Shah et al., 2003) by various authors take advantage of mobility for energy-efficient sensor data collection. The previously discussed sink-based mobility approaches focus on reducing the load of sensor nodes around the sink. To achieve network load balance, sinks move randomly or in pre-defined path over time within the sensor network reducing the effect of non-uniform energy depletion. The use of mobile data collectors has been explored in the literature (Jenkins et al., 2007, Shah et al., 2003, Chakrabarti et al., 2003, Di Francesco et al., 2010, Curino et al., 2005, Henkel et al., 2006). The mobile data collector collects data from sensors within its communication range. Such a wireless sensor network contains both a sink and a mobile data collector as part of the network infrastructure. The data collected by the mobile data collector is delivered to the base station/sink which handles further processing of data. Mobile data collectors employed for data collection can be introduced into the infrastructure with the goal to improve data collection efficiency. This kind of approach gives the system the ability to optimise the characteristics of the mobile data collector to achieve pre-defined performance goals. For example, changing the path of the mobile data collector (controlled mobility environment) aiming to improve the data collection rate within certain parts of the network. The base station-based mobility presented in (Kansal et al., 2004) is one such approach. But this method requires the sink to be constantly available within the sensor network. By separating the sink and the data collector, we introduce more flexibility into the sensor network architecture.

Chakrabarti et al. (2003) propose a pioneering effort in employing mobile data collectors for sensor data collection. They employ predicted mobility-based technique to collect data from sensors that are distributed within an area. In this case, an observer whose path is fixed (pre-known) moves around the sensor network periodically. When the observer is within the range of the sensor, it initiates communication. Since the sensor nodes communicate with the observer only when it is in range, the energy spent in collecting data is significantly reduced. The observer is not power-constrained and is equipped with long range radio enabling single hop data collection. Hence, each node needs to be within the range of the observer to off-load their data. Sensor nodes exploit the availability of observer's path/arrival time to efficiently listen to the communication channel only when the observer is within its communication range. The communication protocols employed for data collection works in three phases, namely, *start-up, steady state and failure*. The *start-up* phase operates in two cycles. In the first cycle, sensors listen to the channel waiting for an observer. Once an observer arrives within range, the sensors estimate the duration the observer is within range and the frequency of arrival. In the second cycle, the observer transmits a beacon to which sensors respond with RTS (Request to send). The observer receiving the RTS from a sensor responds with CTS (Clear to send). The sensor responds with a small information packet which is acknowledged by the observer. Cycle one and two are repeated until the entire network is covered. During the steady phase, the data is collected from the sensors by the observer by transmitting a *wake* signal. As the information about sensors and their sleep schedule is available with the observer, it prioritizes data collected based on available information. A mathematically formulated power comparison between the mobile observer based model and static node-based model is presented. Three classic cases are considered, namely, mobile observer model, a static sensor model with single hop communication and a static sensor model with multi hop strategy. The mobile observer approach produces power savings of up to 300 times over single hop static data collection and up to 3 times over multi hop data collection. Though this approach uses existing mobility to collect sensor data energy efficiently, the primary requirement for the proposed system is the training phase. This requirement may not be feasible in many application scenarios. Further, the approach only characterises single hop data collection. This creates the issue of network coverage i.e. every sensor must be equipped with radios powerful enough to reach the mobile observer in one hop.

Shah et al. (2003) propose a data collection architecture that exploits random mobility to efficiently collect data from a sparsely deployed sensor network. The data collection architecture is made up of three-tiers: 1) lower tier occupied by sensors that periodically sense data within the environment, 2) middle tier consisting of *Mobile Ubiquitous LAN Extensions (MULE)*. MULE's are mobile entities that travel around the sensor network collecting sensed data when within communication range and 3) top tier consisting of access points. The access point is a bridge between the MULE and the sink. Data collected from sensors is offloaded to the access point when the MULE is within communication range of the access point. The lower tier sensors await the arrival of the MULE to off-load collected data. This requires them to be continuously monitoring the communication channel. The MULE move independently using a Discrete Random Walk mobility model. The MULE arrival and data collection process is modelled on queuing theory. The system is analysed over key performance metrics, namely, data success rate and sensor buffer size. The use of short range one-hop communication technique reduces the energy consumed during data transmission. Moreover, the introduction of MULE-based data collection architecture alleviates the problem of coverage i.e. the requirement to either introduce additional access points or deploy more sensors to completely cover sparsely deployed sensor networks.

Jain et al. (2006) further extends the MULE architecture by addressing some of the shortfalls of the MULE model (Shah et al., 2003) making it more energy-efficient. The previous approach (Shah et al., 2003) worked under the assumption that sensors are always listening to the communication channel. This operation is expensive and highly energy consuming. Hence, Jain et al. (2006) explore efficient sensor discovery allowing sensor to implement duty cycling. The use of duty cycle can further enhance the lifetime of the sensor. Moreover, they investigate the data collection phase by analysing the influence of MULE's contact time with the sensor. The contact time is the time the sensor and the MULE are within communication range. Finally, the system is analysed with various sensor duty cycles and MULE movement patterns. Interestingly, the analytical model shows that efficiency in data collection is not substantially affected by low duty cycles. Simulation outcomes exhibits energy savings of up to two-orders of magnitude using the MULE approach over traditionally static data collection approaches.

The MULE architecture is the basic foundation in the use of mobile data collectors. But, the proposed approach has certain drawbacks. They are: 1) the sensor needs to listen to the channel continuously to detect MULE arrival, 2) the system is analytically modelled on parameters like sensor buffer, MULE buffer, MULE arrival rate with little attention on the actual data collection process, 3) the system model is assumed to be distributed in a two dimensional grid where sensors, MULEs and access points occupy specific points in the grid. At every clock tick, MULE moves within the grid from one point to another, 4) the system model also assumes an error free communication channel and 5) the MULE architecture handles only direct (single-hop) communication, hence requiring the MULE to cover the entire sensor network. These challenges need to be addressed to implement the MULE architecture in real-world applications. The MULE architecture is best suited for delay-tolerant networks (Shah et al., 2003, Jain et al., 2006) as the MULE needs to wait until it arrives near an access point to deliver the data. This approach causes delay and is also not reliable as MULE might sometimes not arrive at the access point at all.

The most recent work in the area of mobility-based data collection in sparse sensor network has been presented by Anastasi et al. (2009b, 2009a). Their (Anastasi et al., 2009a) work is motivated by the MULE architecture discussed previously with focus on sensor-MULE discovery and data transfer protocol. The problem of sensor-MULE discovery arises when sensors working at different duty cycles miss contact with the MULE. To this end, they propose a discovery scheme based on periodic wakeups and data transfer protocol based on Automatic Repeat ReQuest (ARQ). This approach investigates a single-hop data collection strategy where MULE communicates with sensors when within communication range. The system model simulates realistic message losses experienced in real-world situations. The discovery scheme sends periodic beacons allowing sensors within the range to respond. The sensors within the range employ an asynchronous sleep/wakeup pattern i.e. sensors wake independently of each other, listen to the channel for beacons from the MULE and return to sleep state. The data transfer protocol uses a windowing technique where messages are segmented into windows. Each windows of messages successfully received is acknowledged by the MULE and messages that are lost are retransmitted in subsequent rounds. The simulation outcomes based on the analytical model provides a good understanding of duty cycle influence on data collection efficiency. The efficiency here is more linked towards data success rate i.e. ratio of amount of data available against amount of data collected.

2.3.6 Summary of Mobile Node-Based Data Collection Approaches

The previous section presented a detailed discussion on mobile node-based data collection approaches. As discussed, the use of mobility is advantageous and has a number of benefits that have been highlighted. The mobility-based data collection approaches discussed in literature are more energy-efficient than static sink-based approaches. However both groups of approaches have advantages and drawbacks. For example, the controlled mobility approach requires the introduction of a specially designed mobile sink. The mobile sink approach is rewarding but the absence of a static sink might result in the network overflowing with collected data when the mobile sink is unavailable. Similarly, the mobile sensor-based approaches are suitable for specific applications as sensors need to be equipped with appropriate hardware for mobility. Finally, the random mobility based approaches use mobile data collectors that exist within the environment to perform data collection. These approaches are mostly analytical models with little attention on the energy consumed in the actual data collection process. Moreover, the proposed mobile data collectors require specific hardware to enable them to talk to sensors in the surroundings, for example, short range ultra wide band radio (Shah et al., 2003). This may not suit pervasive environments as the cost of deploying special hardware on mobile data collectors like buses, cars might be expensive. Further, the random mobility-based approaches rely on single-hop communication with the requirement that sensors within the network will come in direct contact with the MULE. This may not be always practical. We present a comparative summary of the previously discussed approaches in Table 2.6 identifying the key merits and de-merits of each approach.

The previously presented literature on mobile node-based data collection has addressed the challenges introduced by mobility, namely: 1) mobile node arrival rate based on type of mobility i.e., controlled, predicted or random, 2) data success rate based on mobile node's velocity and 3) sensor-data collector discovery. The data collector approaches mostly focus on single hop data collection with less or no discussion on multi-hop data collection. One of the major contributions of this dissertation is the proposal to employ *k*-Nearest Neighbour queries to collect data from multi-hop sensor networks. In the next section, we discuss work related to *k*-Nearest Neighbour query processing in sensor networks.

Mobility Category	**Mobile Sensor**	**Mobile Sinks**	**Mobile Data Collectors**
Algorithms	Zebranet (Zhang et al., 2004), SWIM (Small et al., 2003), Cartel (2006)	Controlled Mobility (Kansal et al., 2004), Hybrid Mobility (Ren et al., 2006), SENMA(Venkitasubramaniam et al., 2004),	MULE (Shah et al., 2003) Predicted Mobility (Chakrabarti et al., 2003) Anastasi et al. (Anastasi et al., 2009b)
Communication Mode	Mostly Single-Hop. Some approaches incorporate Multi-hop	Single-Hop. Multi-Hop approach achieved using clustering	Single-Hop (Direct)
Mobility Type	Random	Controlled. Sink decides next location	Random / Predicted
Infrastructure Cost	High	High	Medium
Data Collection/ Delivery Delay	Medium - High	Low - Medium	High
Query Based	No	Partial/ No	No
Radio Type	Connection-Less	Connection-Less	Connection-Less[5]
Cost-Efficient Metrics	No	No	No
Sensor Type	Connection-Less	Connection-Less	Connection-Less
Merits	Suitable for certain applications where sensors need to be mobile. For example Zebra monitoring.	Reduces multi-hop communication, hence uniform energy depletion within the network. Employing clustering can leverage on multi-hop data collection Good energy savings compared to static approaches	Short range single hop communication. Considerable energy savings due to direct communication. MULE approach, a pioneering work

[5] Approaches based on connection-less radio techniques namely radio with broadcast channel capability requiring constant listening to the channel (E.g. Mica Mote).

		(up to 100% improvement in network lifetime)	in proposing three-tier data collection architecture
De-Metrics	High infrastructure cost. Requires special hardware to sustain mobility (except in cases where mobility is achieved by applications nature. E.g. Zebranet) Sink is static and hence requires the mobile sensor to arrive at the sink.	Sink require specialized hardware for mobility. Sink location needs to be updated within the network. Require complex scheduling approaches to achieve low delay and good data success rate. Clustering approach used for multi-hop data collection requires training and has drawbacks of classical clustering approaches.	MULE needs to be equipped with hardware to communicate with sensors. MULE's analytical model is restricted by two-dimensional grid-based sensor occupancy.

Table 2.6: Comparison of Mobile Node-Based Data Collection Approaches

2.4 Data Collection employing *k*-Nearest Neighbour Queries

k-Nearest Neighbour (*kNN*) queries have been explored as a solution for sensor data collection. *kNN* queries are a class of spatial queries technique traditionally employed in databases to retrieve spatially distributed data (Mouratidis et al., 2005a, Mouratidis et al., 2005b, Roussopoulos et al., 1995). A typical *kNN* query retrieves a list of *k* objects that are closest to a query point *Q*. For example, find the nearest set of restaurants from a given location. The query point is sometimes referred to as the *point-of-interest*. The use of *kNN* queries to determine proximity (closest object) requires some form of distance definition (Becker et al., 2005). This can be geometric coordinates (e.g. GPS) or symbolic coordinates (e.g. cell-Ids, wireless LAN positioning). In traditional databases, a centralised/distributed index has been used to efficiently process *kNN* queries. R-trees (Guttman, 1984) is one such dynamic indexing technique employed in spatial data search. R-tree is a height-balanced tree whose leaf nodes point to individual data objects. The idea behind R-tree is to generate a spatial index for a group of objects which are geographically related. This index is then used to perform fast spatial search. The R-tree is a centralised index maintenance technique that has the feature of updating/deleting index entries when object properties change. A search operation involves traversing the tree finding results that match the query. Roussopoulos et al. (1995) present a branch and bound search technique to process *kNN* queries over R-trees. An R-tree representation is presented in Figure 2.12. Figure 2.12 (left) is the R-tree for the corresponding minimum bounding rectangle (MBR) in Figure 2.12 (right). A MBR is a minimal rectangle that encompasses all child nodes grouped by their geographical location. In case of datasets in databases, the grouping metric can be based on Euclidian distance between data objects.

A sensor network is a classic example of spatially distributed data where *kNN* queries can be employed to improve data collection efficiency. *kNN* queries are highly relevant to sensor network applications (Winter et al., 2004) as they require instantaneous retrieval of data for queries like "get the temperature reading of 5 nearest sensor from Q". The use of *kNN* queries has been studied in sensor networks (Soheili et al., 2005, Winter et al., 2005, Wu et al., 2007, Yao et al., 2009, Yao et al., 2006, Demirbas et al., 2003). The *kNN* query processing in sensor networks can be broadly classified into *infrastructure-based* and infrastructure-less approaches. Infrastructure represents the topological information (indices) maintained by the sensors to respond to *kNN* queries.

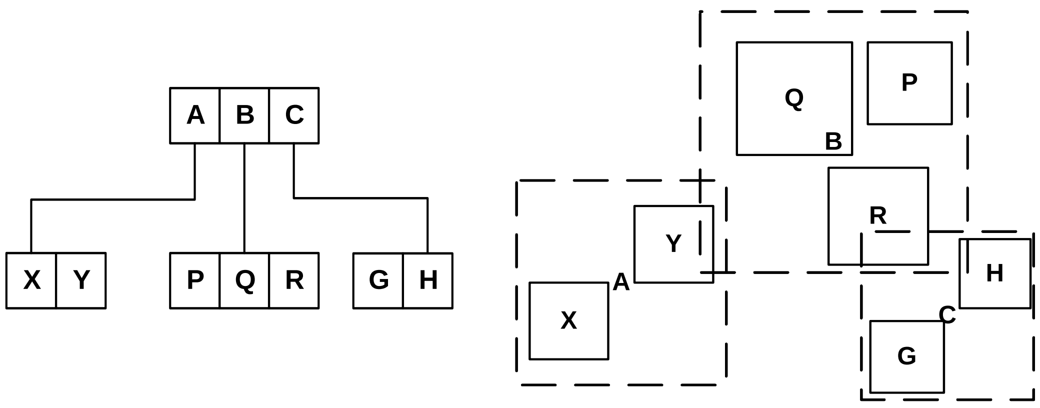


Figure 2.12: R-Tree representation

Murat et al., (2003) present *Peer-trees,* a peer-to-peer index structure to process *kNN* queries in sensor networks. Peer-tree is an energy-efficient de-centralised adaptation of R-trees. The peer-tree technique partitions the sensor network into rectangle-shaped clusters based on number of nodes. A split operation is employed to split nodes into multiple clusters if a given cluster has too many nodes. In peer-tree, the *kNN* query can originate from any point in the network moving towards the root. Since, partition information is available to cluster heads the query propagates to other clusters only when a result is not found within that cluster. This approach reduces network broadcasting. Soheili et al. (2005) propose *SPatial IndeX (SPIX)*, a distributed spatial index approach for processing *kNN* queries. SPIX like R-tree maintains index structures on each sensor node allowing it to respond to queries efficiently. The spatial index is created using flooding and further maintenance is performed by sensors periodically. The key energy consuming function of the index-based approaches is the overhead involved in creating and maintaining indices.

Yao et al. (2006, 2009) propose a technique to track an object closest to the query point Q using nearest neighbour queries in sensor networks. The sensor network is partitioned into a two-dimensional grid with each node within the grid sending sampled data to the grid index node - a node which is closest to the grid centre. The sampled data is the location of the tracked object which is sent to the grid index node only when the object's location changes. The search boundary is estimated by routing the query to each grid cell until at least one object is found. The distance between the query point and the grid cell in which at least one object is found becomes the search area (boundary). An expanded search is performed within the estimated search boundary to determine the object that is closest to the query point Q. The primary area of energy consumption with this approach is the overhead involved in the grid maintenance. Further, grid index nodes are assumed to communicate with one another during query computation. This might require communication involving nodes around the grid index node leading to non-uniform energy consumption.

Winter et al. (2005, 2004) propose partial-infrastructure-based and infrastructure-based *kNN* algorithms. The infrastructure-based algorithm is based on the Geo-Routing Tree (GRT) protocol. The GRT is similar to R-tree requiring spatial indices to process *kNN* queries. Each sensor node in the GRT maintains a minimum bounding rectangle (MBR) encompassing it and its child nodes. The MBR is computed using geographical proximity between sensors. The GRT algorithm works by pruning sensor nodes in MBR which are farther away from the query point Q. To prune the sensors that are outside the *kNN* region, it uses *MINDIST* (Roussopoulos et al., 1995) principle. The *MINDIST* is the distance between the query point P and the closest edge of the MBR. An illustration of the MBR was presented previously in Figure 2.12. The issue with the index-based approach in dense sensor networks is a large storage requirement to maintain index structures on individual nodes. It also has the drawback of high overheads involved in maintaining index structures when sensor nodes fail. To improve the performance of the GRT based *kNN* algorithm Winter et al. (2005) propose a partial-infrastructure-based *kNN* algorithm, namely, KBT. KBT uses Greedy Perimeter Stateless Routing (GPSR) protocol to route sensor information. The partial-infrastructure-based approach adapts well within dynamic environments, e.g. network with mobile sensor networks. The GPSR employs perimeter routing when a path to a specific node does not exist i.e. nodes that are not in direct contact are reached by forwarding the request along the perimeter of the sensor network. KBT employs some approaches including maximum hop distance (MHD), NeighbourClass and NeighbourClass2 to determine the *kNN* search boundary. The boundary estimation approaches use the information collected when the query travels from the sink to a node close to the query point Q. Further, restricted flooding and timers based on fixed TreeHeight are employed to collect data from the sensors. The TreeHeight determines the time the parent node needs to wait before its child nodes respond. The TreeHeight in simple terms is the maximum hop distance from the query point that encloses the set of *k* nearest neighbours. KBT uses static TreeHeight allocation for data collection. Simulation and evaluations results show that KBT performs better than GRT for dynamic sensor environments. Moreover, KBT exhibits better energy efficiency than its counterpart GRT approach. The use of fixed TreeHeight restricts KBT's operation.

The approaches discussed so far require some sort of network partition/infrastructure information to process *kNN* queries. Further, certain approaches require cluster formation and maintenance for successful creation/maintenance of indices. When a cluster head fails, a re-election algorithm is used to re-elect new cluster heads.

These requirements are energy consuming and introduce communication overheads. Adapting index-based approaches in sensor networks is expensive due to the amount of communication involved in maintaining indices. Moreover, index-based approaches are vulnerable to network topology changes. Hence, infrastructure-less sensor query processing approaches (Xu et al., 2006, Wu et al., 2007) have been explored to overcome index maintenance problems.

Xu et al. (2006) propose an infrastructure-less technique to process window queries in sensor networks. The window query is different from a *kNN* query as window queries focus on collecting data from a set of sensors that are within a query window. Though this approach does not investigate *kNN* query processing directly, it explores window query-based data collection which involves identifying sensors within a query window in an infrastructure-less sensor network. The query window represented as a rectangular region defines the data collection area. To this end, they propose an itinerary based data collection scheme called itinerary-based window query execution (IWQE). IWQE employs Geo-Routing protocol to route the query towards the query window. IWQE introduces the concept of *Query nodes* (Q-nodes) and *Data nodes* (D-nodes). The Q-node forwards the query to adjacent Q-node within the query window while the D-nodes respond with sensed data. By employing the itinerary-based forwarding among Q-nodes, network flooding is reduced improving the energy efficiency of the IWQE technique. Performance evaluations of IWQE protocol show improved energy efficiency over the infrastructure-based query processing protocols.

Wu et al. (2007) present an infrastructure-less itinerary-based *kNN* query processing algorithm, namely, DKINN. The DKINN approach is motivated by Xu et al. (2006) itinerary-based window query processing technique. DKINN propagates the *kNN* query from the sink to a node closest to the query point P. DKINN employs the concept of Q-nodes and D-nodes introduced previously to route the query from the source to a node adjacent to Q. While the query routes to the nearest node, it collects network information along the path. The *kNN* boundary is estimated by using the information collected during the query propagation phase. The estimated *kNN* search boundary is then searched to compute the *k* nearest neighbours. The performance evaluation of DKINN has been compared with the GRT approach proposed by Winter et al (2005). With assumptions that itinerary-based query propagation is feasible, DKINN seems to perform better than GRT approach with good savings in energy.

The DKINN and IWQE techniques depend on itinerary information to propagate query through the network. The itinerary creation and maintenance is feasible only if the sensor network is trained to forward data based on pre-computed trajectories (Niculescu et al., 2003). The trajectory provides the path the query needs to take, to reach the node adjacent to the query point Q. The trajectory computation results in the election of Q-nodes and D-nodes which are the basis for DKINN and IWQE execution. To pre-compute the trajectory, the sensors need to be equipped with specialized hardware (parallel array antennas) (Niculescu et al., 2003). Further, Q-nodes are similar to cluster heads and are responsible for forwarding the query to the next Q-node while collecting information from D-nodes within the neighbourhood. Query propagation happens only between Q-nodes resulting in non-uniform energy depletion of the sensor network. Finally, the requirement of specialized hardware to compute trajectory may not be feasible in every sensor network application. Hence, we conclude that window query processing technique IWQE and more specifically the *kNN* query processing technique DKINN may apply only to a specific class of sensors. Table 2.7 presents our comparative analysis of *kNN* query processing approaches.

2.4.1 Summary of Data Collection Techniques Employing *kNN* queries

The previous section presented the literature on *kNN* query-based data collection techniques in sensor networks. The plethora of research on *kNN* query processing depends on sensor network infrastructure knowledge to efficiently process the query. Moreover, the infrastructure-less approaches are tailored for a specific class of sensor network applications with special hardware requirements. They assume a classic sensor network model i.e. the sink acts as the query source and nodes within the network assume the role of co-ordinators managing query processing. Further, current *kNN* query-based data collection approaches assume a two dimensional sensor network. Our proposed approach discussed in depth in Chapter 4, investigates the feasibility and efficiency of a *kNN* query-based sensor data collection approach in a three dimensional sensor network using mobile data collectors that have no prior knowledge of the sensor network topology.

Approach	Sensor Mobility	Mobile Data Collectors	Cost Efficiency Metrics	Advantages	Limitations
Infrastructure-Based	NO	NO	NO	Better query execution efficiency due to the maintenance of indices Uses in-network aggregation for energy-efficient query processing	High cost in maintaining indices Does not adapt well for dynamic networks i.e. cases where sensor nodes fails/disappear Requires training phase to generate network index Sink based query origination
Infrastructure-less	YES	NO	NO	Adapts well for dynamic networks e.g. mobile sensors On-request computation of nearest neighbours Energy-efficient than infrastructure-based approaches	Current approaches (Wu et al., 2007, Xu et al., 2006) require pre-computed trajectory to compute routing paths. Query originates from the sink. A single node close to the point-of-interest needs to be elevated to successfully process the query.

Table 2.7: Analysis of *kNN* Query Processing Approaches in Sensor Networks

2.5 Context Modelling Approaches

One of the contributions in this thesis is the proposal of a dynamic situation modelling approach based on Context Spaces (Padovitz et al., 2004) theory. Context Spaces is a theoretical approach for modelling and reasoning about context, based on situations. In the following subsections, we present an overview of context-aware computing and explore existing context modelling approaches.

2.5.1 Context-Aware Computing: Overview

Mark Weiser's vision for 21st century computing devices (Weiser, 1999), laid the foundation for the pervasive era of computing - a new computing system paradigm. He coined the term *"ubiquitous computing"* with the vision of embedding user environments with communication and computing capabilities yet making them transparent. Research into various streams of pervasive computing has begun to look into various challenges concerning realisation of smart pervasive systems. These challenges range from architectural and technological challenges (Yoshimi, 2000, Pascoe et al., 1999) to security, privacy and social issues (Satyanarayanan, 2000, Bellotti et al., 1993). Of the various significant research areas investigated, context-aware computing was identified as a key ingredient in realising the pervasive computing vision (Satyanarayanan, 2002). In the context of pervasive computing, context-aware computing enables a system to adapt its behaviour dynamically in changing environments. The dictionary meaning of the term *context* refers to *"That which surrounds, and gives meaning to, something else*[6]*"*. The interest in context-aware computing has given rise to number of definitions for context (Brown et al., 1997, Abowd et al., 1999, Chalmers, 2002). A broader definition of context given by Dey and Abowd (Abowd et al., 1999) is

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

Chalmers (2002) provides another abstraction of context from a more user-based perspective as a "*circumstances relevant to the interaction between a user and their computing environments*". This definition fits into Dey and Abowd's (1999) generic description of context. In real-world situations, context can represent any information that enables a system to adapt its operation to achieve pre-defined system goals. For example, current temperature in a room is the contextual information that can be used by a heating, ventilating and air-conditioning (HVAC) system to adapt its performance to achieve ambient temperature settings. Other context includes location of user, location of device, current time, device resource e.g. battery level, signal strength, etc. Some example of context-aware applications within the domain of pervasive computing that have utilized

[6] Dictionary Reference of context: http://dictionary.reference.com/browse/context

contextual information to adapt system operations include Active Badges (Want et al., 1992), Cyberguide (Abowd et al., 1997), Smart Kindergarten (Srivastava et al., 2001).

Active badge (Want et al., 1992) is a location system designed to compute location of people within an office environment. Specially designed badges worn by staff in an office are responsible for transmitting information regarding location to a centralised server. Cyberguide (Abowd et al., 1997) is another example of a system that captures user's physical location and orientation as contextual information to provide appropriate information on user's current location. An envisioned application is a museum tour guide that provides users with information on exhibits based on their location and direction in which they are facing. Smart Kindergarten (Srivastava et al., 2001) envisions an individualised learning environment to enhance education in a smart kindergarten. The system comprises sensor-enhanced toys that interact wirelessly with other toys and a back-end middleware service that offers data management services. The contextual information used in this application includes object locations, children location and relative location information i.e. child A is near toy B. Location management is a key to context-aware applications. Becker et al. (Becker et al., 2005) presents location models for pervasive computing systems for different types of location-based queries, namely, range queries (e.g. object within a specific area), nearest neighbour queries (e.g. find the closest printer), positions queries (e.g. location of buildings, bus stops, etc.) and navigational queries (e.g. route from one location to another).

2.5.2 Situation Reasoning and Context Modelling

A situation is generally defined as "*the combination of circumstances at a given moment*[7]". We perceive situation as a combination of contextual information that best matches its real-world counterpart. Context modelling is the process of representing situations in a system that best symbolises its real-world counterpart. Reasoning allows inferring situation occurrence defined in the context model, based on contextual information sensed from real-world environments. Hence, situations are high level

[7] Dictionary Reference of situation: http://dictionary.reference.com/browse/situation

abstractions of context inferred from single/multiple sources based on rules or reasoning techniques.

Akman et al. (1997) present a situation theory based context model. The situation theory is a mathematical theory of information that allows situations to be represented as discrete items of information (infons). Two types of infons represent situations and their relations, namely, factual infons, stating the facts and conditional infons that have rules relating facts. Goslar et al. (2004) considers contextual representation of situations based on topic maps (Topicmap, 2010). Topic maps allow creation of relationships between objects located inside and outside the system. Dey (2001) discusses the importance of understanding context by defining situations as an abstraction that describe the states of relevant entities. The object oriented *Context Composition* (CoCo) model (Buchholz et al., 2004) uses classes, factory-nodes, operator-nodes and scales (data formats) to represent context and situations. Padovitz et al. (2004, Padovitz, 2006) defines situation as meta-level description of real-world situations that is inferred from available context. Context in this case is a fusion of raw data collected from low-level sensors. Context gives relation and meaning to raw data streaming from sensor sources.

Situation-based reasoning approaches have to take into consideration varying degrees of uncertainty. Uncertainty in this case deals with the inherent gap between representations of context in real-world against context perceived by the system. A context model is a generic way of representing information that can be used to reason about context As mentioned before, one of the research challenges addressed in this thesis is dynamic situation composition using Context Spaces theory (Padovitz et al., 2004, Padovitz, 2006, Padovitz et al., 2005). Hence, in this section, we provide a review of context modelling techniques based on their underlying principle and inference technique. We justify our choice of Context Spaces over other context modelling approaches, identifying scope for improvement in Context Spaces which is later addressed in this thesis.

Bayesian reasoning is a statistical inference technique in which known evidence is used to compute a degree of belief (probability) of an underlying hypothesis. (Bayes et al., 1763). In (Fox et al., 2003), a Bayesian technique is used for location estimation using inaccurate sensor reading. Castro et al. (2000) present another Bayesian approach to compute location of indoor devices using signal-to-noise ratio (SNR) between devices and base stations. Dempster-Shafer is considered a generalised Bayesian statistical theory (Wu et al., 2003). It allows computation of support for a proposition (e.g. this is a meeting) using an upper and lower bound of probabilities defined as confidence interval. Jian et al. (2007) present a Dempster-Shafer evidence theory-based context-aware architecture. The proposed architecture uses planes to group sensors that produce similar contextual information. A Dempster-Shafer approach is preferred over Bayesian reasoning due to the ability of Dempster-Shafer to account for general uncertainty. Further, aggregators and registrars are used to process low-level sensor information and present it to context-aware applications that require adaptation advice. Wu et al. (2002) present a weighted Dempster-Shafer evidence combination rule to deal with context from multiple sensor sources. The idea of using weights introduces the concept of sensor-reliability. The weights are computed from historical observations on correctness of sensor data. This effort is further extended in (Wu et al., 2003) by introducing dynamic weights which is adapted continuously based on sensor performance using Kalman filtering.

Fuzzy based approaches have been employed for reasoning about situational context. Mäntyjärvi et al. (2002) propose a fuzzy based approach that adapts context aware applications based on multiple fuzzy contexts. The use of fuzzy logic (Mendel, 1995) allows context-aware applications to adapt to state transitions e.g. changing from walking state to running state. Byun et al. (2003) extends fuzzy approach by using fuzzy decision trees and historical context to handle uncertainty. The use of historical context makes the system proactive allowing it to predict user actions intuitively. More recently, Haghighi et al. (2009) propose the incorporation of fuzzy logic in Context Spaces (Padovitz, 2006). Their approach aims to identify delta level changes in situations useful in medical applications.

Ontology based reasoning approaches have also been explored in the literature (Ko et al., 2008, Gu et al., 2004, Chen et al., 2003). Ontology is a formal representation of concepts and its relationships within a particular domain (Chen et al., 2003). Ontology provides a way to precisely describe domain knowledge and situation information (Ko et al., 2008). Concepts (information) are represented in ontology using ontology languages such as OIL[8] (Ontology Interface Layer) or DAML[9] + OIL (DARPA Agent Markup Language). A further standardised language used to represent ontology is Web Ontology Language (OWL) which is based on DAML+OIL (Chen et al., 2003, Gu et al., 2004). Chen et al. (2003) propose an ontology-based Context Broker Architecture (CoBra) which employs agents to acquire, reason and share context. The ontology is divided into four categories, namely, people, agents, places and events. The ontology is also used to describe the relationship between categories. The architecture is domain specific and reasoning deals with detecting and solving inconsistencies between facts. Ranganathan et al. (2004) uses predicates to represent context with associated confidences. DAML+OIL are used to specify the predicate structure and semantics. The context architecture *Gaia* (Ranganathan et al., 2004), facilitates reasoning using various mechanisms including fuzzy logic, probabilistic and Bayesian networks.

Parallel to context modelling, research in context-aware computing investigates context-aware frameworks that ease the task of developing context-aware applications. Dey et al. present (2001) Context ToolKit, a framework that allows creation of interfaces between devices and software entities that provide contextual information. Five context abstractions, namely, Widgets, Interpreters, Aggregators, Services and Discoverers can be used by context-aware application developers to prototype applications. The widgets acquire data from sensors, interpreter interprets that into high level information and aggregator collects related context for a specific entity. Services execute behaviours and discoverers maintain a list of services available to the application. Applications use discoverers to find specific components or a set of component that match certain criteria. This work is extended by Dey and Mankoff (2005) to handle ambiguous context. They use the term *mediation* to refer to the process involved in solving ambiguous context involving human computer interaction. In a shell, mediation allows the system to 1) identify ambiguous context, 2) provide/obtain feedback to/from the application/user on ambiguous context and 3) store feedback and re-use for final context interpretation. A plethora of research that facilitates context-aware application development can be found

[8] www.ontoknowledge.org/oil/
[9] www.daml.org/

in the literature (Khedr et al., 2005, Capra et al., 2001, Glassey et al., 2003, Bagci et al., 2004).

Review of context modelling approaches presented previously show that current approaches are application specific, making it difficult to provide a generic approach for context modelling under changing situations. Bayesian approach has the limitation of knowing prior probabilities which in many cases may not be feasible. The Fuzzy logic-based approaches can determine situation states in terms of percentages which may not always be useful. For example, concluding that a person has 60% cold and 40% flu may not help in many applications. The ontology-based approaches have been used to represent context within specific domains. Though ontology-based approaches provide a deep understanding of specific domains it may not be feasible to use generic ontology to represent a host of domains i.e. a generic representation.

These shortfalls have been addressed by Context Spaces (Padovitz et al., 2004, Padovitz, 2006). Context spaces, is a spatial metaphor-based context modelling approach. It uses situations to reflect real-world situations. A situation is represented as a collection of context attributes. It uses Multi Attribute Utility Theory (MAUT) based sensor data fusion algorithm (ConSpaF) to fuse data from multiple sensor sources. It provides powerful reasoning technique to infer situations based on current contextual information. Context Spaces Algebra based on Context Spaces model is used to manipulate relations between situations. The reasoning approach of Context Spaces takes uncertainties in sensed values and significance of context attributes across different application domains to compute a confidence measure. The strength of Context Space is its ability to generically represent context using spatial metaphors of state and space. Moreover, Context Spaces model is developed on the basis of sensor originated data. In this thesis we propose techniques for energy-efficient sensor data collection. We then look to extend our approach by exploring ways in which collected sensor data can be used to dynamically model situations aiding the reasoning process. These strengths of Context Spaces supported our choice to use Context Spaces as our basic context model for further investigation.

2.6 Summary

Pervasive computing applications are highly dependent on real-world data for efficient decision making. These real-world data are available from sensors embedded in pervasive environments. In this section, we identified sensor networks as a key technology that drives current pervasive computing applications. This opens up the area of research that addresses the challenges involved in energy-efficient data collection. The energy and processing limitations of sensor nodes justifies the need for energy-efficient data collection strategies. In this chapter, we have presented a background on sensor networks identifying the current state-of-art technologies. A discussion on current sensor node platforms from an energy perspective has been presented identifying the major components that drain sensor energy. We identified communication as one of the primary energy consuming operations making it a key design consideration for data collection protocols.

We reviewed current data collection approaches based on a broad classification of static node-based and mobile-node based techniques. The static node-based approaches have been identified to cause sensor hot-spots i.e. sensor nodes near the sink deplete energy quickly. Introducing mobility leverages the fact that a mobile node can move close to sensor node's location to collect data. The literature has reviewed mobile data collection approaches by broadly classifying them into sink, sensor and data collector mobility. The mobile sink approaches are energy rewarding but are more feasible in sensor networks that have specifically designed infrastructures. Mobile sensor nodes introduce the limitation of specially designed hardware for mobility. Hence, in this dissertation, we explore a mobile data collector-based approach. Our mobile data mule approach complements Data Mule (Shah et al., 2003) approach and also addresses the shortfalls of Data Mule approach elaborated in the literature. Further, we explore a *kNN* based data collection approach using mobile sensor data collectors.

Mobile data collector-based approaches to sensor data collection using *kNN* are in their infancy. Further, system framework for data collection using mobile data collector has not been addressed elaborately in the literature. Our objective is to investigate *kNN* based mobile data collection as a cost-efficient alternative for sensor data collection.

More specifically, we note that current approaches in data collection are missing the following desirable characteristics:

- A sensor data collection system framework with energy-efficient sensor data collection algorithms applicable in real-world pervasive environments.
- A data collection protocol for energy-efficient multi-hop sensor data collection using mobile data mules that have the ability to compute data collection decisions on-the-run.

Finally, we reviewed context modelling and situation-based reasoning approaches as a high level application of collected sensor data. We identified Context Spaces (Padovitz, 2006) as our choice of context model due to its generic representation of context using spatial metaphors. We note that Context Spaces model can be improved further by incorporating:

- A dynamic situation model that can be composed on-the-fly with capability to incorporate newly discovered context (using the above discussed data collection techniques)
- Incorporating additional sensor quality information leveraging on existing sensor measurement inaccuracies.

3

sGaRuDa[10]: Sensor Data Collection Using Heterogeneous Mobile Devices

3.1 Introduction

Evolution of pervasive computing paradigm has given rise to the integration of intelligent computing devices into physical spaces turning them into smart spaces/environments (Lewis, 2004). Sensors are a key element for continual existence of smart spaces. We presented a literature survey in Chapter 2 on current approaches that addressed challenges involved in sensor data collection. In this thesis, we will be employing a mobility-based sensor data collection approach. We identified the key benefits in using mobile data collectors for sensor data collection in Chapter 2. The introduction of mobility opens up new dimensions in approaching the sensor data collection. We also identified gaps in existing approaches and room for improved data collection. More specifically we identified the following in current approaches:

(1) Focus on connection-less sensor network, i.e. sensors equipped with radio hardware that have broadcasting capabilities. Research in the area of using mobility to collect and deliver sensor data from Bluetooth-based sensor nodes is still immature

(2) Employ mobile sinks for data collection which require frequent location updates from the sink to the sensors. This problem has diverged into mobile

[10] sGaRuDa is the name of the proposed system. It signifies sensor data carrier

sink scheduling problem (Somasundara et al., 2004, Gu et al., 2006). Though mobile element scheduling handles energy-efficient sink mobility, the approach is specific to sensor network applications where mobile sinks can be controlled.

(3) Absence of a software-based system framework for sensor data collection that can be implemented on today's heterogeneous [11]mobile devices.

(4) Focus on data collection protocols for broadcast-based networks with sufficient support lacking for connection-oriented networks (e.g. Bluetooth-based sensor networks).

Summing up, current data collection approaches employing mobility lack support for connection-oriented sensor networks (Bluetooth-based) as the operation of these networks differ significantly from connection-less networks (refer 2.2.4) . Our choice to investigate Bluetooth-based sensor networks is driven by the ubiquitous acceptance of Bluetooth technology available on most mobile device platforms. Moreover, studies of the feasibility of employing Bluetooth-based sensor networks (Nachman et al., 2005, Lundberg et al., 2005, Leopold et al., 2003) have shown that with proper use of Bluetooth low-power mode (Lundberg et al., 2005), the lifetime of Bluetooth-based sensors can be considerably extended. The widespread adaptation of Bluetooth and research initiatives in the area of Bluetooth-based sensor networks (BSN) open up a new dimension of sensor network applications that can be deployed within pervasive environments with ease.

In this chapter, we propose sGaRuDa, a system framework that can be implemented on multitude of *day-to-day* computing devices e.g. smart phones, Personal Digital Assistants (PDA), laptops, etc, giving them the capability to discover, collect and deliver sensor data on-the-fly. Summing up our proposed approach:

We propose, investigate, implement and evaluate a system framework (sGaRuDa) that can run on a multitude of current generation heterogeneous mobile device platforms that require no additional hardware to perform cost-efficient sensor data collection.

[11] Heterogeneity represents the plethora of different mobile device platforms (hardware and software)

The term *"cost-efficient"* is used to define the overall data collection cost which includes both time (processing efficiency) and energy parameters. We use the term *"heterogeneous"* to represent the class of different mobile device platforms. A good sensor data collection system framework needs to have the following desired characteristics to achieve cost-efficient sensor data collection:

Dynamic Device Discovery: The mobile devices in our proposed model are current day computing devices. The Bluetooth discovery protocol differs from broadcast-based sensors. Hence, it is important to analyse and efficiently use the Bluetooth discovery protocol to increase system efficiency.

Data Collection/Delivery Protocol: The use of Bluetooth-based network puts forward the requirement to establish a connection before data can be transferred. The connection process in Bluetooth is sometimes time consuming, hence requiring protocols that can efficiently and effectively use the Bluetooth baseband radio. Further, data delivery protocols address delivering sensor data to the sink.

Intelligence/Context: We defined the term "context" in the literature as *"That which surrounds, and gives meaning to, something else*[12]*"*. Context within our framework represents sensor and mobile device parameters that are used to arrive at data collection decision. For example, sensor location, residual energy, amount of data to be transmitted represent sensor context while mobile device location, mobile device velocity (state: static/moving), mobile device capabilities (communication) represent mobile device context.

The rest of the chapter is organised as follows. Section 3.2 presents an overview of our proposed system. Section 3.3 elaborates on the system components presenting details of algorithms proposed for device discovery, data collection and delivery. Section 3.4 presents examples of real-world application where the proposed approach can be employed. The chapter concludes with a summary presented in section 3.5. The system framework and the corresponding data collection and sensor adaptation algorithms presented in this chapter are extended versions from the following published papers

[12] Dictionary.com meaning, http://dictionary.reference.com/browse/context

(Jayaraman et al., 2007, Jayaraman et al., 2008a, Jayaraman et al., 2008c, Jayaraman et al., 2010b).

3.2 sGaRuDa: System Architecture Overview

A recent article by Wodajo (2010) on application of the new Bluetooth specification v4.0 in health care sensor devices that require low-powered operations is a classic real-world example of the widespread adoption of Bluetooth in pervasive environments. The application discussed in (Wodajo, 2010) requires sensors fitted on patients to monitor and deliver data over Bluetooth to fixed access points. The low cost of Bluetooth hardware (Warneke et al., 2002) and increasing applications have led to a new class of disposable wireless sensors requiring new approaches to collect and deliver data. Our proposed data collection approach is illustrated in Figure 3.1. The sGaRuDa system addresses challenges related to collecting data from a connection-oriented sensor network with some similarity and differences from the Data Mule (Shah et al., 2003) architecture. The architecture consists of three layers:

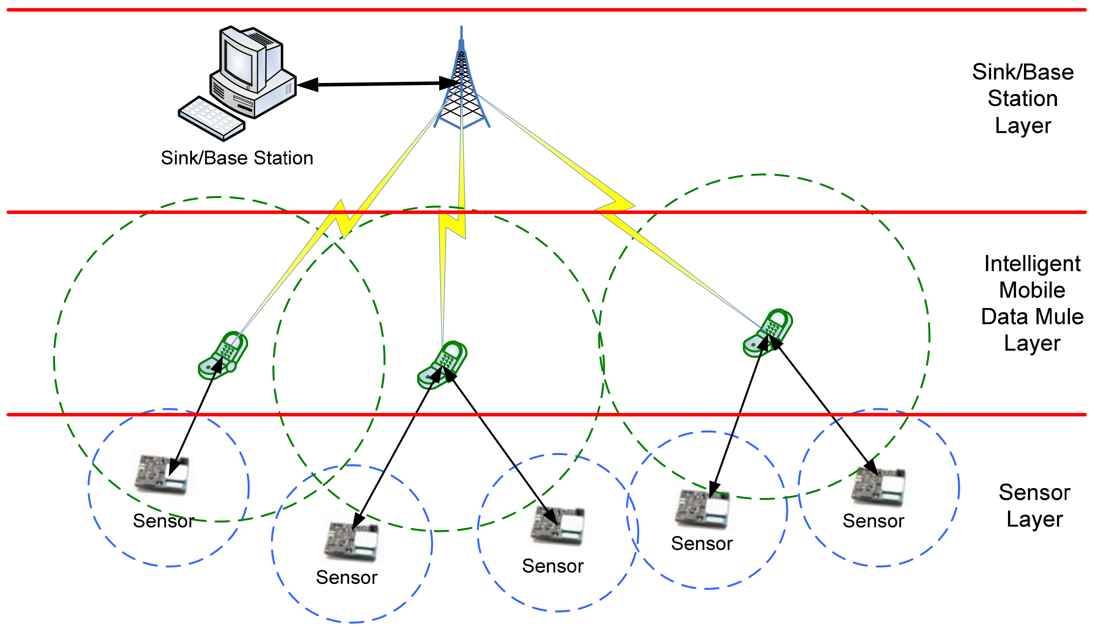


Figure 3.1: Overview of Proposed Data Collection Approach

Sensor layer: The sensor layer comprises of heterogeneous sensors i.e. sensors with different sensing capabilities from various manufactures. The sensors in this layer mostly share the common sensor characteristics of low-powered operation but might differ in their hardware specification. For example, the layer might comprise a combination of Mulle (EISLab, 2010), BTnode (ETH-Zurich, 2007) or Intel mote (Nachman et al., 2005). The sensors in this layer primarily use Bluetooth for communication while they might be equipped with alternative radios.

Intelligent Mobile Data Mule Layer: This layer comprises of mobile data mules. The mobile data mule represents the class of *day-to-day* mobile devices that are widely available in pervasive computing environments. We term these devices "*intelligent*" since they have the capability to compute data collection decisions based on information available within the environment. The data mule is a powerful energy-rich device with recharging capability.

Sink/Base Station Layer: This layer comprises sink/base station responsible for further analysis of sensor data. We use the term *base station* in parallel with the term *sink* since, in the proposed architecture, when the data collector is a mobile phone, the base station the mobile device is connected to performs the operation of a sink. The sink/base station may or may not have complete knowledge about the sensor network architecture but learns it over time. For example, let us consider the case of the disposable sensor for medical purposes. When replacing the old sensor with a new sensor, sensor information changes. This change is automatically propagated to the sink/base station when data from new sensor is delivered by the data mule.

The communication network deals with communications between sensors, mobile data mules and base stations/sinks. The communication between the sensor and mobile data mule in our architecture is primarily Bluetooth but our approach can be extended to accommodate other radio technologies e.g. Zigbee (2009b). The communication between the mobile data mule and the sink/base station can use any existing communication infrastructure, namely, GSM/GPRS, UMTS, Wi-Fi, WiMAX etc. The choice of communication depends on communication technology available on the data mule.

The smart phone in Figure 3.1 represents the plethora of mobile devices that have the capability to act as intelligent mobile data mules. These devices in real-world may include laptops, PDAs, cars equipped with Bluetooth, etc. We assume that data mules are not controlled. But, in situations where the mobile data mule can be controlled, the architecture provides flexibility to implement additional modules to take advantage of controlled mobility.

3.3 sGaRuDa: System Overview

Classic data collection architectures employing Bluetooth (Nachman et al., 2005, Handy, 2004) depend on piconets and scatternets. The use of piconets and scatternets (refer 2.2.4) enables multi-hop connection-oriented communication without the need of a Bluetooth access point. This approach has certain drawbacks: 1) slave nodes need to be constantly synchronised (maintain active connection) with the master, 2) dynamic route generation requires frequent inquiry and paging which consumes energy and 3) in disconnected networks it may not be feasible to install multiple Bluetooth access points within the network. This problem worsens when the Bluetooth radio used by the sensors is Class B (10 Meters).

The energy consumption of the connection-oriented (master/slave) operation is validated by experimental results presented by Nachman et al. (2005). Nachman et al. (2005) concludes that lifetime of the sensor node is reduced to a few days when working under a connected network scenario. We introduce mobility-based data collection strategy to overcome these drawbacks leveraging on mobility available within the environment.

3.3.1 sGaRuDa: System Framework Black Box

Figure 3.2 presents an overview of the proposed system framework. The main operations involved are discovery, data collection and delivery. The sink is a simple representation of a data store which can be extended based on application needs. For example, the sink can be a data centre that provides information to a community of users depending on user/application requirements. The proposed system can work in disconnected mode i.e. the sensor discovery and data collection does not require the ubiquitous presence of the centralised sink. But, the availability of the sink can be leveraged upon to increase the data collection efficiency. In a disconnect mode, sensor discovery is done locally by the mobile device which is later synchronised with the sink whenever sink connectivity becomes available.

The mobile data mule can employ a range of communication technologies including Wi-Fi, WiMAX, GSM, UMTS, etc to communicate with the sink. We illustrate the discovery process by a short arrow labelled *Discover* due to the connection-oriented operation of Bluetooth-based sensor network (BSN) presented in section 2.2.4, i.e. device discovery does not establish a connection.

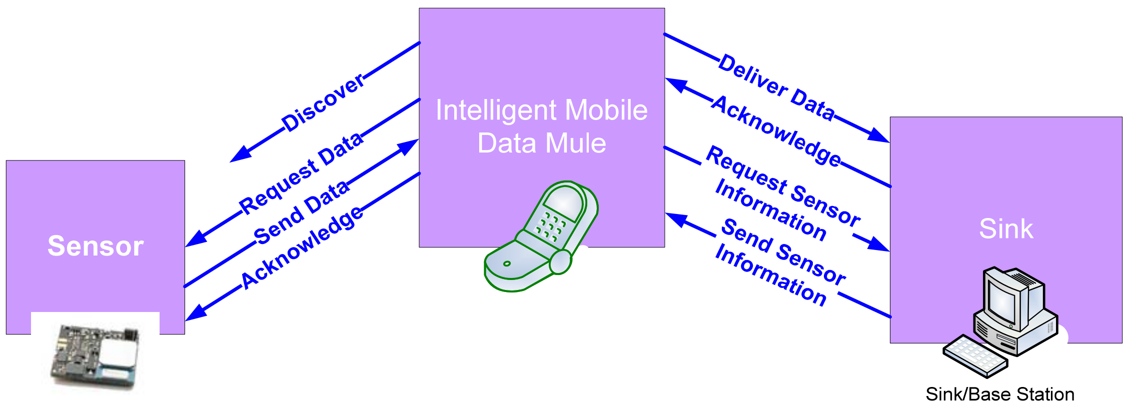


Figure 3.2: System Black Framework Overview

3.4 sGaRuDa: Mobile Data Mule System Framework

Figure 3.3 illustrates the proposed system framework. The system is divided into two platforms Mobile Data Mule platform and Data-Collection platform. The Data-Collection platform is device independent, allowing it to be ported across any mobile device. The Mobile Data Mule platform is device dependent hence taking advantage of specific mobile device capabilities. For example, if the mobile computing device is a mobile robot the device dependent implementation might include a robot control module.

3.4.1 Mobile Data Mule Platform

The Mobile Data Mule Platform comprises Location Manager, Communication Manager and Profile Manager. The mobile data mule platform acts as an interface exposing mobile platforms capabilities to the Data-Collection platform. These components are developed based on mobile device capabilities. The Mobile Data Mule platform facilitates easy integration of additional device-specific plug-ins. The components of the Mobile Data Mule Platform are discussed below:

***Location Manager*:** The location manager provides an interface to the mobile data mule's location subsystem. The location subsystem in its simplest form can be a GPS or cellular-based (Cell ID) that provides the Data-Collection platform with mobile data mule's path. The capability allows interfacing with other external location providing services e.g. real time location service. The interface also determines what sort of location information will be available to the Data-Collection platform.

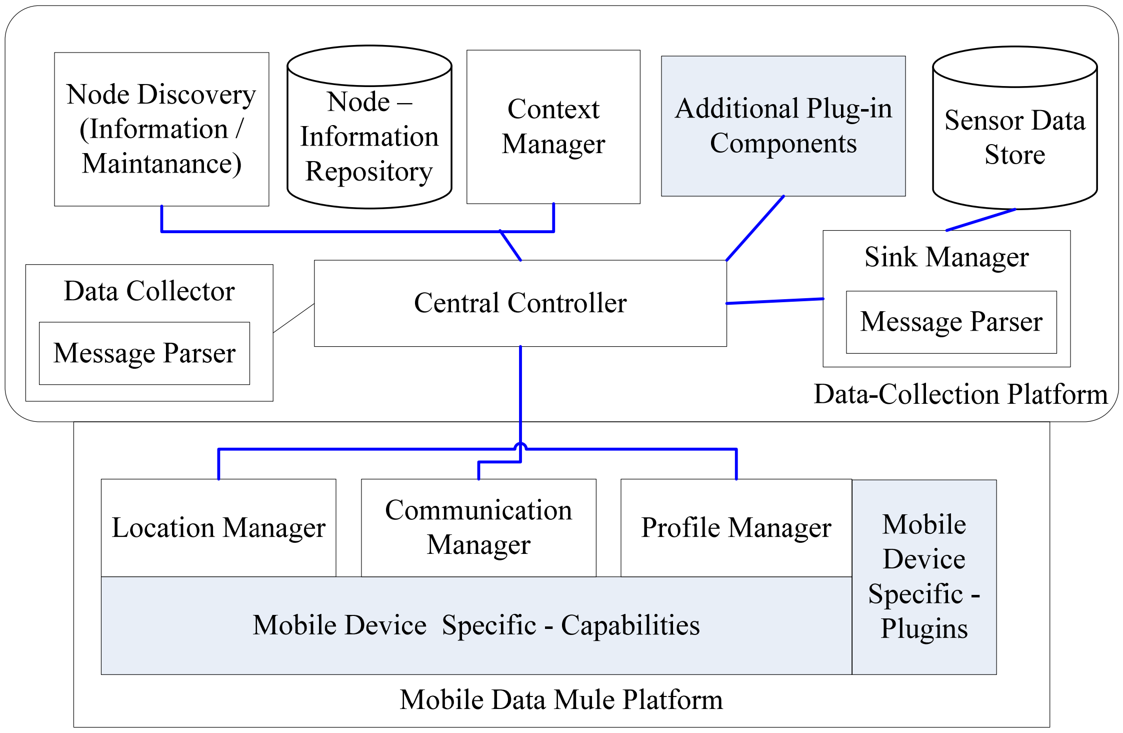

Node Discovery (Information / Maintanance)
Node – Information Repository
Context Manager
Additional Plug-in Components
Sensor Data Store
Sink Manager
Message Parser
Data Collector
Message Parser
Central Controller
Data-Collection Platform
Location Manager
Communication Manager
Profile Manager
Mobile Device Specific - Plugins
Mobile Device Specific - Capabilities
Mobile Data Mule Platform

Figure 3.3: Data Collection System Framework

***Communication Manager*:** The communication manager is an interface to the mobile device's communication hardware. The minimum communication facility that is available on the data mule is Bluetooth allowing it to communicate with underlying sensors. The data mule can be equipped with other communication technologies like Wi-Fi, UMTS, GSM, etc. The communication capability of the data mule determines the data delivery latency. For example, if the data mule is equipped only with Bluetooth and Wi-Fi capability it has to wait till it is within a Wi-Fi network to transmit the collected data to the sink/base station. This is acceptable in cases where data collection latency is acceptable. In applications where this is not acceptable the data mule will not participate in data collection.

Profile Manager: Our approach is to use *day-to-day* mobile devices for data collection. The priority of the mobile devices is to provide the user with primarily device functionality. Sensor data collection is performed only when the device is idle or has the resources available for data collection. The profile manager provides the Data-Collection platform with information of the data mule's availability. This availability for example may be resource oriented i.e. below a certain battery threshold the data mule will not participate in sensor data collection. The system provides capabilities to define additional parameters.

Mobile Device-Specific Plug-in: Mobile device-specific plug-ins is used to expose specific mobile device capabilities to the Data-Collection platform. This feature introduces flexibility in the proposed system framework. For example, the mobile device platform for mobile robot platform is a robot movement controller module. This module may be used by the Data-Collection platform to adapt the robot's speed to increase data collection efficiency. This of course will require additional modules to be integrated into the Data-Collection platform.

3.4.2 Data-Collection Platform

The Data-Collection platform is the core component of the proposed system framework. It performs three primary functions, namely: 1) sensor node discovery, 2) sensor data collection and 3) sensor data delivery. The three main modules *Node Discovery*, *Data Collector* and *Sink Manager* are responsible for handling one of the three primary functions. The *Central Controller* module coordinates between the primary modules and is responsible for computing data collection decisions. A detailed discussion of each function module is presented in the following sections.

Node Discovery/Management: The node discovery and management module performs handles sensor node discovery and management. The discovery process is used to identify sensor nodes within the vicinity of the mobile data mule. For example, in a Bluetooth-based sensor network the mobile data mules use Bluetooth discovery technique. The node repository is responsible to store sensor node information. Newly discovered sensor nodes are added to the repository. The node repository periodically synchronises sensor node information with the sink.

Data Collector: The data collector module is responsible for establishing connection with the sensor and collection data from the sensor. The connection is established using information collected during the node discovery phase. The *Data Collector* employs a multi-part data collection algorithm. This algorithm facilitates collection of sensor data by independent multiple mobile data mules.

Sink Manager: The sink manager is responsible for buffering and delivering the collected data. The buffering operation simply stores collected data from the sensor in temporary storage before offloading the same to the sink. It employs techniques to identify the communication medium that needs to be employed to deliver data efficiently.

Central Controller and Context Manager: The central controller co-ordinates the operations of the above modules. It uses the context manager to store context i.e. information relating to the mobile data mule. For example, current location (updated as data mule moves), trajectory (path/direction of the mobile data mule), communication capabilities, resource availability (battery level). The central controller manages the node discovery and the data collection, storage and delivery.

3.4.3 Node Discovery/Management

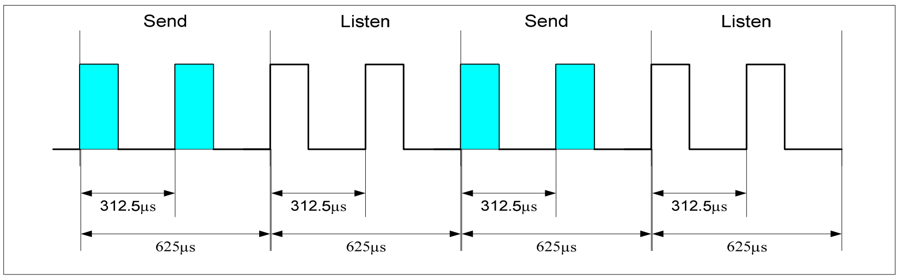
The *node discovery module* handles node discovery and management. The discovery process involves periodic discovery of sensor nodes within the environment. It comprises a node-information repository that is used to maintain a local copy of discovered sensor information. This information includes sensor id/name, location, duty cycle, last successful data collection time, sensor residual energy during last successful data collection cycle and sensor Bluetooth address. The node discovery function

Bluetooth-based sensor network differs from broadcast-based sensor networks. Hence, in the next subsection we analyse the working of the Bluetooth discovery protocol.

3.4.3.1 Bluetooth Discovery: An Analysis

A Bluetooth radio has three primary states, namely: *STANDBY*, *CONNECTION, PARK*; and seven sub-states, namely: *page, page scan, inquiry, inquiry scan, master response, slave response* and *inquiry response* (BluetoothSIG, 2010d). The inquiry and the inquiry scan sub-states belong to the device discovery phase. We present an analysis of the inquiry and inquiry scan states as this determines the Bluetooth discovery time.

The device that wants to be discovered, namely, the sensor enters an inquiry scan sub-state. The device performing discovery, in this instance, the mobile data mule enters the inquire sub-state. The sensor acts as the slave while the mobile data mule (inquiring node) assumes the role of the master. The slave node when in the inquiry scan sub-state listens repeatedly for an inquiring node (master). The master node in inquire sub-state sends periodic inquiry messages on 32 (dedicated) of the 79 frequencies. This choice of frequencies and hopping sequence is determined using the General Inquiry Access Code (GIAC) address (BluetoothSIG, 2010d). In each send operation, inquiry messages are transmitted in two consecutive time slots of 312.5µs each. The same operation is performed during the listen phase resulting in one send/listen time slot lasting for 625µs. The inquiry process is illustrated in Figure 3.4.



.

Figure 3.4: Bluetooth Inquiry Process

The 32 frequencies used to broadcast the inquiry message are divided into two sets of 16 frequencies, namely, train A and train B. The time taken to send inquiry messages across the 16 frequencies (one train) is 10ms. The Bluetooth specification suggests each train be repeated at least 256 times (BluetoothSIG, 2010d). Hence, the inquiry on a single train lasts for 2.56 seconds. The sensor node (slave) in the *inquiry scan* sub-state performs two types of scans, namely, standard and interlaced. In standard scan, the slave node scans the channel for one time interval of 11.25ms. In interlaced scan, two back-to-back scans of 11.25ms time slots is performed. The inquiry scan is performed over the 32 dedicated frequencies. The 11.25ms inquiry scan window used by the slave node increases the chance of it receiving the inquiry message from the master node whose window lasts 10ms. After an inquiry scan, the slave node enters a sleep mode before scanning the channel again. This value T_{scan} according to the specification can be less or equal to 2.56s. The master slave inquiry operation is depicted in Figure 3.5.

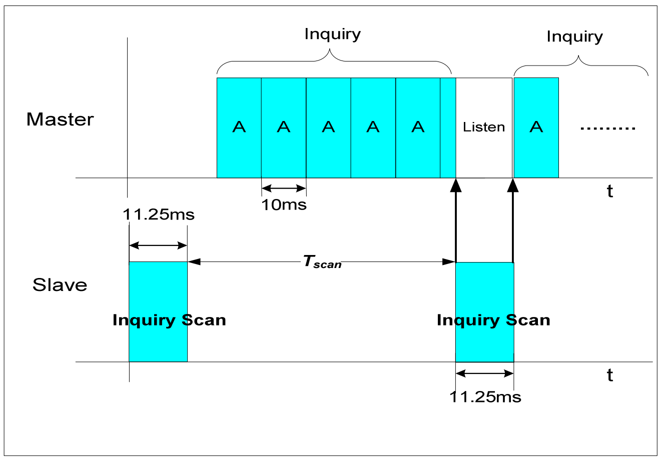


Figure 3.5: Overlapping inquiry time slots of Master and Slave in a Bluetooth inquiry scan

The master sends inquiry messages over consecutive train A's via frequencies each lasting 10ms. The slave scanning the channel every T_{scan} seconds on receiving the inquiry packet responds to the master within that time slot. The Bluetooth discovery operation involving the mobile data mule (master) and the sensor (slave) is illustrated in Figure 3.6. The sensor responds to the mobile data mule after successful reception of inquiry message. The operation presented in Figure 3.5 illustrates the timeslots overlap during the inquiry process while Figure 3.6 illustrates the detailed communication that takes place between the sensor and the mobile data mule within one time slot. The Frequency Hopping Synchronization (FHS) packet shown in the figure is a special packet (BluetoothSIG, 2010d) containing Bluetooth device address, page scan mode, clock information (used for synchronisation), etc.

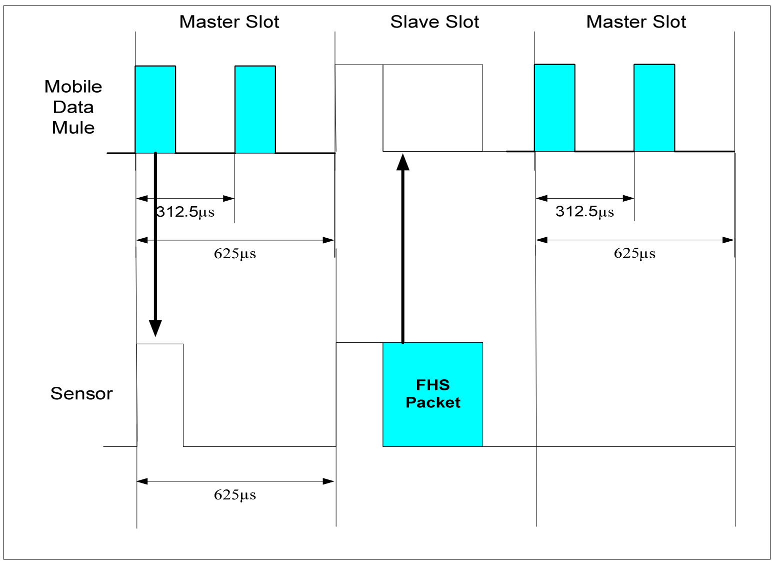


Figure 3.6: Bluetooth Discovery Process

3.4.3.2 Bluetooth Sensor Discovery: Proposed Technique

The Bluetooth discovery process (inquiry procedure) lasts 10.24 seconds. This delay is acceptable in applications where Bluetooth is used as a cable replacement technology. Within the scope of sensor network operations more specifically within the proposed mobile data mule-based data collection technique, the inquiry scan operation is too long. From the previously presented analysis of the Bluetooth discovery protocol, we have identified the primary reasons for the long inquiry operation. They are: 1) the sensor nodes (slaves) need to back-off T_{scan} seconds before it can perform another inquiry scan and 2) the Bluetooth specification (BluetoothSIG, 2010d) indicates that for a successful discovery, the inquiry process needs to be performed over 4 alternating trains which adds up to 10.24 seconds. Hence, we propose to use the following two techniques to reduce the inquiry time namely:

1) Use interlaced scan with a T_{scan} interval of 0 seconds. This allows the sensor to continuously scan the channel for short independent durations. The use of 0 second back-off increases the probability of a successful inquiry by the mobile data mule

2) Reducing the inquiry time interval. The recommendation of 10.24 second inquiry interval includes a back-off interval (T_{scan}) of 2.56 seconds. By

reducing the back-off interval to 0 seconds, the inquiry can be performed in 2.56 seconds i.e. 4 times less than the normal inquiry interval. We validate our proposal by experimental evaluations presented in Chapter 7.

Bluetooth discovery process uses the inquiry scan to collect sensor information that is further used by the Data-Collection platform to compute data collection decision. Bluetooth device's information is propagated to the master as a part of the inquiry process. This information is part of the FHS packet previously discussed. Apart from this, a friendly name parameter of size 248 bytes is transmitted by the slave device (sensor). Classic Bluetooth operations use the user-friendly names to identify other Bluetooth devices. We propose a modified usage of the friendly name parameter. We use the friendly name to transmit sensor meta-data i.e. data that describes the particular sensor. The 248 byte long friendly name is encoded using UTF-8 standard (BluetoothSIG, 2010d). This allows a typical Latin character to be encoded in 1 byte. We propose the following naming strategy:

$$< sensor\_name\ , sensor\_group, sensor\_type, x, y, z, residual\_energy, flags >$$

sensor_name: Identifies the sensor name. This information is defined before the sensor is deployed, though it is not a requirement. The sensor, once deployed, has the capability to self-assign a name during the initialisation phase.

sensor_group: This information identifies the group to which the sensor belongs. The grouping can be geographical location based or *sensor_type* based. This parameter can assume a null value during initial sensor deployment.

sensor_type: The sensor type identifies the kinds of sensors equipped on the particular sensor node. For example, sensors of type 1 comprises temperature sensor and sensors of type 2 comprise temperature and humidity sensor.

x,y,z: The sensor location is represented using the three parameters. This information is hard-coded in sensor that are within structured deployments while is determined using sensor localisation techniques in un-structured deployments.

residual_energy: This information indicates the sensors remaining energy. This information is updated by the sensor periodically.

residual_data: This information indicates the remaining number of packets that needs to be collected and delivered.

flags: Flags are application dependent values that vary based on requirements. We define certain flags but allow room for extension. The flags that may be pre-defined include C-critical, and I-ignore respectively. The critical and ignore flags are used to indicate the importance of data stored at the sensor node.

Figure 3.7 gives an illustrative example of the proposed sensor naming format. The total number of bytes used in the given example is 41. The maximum length of the name field is 10 bytes with a 3 byte constant that is appended to all sensors allowing 7 bytes for sensor specific name. As mentioned earlier, this name need not be hardcoded but can be computed during deployment. For example, the Bluetooth device address, sensor location and sensor type may be used to generate the friendly-name. Since the device address is unique, the generated name inherits its uniqueness. The prefix "EIS" in the given example is used to identify specific sensors, in this case a Mulle (EISLab, 2010) sensor developed at EIS labs Sweden. The separator operator "@" is used by the mobile data mule to parse the friendly-name.

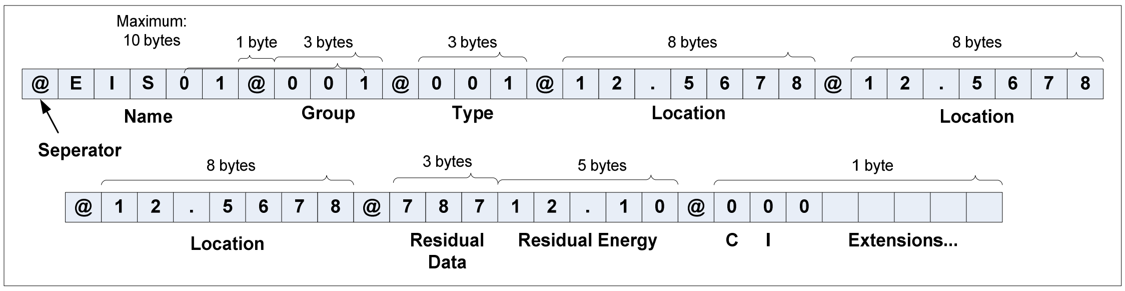


Figure 3.7: Example of proposed sensor naming format

Reducing the inquiry time only solves one part of the discovery problem. The second challenge with discovery involves timely discovery of sensor nodes by the mobile data mule i.e. the sensor node needs to be available for communication when the data mule is within the communication/collection range. This problem has been analysed by some researchers as a data mule scheduling problem (Somasundara et al., 2004, Gu et al., 2006) which involves adapting the data mule's arrival rate to the sensor network's data generation rate. This approach is promising but its scope may be limited to mobile data mules that work within a controlled sensor network environment. We use the term *controlled sensor network environment* to represent the class of sensor network applications that have a dedicated mobile data collector whose movement can be controlled by the application. Our approach addresses the scheduling problem by proposing an application-based sensor duty cycle adaptation approach to improve sensor discovery rate.

3.4.3.3 Sleeping Node Problem

The sleeping node problem occurs when a node that is passive (sleeping) to one mobile data mule (may be active to another mobile data mule) is declared inactive. The proposed data mule-based data collection approach is targeted at mobile devices that are part of smart space environments. Hence, mobile data mule's arrival is independent of each other. This mode of operation significantly differs from previously proposed approaches which assume coordinated operation (Shah et al., 2003, Jea et al., 2005). The sleep node problem is illustrated in Figure 3.8. Consider a sensor node S which communicates with mobile data mule M_1 at time t and mobile data mule M_2 at time t_2. If at time $t_2 + \Delta t$ when mobile data mule M_1 tries to communicate with sensor S and does not receive a response there arises two possible cases: 1) sensor S is dead 2) sensor S is in sleep mode as it offloaded its data at time t_2 to M_2.

To handle the sleeping node problem, we propose a sink-assisted synchronisation approach. The reason we propose a centralised technique is attributed to the fact that no single mobile data mule has complete information about the sensor network. Moreover, the mobile data mules perform data collection independent of one another. Hence, it is not practical to assume ad-hoc communication capabilities among mobile data mules that are part of the network. When a data mule is unable to discover/connect to a sensor, it sets an *inactivity counter* for that particular sensor. The *inactivity counter* is a value associated with each sensor. When the mobile data mule is unable to discover/connect to a sensor, it increments the inactivity counter value associated with the sensor. The *inactivity counter* value of each sensor is synchronised with the sink during the data delivery operation. The synchronisation operation compares the *inactivity counter* value stored at sink with the value obtained from the mobile data mule. The final inactivity counter value for the sensor S is computed by:

$$\boldsymbol{inactivity\ counter}_{(S)} = \begin{cases} \mathbf{0}, & sink_{inactive} = \mathbf{0} \\ \boldsymbol{sink_{inactive}}, & sink_{inactive} \geq \boldsymbol{datamule_{inactive}} \\ \boldsymbol{datamule_{inactive}}, & sink_{inactive} < \boldsymbol{datamule_{inactive}} \end{cases} \tag{3-1}$$

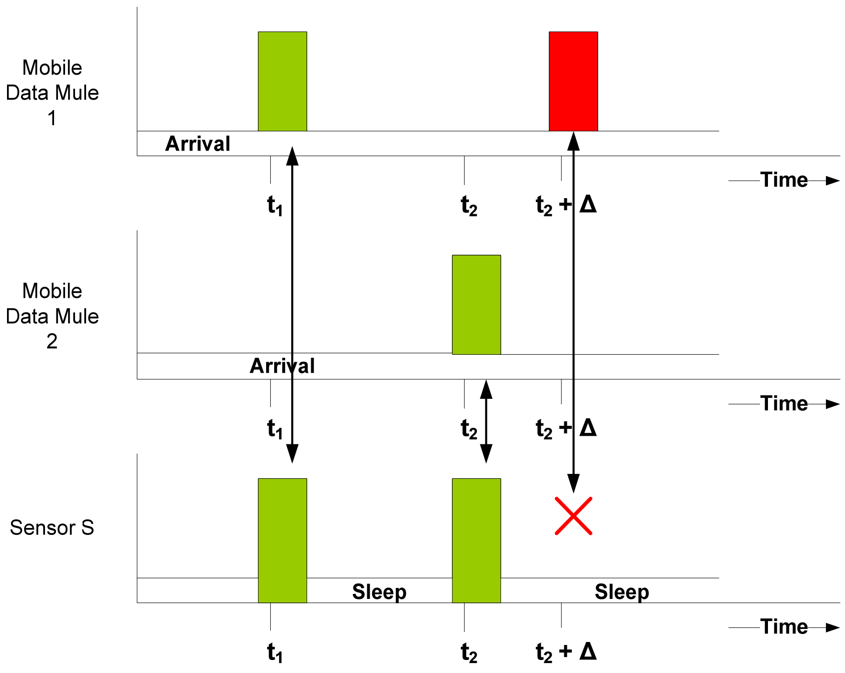


Figure 3.8: Sleeping Node Problem

3.4.3.4 Node Discovery and Management: Algorithm

The node management module maintains the list of discovered sensor nodes. It uses this information to perform data collection during subsequent arrivals. We use XML to store the sensor information on the mobile data mule. The use of XML allows us to easily share this information with the centralised sink and in certain special cases, with other mobile data mules. A sample XML schema is presented in Figure 3.9.

The mobile data mule's location tag is used to store location information of the data mule. This helps in mapping the list of sensors that were discovered at various data mule locations. The XML schema presented is only a sample with scope for extensions. Figure 3.10 presents the node discovery/management module pseudo code. The node discovery/management handles sensor node discovery/storage and the sleep node problem. A description of the algorithm is presented following Figure 3.10.

```
<?xml version="1.0">
<sensor_info>
      <sensor_id></sensor_id>
      <location>
            <x></x>
            <y></y>
            <z></z>
      </location>
      <sensor_group></sensor_group>
      <sensor_type></sensor_type>
      <residual_energy></residual_energy>
      <mobile_data_mule_location_id>
      <mobile_data_mule_location>
            <x></x>
            <y></y>
            <z></z>
      </mobile_data_mule_location>
</sensor_info>
```

Figure 3.9: XML Schema used to represent sensor information in the node repository

```
Pseudo Code: Node Discovery/Management
Input: Node List $L_N$ (Node Repository)
Output: discovered_list
BEGIN Discover Node
   1  discovered_list = discover(inquiry_time)
   2  for each node n in discovered_list
   3        search for n in $L_N$
   4        if found then
   5              Set inactive_counter = 0
   6        Else
   7              Add n to $L_N$
   8        end if
   9  end for
   10 for each node n in $L_N$ not in discovered_list
   11       if (location(n) is within location(mobile_data_mule)
   12             inactive_counter +=1
   13       end if
   14 end for
   15 connect to sink
   16 synchroniseSink($L_N$)
END
```

Figure 3.10: Node Discovery Module - Pseudo Code

Step 1 performs the Bluetooth discovery for the specified inquiry time. Steps 2 to 9 check newly discovered sensor nodes against the node repository. If an existing sensor node is discovered, its inactivity counter is set to 0. If a new sensor node is discovered, the sensor information is added to the node repository. Steps 10 to 12 increment the inactivity counter for sensor nodes that were previously discovered around the mobile data mule's current location. For example, consider the sensor node *S* was discovered when the mobile data mule was at location (x_1, y_1, z_1). *S* is marked inactive if discovery of *S* fails when the mobile data mule revisits the location (x_1, y_1, z_1). The vicinity in this case is determined by the Euclidean distance using a radius of 30 to 100 meters (Bluetooth radio range). Step 15 and 16 performs the synchronisation of sensor nodes with the sink i.e. synchronising L_N with the sink. The *synchronise* function employs the logic presented in section 3.4.3.3 to resolve sensor node inactivity.

3.4.4 Sensor Data Collection

The sensor data collection operation involves establishing connection with discovered sensors based on information collected during the discovery process. The *Data Collector* module handles this operation. Bluetooth employs the technique called paging (BluetoothSIG, 2010d) to establish a connection between the master and slave device. The process of paging is similar to the inquiry process described earlier. A successful paging operation indicates a successful connection. Since, our chosen sensor network platform is Bluetooth-based, establishing a connection is the most energy expensive operation. Hence, the challenge is to efficiently collect data by reducing the number of connections between the sensor and the data mule. In the proposed approach, we consider two cases: 1) a single mobile data mule can collect all the data from the sensor and 2) multiple mobile data mules collect data from sensors in parts. The use of mobility does not always guarantee single successful data transfer. The Data Mule (Shah et al., 2003, Jain et al., 2006) discussed in the literature work under the assumption that case 1 is likely feasible in most situations. Moreover, predicted (Chakrabarti et al., 2003) and controlled mobility approaches (Somasundara et al., 2006, Kansal et al., 2004, Jea et al., 2005) assume a controllable data mule. Our proposed approach employs independent mobile data mules as data collectors that may or may not be controllable. Hence, the data collection strategy needs to be independent of data mule control. It also needs to facilitate data collection from sensors by independent mobile data mule's i.e. mobile data mules that are not coordinated. To handle data collection using independent data mules we propose an adaptive data collection strategy. The mobile data mule is aware of its current location, movement direction, velocity and future location. Further, information from the sensor is collected during the discovery phase which includes sensor location, amount of data remaining, residual energy, etc. To handle the aforementioned data collection criteria we propose a sliding window (Tanenbaum, 2002) protocol inspired data collection algorithm. The connection-oriented operation of Bluetooth networks eliminates channel sharing problem as a dedicated channel is used between the mobile data mule and the sensor node. The proposed approach breaks the data transfer operation into multiple rounds. By breaking the data transmission into multiple parts/rounds, the sensor is able to clear its buffer partially, instead of waiting for a single data mule to collect all the data.

3.4.4.1 Adaptive Data Collection

The proposed adaptive data collection strategy uses contextual information to determine sliding window acknowledgement intervals. The window determines the number of packets transmitted between acknowledgements. Since, Bluetooth is connection-oriented, a disconnection is detected by the Bluetooth link layer (BluetoothSIG, 2010d). Once a disconnection is detected, the sensor stops sending any further packets. Figure 3.11(a) presents an illustration of successful data collection using the proposed adaptive data collection algorithm. The *start (w)* parameter is used to signal the sensor to start sending data packets. The parameter *w* determines the size of the window after which the sensor needs to wait for an acknowledgement from the mobile data mule. Due to the connection-oriented operation of Bluetooth, data packets transmitted between the sensor and the mobile data mule arrive sequentially. The type of Bluetooth packet used is application dependent. Figure 3.11(b) illustrates the operation of the adaptive data collection when a Bluetooth disconnection occurs during data transmission. In the figure, a Bluetooth disconnection occurs when packet 4 is being transmitted. The Bluetooth disconnection is depicted as a cross. When mobile data mule

2 begins data collection, the adaptive data collection algorithm resumes data transmission from packet 3 as it successfully received acknowledgement for packets 1 and 2.

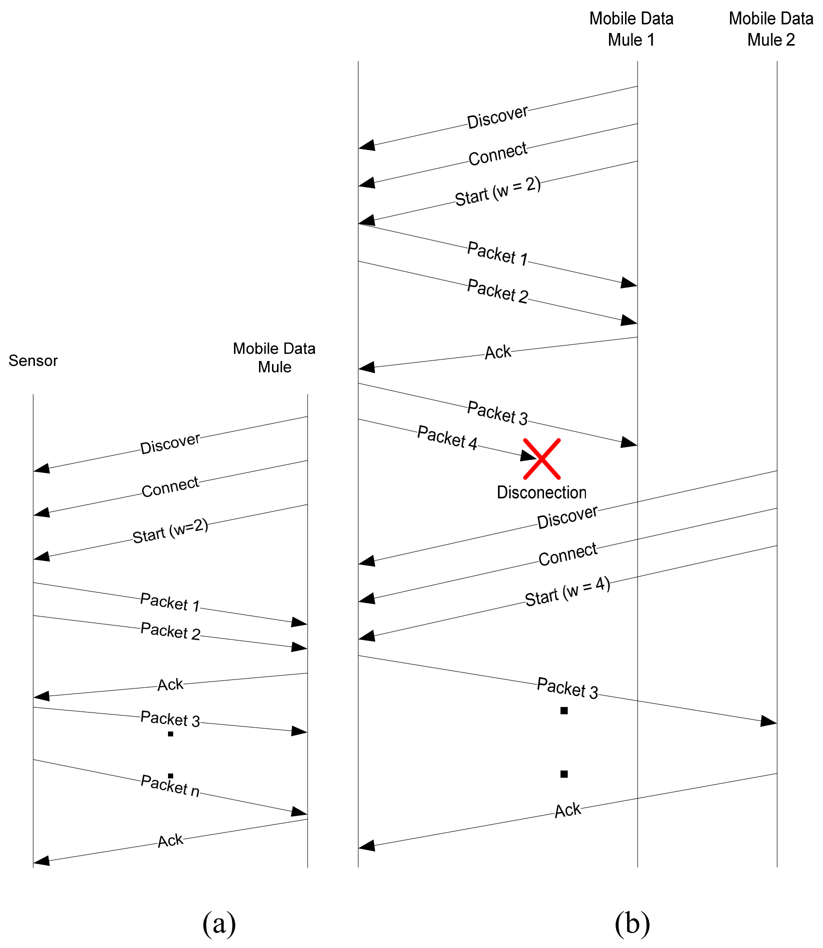


Figure 3.11: Adaptive Data Collection Algorithm - (a) Successful Data Transfer (b) Disconnection Handling

The primary consideration for the proposed adaptive algorithm is to determine the window size *w* i.e. the acknowledgement interval. A window size of one ($w = 1$) would be the most reliable but not efficient while a window size of *n* ($w = n$), where *n* is a very large integer, will prevent the acknowledgement from reaching the sensor before disconnection. Hence, it is important to use an efficient window size. To this end, we propose the use of contextual information available to the mobile data mule to compute a window size on-the-fly. The contextual information represents a set of mobile data mule parameters such as speed, trajectory and signal-to-noise ratio (SNR). The context information is used to compute a data collection threshold that determines if the mobile data mule has the capability to collect sensor data. Based on the threshold value, a suitable window size is computed. The threshold is used to determine the best mobile data mule among the available set of mobile data mules. Since, mules are independent and we do not assume inter-mule communication, the threshold computation is done at the mobile data mule-based on available context information with little or no assistance from the sink.

Consider a set P of context parameters $\{p_1, p_2, p_3 \dots p_n\}$ with a set V of values$\{v_1, v_2, v_3 \dots v_n\}$. Each context parameter in set *P* has an associated best case value given by$\{v_max_1, v_max_2, v_max_3 \dots v_max_n\}$. Set *V* represents the value of the context parameter at a specific instance in time. Set *P* represents the ideal case value of the corresponding context parameter in set *P* used to compute the influence (weight) of the context parameter. To determine the influence $I \in (0,1)$ for a value in set V, we use the formulae given below.

$$\boldsymbol{I_n} = \begin{cases} \dfrac{v_n}{v\_max_n}, & v_n \; \textbf{\textit{proportional to I}} \\ 1 - \dfrac{v_n}{v\_max_n}, & v_n \; \textbf{\textit{inversly proportional to I}} \end{cases} \qquad \textbf{(3-2)}$$

The computation of *I* is dependent on the relation between *I* and *v*. *I* and *v* are said to be inversely proportional if increase in value *v* reduces data collection efficiency. Similarly they are directly proportional if increase in *v* increases the data collection efficiency. For example, increase in distance would increase the data collection time while increase in SNR indicates better channel quality reducing data collection time. To compute the data collection threshold (φ), we propose the use of weighted average. The weights assigned to the context parameters are pre-defined and are shared across every mobile data mule. The data collection threshold is given by equation (3-3).

$$\varphi = \sum_{i=1}^{n} \frac{w_i * I_i}{w_i} \qquad \textit{(3-3)}$$

Example: Consider a sensor *S* with contextual information: residual energy e_r, location *(x, y, z)* and a mobile data mule *M* with context information current location *(x,* *y, z)*, trajectory vector$\overrightarrow{T_V}$, velocity *VL,* signal-to-noise ratio *SNR*. The residual time *t* is defined as the remaining time the mobile data mule will stay within the coverage of the sensor node calculated using the velocity *VL* and mobile data collector's trajectory information$\overrightarrow{T_V}$. The calculation of the residual time *t* can be further explained with the help of illustration presented in Figure 3.12. The distance before which the sensor leaves the boundary can be estimated as R_d. Since the mobile data mule's velocity is known, the residual time is given by:

$$\boldsymbol{Residual\ time\ (t) = T_{static} + \frac{R_d}{VL}} \qquad \textbf{(3-4)}$$

The parameter T_{static} is estimated as the time the mobile data mule is stationary at the specified location. We assume the T_{static} information is available as part of context within the environment. For example, consider a data mule waiting at an intersection. Using available traffic information obtained from active traffic management systems (UTMS, 2010) the time the mobile data mule will be stationary at the intersection can be determined.

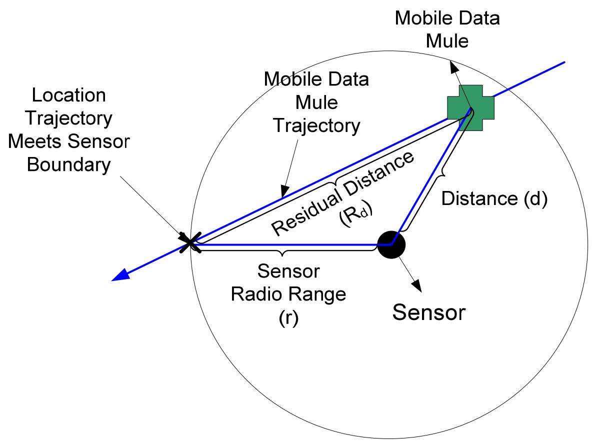

Mobile Data Mule
Location Trajectory Meets Sensor Boundary
Mobile Data Mule Trajectory
Residual Distance (R_d)
Distance (d)
Sensor Radio Range (r)
Sensor

Figure 3.12: Estimating residual time (t)

To calculate the value *v_max* for residual time i.e. maximum time required to transmit all the data, we use (5)

$$\boldsymbol{v\_max_t = \frac{x}{C}} \qquad \textbf{(3-5)}$$

where x is the amount of data to be transferred and C is the channel capacity. The unit for the quantities in this specific example is bits and bits per second. Since, the computation of v_max varies for each parameter a generic approach cannot be followed. For example, v_max for distance parameter may be maximum radio range. As the value v for radio range nears its v_max, the influence I for the parameter distance will tend to 0. Table 3.1 provides sample data based on the example scenario presented earlier. In the sample data we assume total bits to be transferred is 1120 kb and the channel capacity is 56 kb/sec. The data collection threshold (φ) for each case is computed.

Parameters / Sample Data	Distance (m) $w_i = 0.1$	Residual Time (seconds) $w_i = 0.5$	Residual Energy (%) $w_i = 0.2$	SNR (dB) $w_i = 0.2$	Data Collection Threshold (φ)
1	30	10	80	20	0.476
2	20	20	70	40	0.806

Table 3.1: Data Collection Threshold Computation- Sample Data

The data collection threshold is determined by the mobile data mule independent of each other. In a typical real-world situation multiple mobile data mules might compete to collect data from the sensor. Hence, we propose a data collection threshold that will enable a mobile data mule to collect data. If multiple mobile data mules have data collection threshold higher than the pre-set threshold, the data collection mechanism employs a first-in-first-out policy. We propose a simple technique used to dynamically improve the data collection threshold. More complex techniques e.g. genetic algorithms (Mitchell, 1998) may be employed to further improve this process. A basic approach could employ a first-in-first-out (FIFO) policy i.e. the first mobile data mule at the location collects the data since no initial threshold value is available. Subsequently, the threshold value along with the data collected is delivered to the sink. The sink compares reported threshold value by all mobile data mules for each sensor. With the condition that sensor's energy status is healthy (above 30%) the sink updates mobile data mules with new threshold value. The new threshold value is determined to be the highest among reported thresholds. As the sensor energy begins to deplete or when the data arrival latency of a sensor increases, the sink dynamically reduces the data collection threshold.

This operation can also be performed by the sensor by using the *flags* (refer to section 3.4.3.2). By setting the flag *I* (ignore) to 1 the data will be collected by the next available mobile data mule ignoring data collection threshold computation.

The computed data collection threshold is used to determine the window size. The higher the threshold, the higher the window size can be and vice versa. Since the threshold value is heavily dominated by the residual time *t*, our observation stands true and is supported by experimental runs. The actual mapping of window size to the data collection threshold is application dependent i.e. for a threshold range of 0.5 to 0.6 the window size can be 10 while for a threshold greater than 0.8 the window size is 20. This value is application dependent and can be dynamically updated. The adaptive sensor data collection algorithm is presented in Figure 3.13.

```
Pseudo Code: Sensor Data Collection
Input: Node List $L_N$ (Node Repository)
BEGIN Data Collection
   1  Discover Node
   2  while k < n (number of nodes in discovered_list)
   3        calculate $\varphi_k = \sum_{i=1}^{n} \frac{w_i * I_i}{w_i}$
   4        search for $\varphi(k \text{ in } L_N)$
   5        if $\varphi_k > \varphi(k \text{ in } L_N)$ or $\varphi(k \text{ in } L_N)$ = nothing
   6              establishConnection(w)
   7        end if
   8  end while
END
```

Figure 3.13: Sensor Data Collection - Pseudo Code

Step 1 calls the previously described discover node module which provides the list of discovered nodes in the surrounding along with the initial sensor contextual data. Step 2 - 3 iterates through the nodes in the discovered list computing the data collection threshold for each sensor in the list. Step 4 performs a search in the node list repository to determine a minimum threshold for the particular sensor. As we can see, this operation works in a disconnected fashion i.e. it does not require the sink's assistance to complete. Step 5 checks if the computed threshold is greater than the pre-defined threshold for the particular sensor. Step 6 calls a function that handles establishing connection and collecting data from the sensor using the defined window size *w*.

3.4.4.2 Data Exchange: Message Protocol Format

The previous subsection presented the adaptive data collection algorithm. In this section we present details of the proposed data exchange protocol. There are two types of messages used for communication, control messages and data messages. The *message parser* component of the *data collector* module takes care of parsing messages that are exchanged between the sensor and the mobile data mule.

The control messages are used to initiate the data exchange process. Some of the control messages include sending window size, requesting sensor to start streaming data, sending acknowledgement, etc. Further, the control messages allow dynamic upload and download of activation schedule which is discussed later. The data message is used to wrap the sensed data with application specific information. For example, in applications where the time and date of the sensor data is important, data sensed by the sensor is wrapped with time and date parameters. In applications where sensed data over a time period is required, the message header information contains timestamp periods. Since a generic method may not be applicable for the plethora of sensor network applications, we allow room for application specific definition. The control data bit used by the sensor and the mobile data mule is 3 bits in length. The message format is depicted in Figure 3.14 and the control bits used for negotiation are presented in Table 3.2. The control message has an optional payload which is used in cases when the window size needs to be negotiated with the sensor.

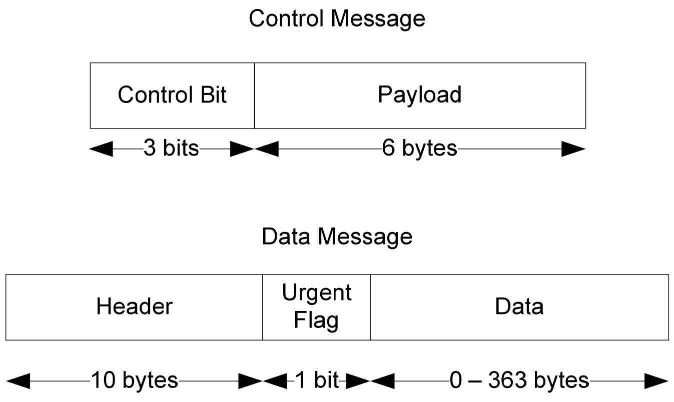


Figure 3.14: Control and Data Messages Format

Control Bit	Operation
000	Start Data
001	Window_Size
010	Acknowledgement
011	Wait
100	Activation_Schedule - Upload
101	Activation_Schedule - Download
110	End

Table 3.2: Control Bits used during Data Collection

3.4.5 Data Delivery

The data delivery operation is performed by the *sink manager*. The data delivery handles delivering the collected sensor data to the sink in a timely manner. The data delivery modes are restricted by the communication medium available on the mobile data mule. For example, a mobile data mule with GSM/GPRS capability can deliver data immediately while a mobile robot with WLAN capability needs to wait till it reaches an access point to deliver the data. In the proposed approach, the assumption is that mobile data mule has sufficient buffer capacity to store sensor data.

The data delivery module is responsible for identifying when and how data is to be delivered to the sink. The data exchange between the sink and the mobile data mule uses XML representation. This facilitates easy sharing of sensor data across multiple data sinks. The data format as mentioned earlier is application specific. The mobile data mule is not responsible for parsing collected data. It only acts as an intermediate relay between the sensor and the sink. The sensor data message illustrated in Figure 3.14 uses flags to indicate message priority. The sink manager only unpacks the header to determine the message delivery priority. It uses appropriate data delivery medium based on message priority. The pseudo code for data delivery based on message priorities is presented in Figure 3.15.

The *sensor data store* connected to the sink manager is a temporary sensor data buffer where data collected from the sensor is stored. This buffer is cleared when the data is delivered to the sink. The urgent data messages are delivered using priority communication that may incur cost e.g. GSM/GPRS. Sensor data that is not urgent is buffered until a cheaper communication source is available e.g. wireless local area network (WLAN). The introduction of urgent message aims to reduce the data delivery latency which is incurred in approaches proposed in the literature, where the mobile data mule needs to wait until it reaches the base station to off-load the data. The range of communication technologies available on *day-to-day* mobile devices alleviates this problem allowing timely delivery of high priority sensor data. In the next section we propose a novel dynamic sensor adaptation technique. This approach ensures timely sensor node discovery (by the mobile data mule) which results in increased sensor node lifetime.

```
Pseudo Code: Sensor Data Delivery
Input: Collected Data
BEGIN Data Delivery
   for each received message
   unpack header data
   if urgent_flag is set
         while delivery = done
               delivery data immediately using quickest
               medium
         end while
         delete data from buffer
   else
         read header timestamp
         store (timestamp, data)
   end if
END
```

Figure 3.15: Data Delivery-based on message priority - Pseudo code

3.5 Sensor Adaptation using Dynamic Activation Schedule

The discovery process requires the sensor to be constantly listening to the communication channel. Continuous listening to the channel increases discovery rate but reduces sensor lifetime. As identified in the literature, radio is one of the major energy consuming components of the sensor. Our proposed approach facilitates sensor adaptation using mobile data mules to improve sensor discovery rate. The term "*sensor adaptation*" refers to the process of changing the sensor's operational state based on application needs. To adapt sensors dynamically we propose the use of activation schedule that allows sensors to modify its operations on-the-fly. The term "*activation* *schedule*" refers to a set of instructions that determines the sensor's operational state. The assumption is that activation schedule is generated at the sink using powerful data mining (Chong et al., 2008) algorithms. The key challenge is to distribute the activation schedule across the entire sensor network. We propose the use of mobile data mules as a solution to this problem. The proposed mobile data mule-based dynamic sensor adaptation technique has number of applications other than improving discovery rate. For example, increasing sensing interval during specific time period based on application requirement.

The following sections focus on: 1) activation schedule format 2) protocol for dynamic exchange and decoding of the activation schedule between sensor and mobile data mule and 3) incorporating capabilities into the system to support activation schedule exchange. We begin our discussion by providing an overview on sensor operational states and transitions.

3.5.1 Sensor States/Operations and Transitions

The duty cycle D_c can be defined as the fraction of time the sensor is in an active state within a given time window given by (3-6)

$$\boldsymbol{D_c = \frac{T_{active}}{T_{Window}}} \qquad \textbf{(3-6)}$$

T_{active} is the amount of time the sensor is in active state and T_{Window} is the total time interval. For example, the duty cycle is 10% if the sensor is active for 6 minutes within a 60 minute time window. The cumulative duty cycle of the sensor can be estimated by the total amount of time the sensor is active. Figure 3.16 presents different sensor states modelled using finite state machine (FSM) representation. Each active state of the sensor corresponds to a specific sensor operation. The states illustrated in Figure 3.16 are highlighted to signify the amount of energy consumed during each corresponding operation. The sensor states are:

Sleep: This is the most power-efficient state where the sensor's components are put into continuous sleep. The only active component during this state is a real time clock that sends an interrupt to the microcontroller unit (MCU) when needed. This state is also referred to as low-power state.

Sense: The *sense* state represents the operation when the MCU requests the on-board sensor (temperature sensor) to perform a sampling operation. This involves the use of analog-to-digital convertor (ADC) to convert analog data to digital data. This state corresponds to the third most expensive operation performed by the sensor.

Radio Listen: In this state the MCU turns on the radio device into listening mode. In listening mode the sensor node performs a periodic inquiry scan of the channel until a successful connection is made. This state corresponds to the second most energy consuming operation performed by the sensor.

Transmit: In this state the connection between the sensor node and the mobile data mule is established and data transfer operation is performed. This is the most (first) energy expensive operation performed by the sensor.

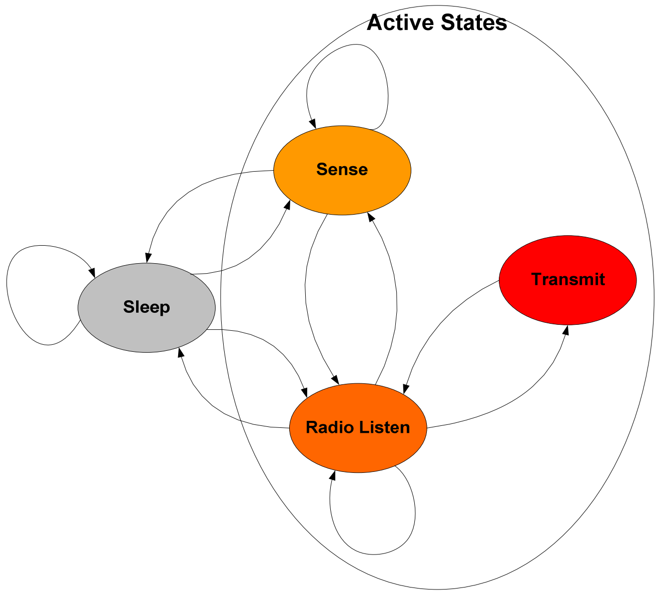


Figure 3.16: Sensor Node Operational State with State Transitions

The *sense*, *radio listen* and *transmit* states are cumulatively termed "*active states*" while the sleep state can be termed "*passive state"*. Since each sensor state consumes energy with the sleep state being the most energy-efficient, the sensors need to constantly change its state for energy-efficient operation. The state transitions represented in Figure 3.16 determines how the sensor changes from one state to another. As illustrated in Figure 3.16, except for the transmit state, the rest of the states are recursive i.e. the sensor can recursively extend its operation in the specific state. The state transitions are described below:

Sleep-Sense: The sensor moves from sleep state to sense state and vice-versa to perform the sense operation. Allowing the sensor node to remain permanently in the sense state is energy-expensive. On the contrary extended sleep might result in data (sensed data) loss. The trade-off between sleep and the sense state is determined by the sensor network application. From his state the sensor can switch back to sleep state or change to radio listen state.

Sense-Radio Listen: This transition happens when the sensed data needs to be transmitted immediately to the base station. The radio listen state may switch to the sleep state if a successful connection is not made within a pre-determined time window.

Radio Listen-Transmit: This transition happens when the sensor makes a successful connection with a data mule in the surrounding. The states transition defines how the sensor moves from listen to connected state.

Each sensor state except the transmit state can switch to the sleep state at any point in time. The state transitions depicted as arrows in Figure 3.16. When in the transmit state, the sensor has to return to the radio listen state before returning to the sleep state. This is done to avoid state transition when a connection (data transfer) is in progress. The total active duty cycle is computed as the cumulative sum of time the sensor spends in active states. The total energy consumed by the sensor is computed by the active and passive duty cycle duration given by (3-7).

$$\begin{aligned}\boldsymbol{Total\ Energy\ Consumed} &= \boldsymbol{E_{sense}} * \boldsymbol{D_{C(sense)}} + \boldsymbol{E_{radio\ listen}} * \boldsymbol{D_{C(radio\ listen)}} \\ &+ \boldsymbol{E_{transmit}} * \boldsymbol{D_{C(transmit)}} + \boldsymbol{E_{sleep}} * \boldsymbol{D_{C(sleep)}}\end{aligned} \qquad \textbf{(3-7)}$$

In the above equation the total energy spent by the sensor in each operational state is denoted by $E_{<state>}$ and the total active time in a particular state is given by $D_{c(<state>)}$.

3.5.2 Dynamic Activation Schedule

The previous section identified sensor states and corresponding state transitions. Further, we propose two modes of sensor operations that employ frequent sensor state transitions to maximise energy efficiency. The operating modes depicted in Figure 3.17 are:

Passive-Reactive mode: In passive mode, the sensor switches the radio to listen state permanently. This allows the data to be transferred immediately but is highly energy consuming.

Time Synchronous mode: In time synchronous operation the sensor switches states periodically. The time synchronous operation efficiently uses the active and passive states. This results in energy-efficient sensor operation. On contrary, alternating between passive and active states may result in sensing/data delivery delays.

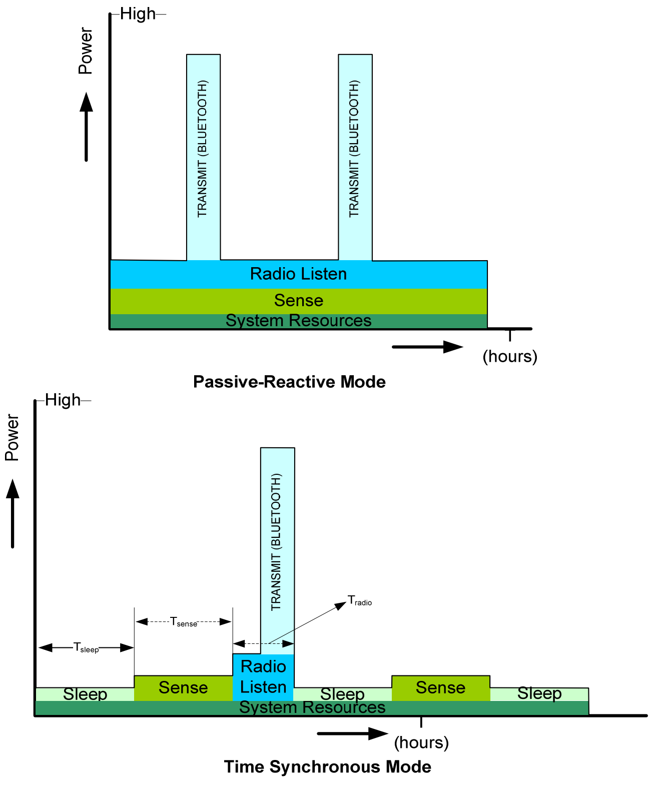


Figure 3.17: Sensor Operational Modes

In time synchronous operational mode the sensor periodically performs state transitions at pre-defined time intervals. The use of time synchronous operation mode guarantees energy efficiency but may not always guarantee successful sensor discovery (by the mobile data mule). This is because the classic time synchronous operation is not adaptive. Hence, the sensor listen duty cycle period may not match the mobile data mule's arrival. This results in extended listening periods or sensor buffer overflow (collected data overflow). To address these issues we propose a dynamic time synchronous operation mode. The dynamic time synchronous operation enables on-the-fly adaptation of sensor's duty cycle (active and passive states). The proposed framework facilitates on-the-fly sensor duty cycle adaptation. We introduce the term *activation schedule* to define the dynamic time synchronous operation. An activation schedule provides the sensor with duty cycle time periods i.e. the duty cycle for each sensor operation (state). By mapping the sensor's listen operation based on the arrival of the mobile data mule increases the probability of sensor discovery. Our proposal handles the challenges involved in updating the sensor with a new activation schedule. The distribution of the activation schedule is performed by the mobile data mule. We assert:

By enabling sensor adaptation using the proposed dynamic activation schedule approach, the sensor discovery rate can be improved.

To facilitate the exchange of activation schedule between the mobile data mule, the sensor and the sink we use an open source iCalendar (Dawson et al., 1998) format. The iCalendar allows us to define sensor operations over certain time periods. The use of iCalendar also facilitates easy sharing of the activation schedule among mobile data mules and sinks. Further, the iCalendar can be published on the internet with relative ease. This provides the user with the ability to visualise the entire sensor network. To encode sensor operations, namely, sense, listen and sleep into the iCalendar, we use the following constructs given by iCalendar specifications (RFC):

BEGIN: The entry point of the calendar is identified by the *BEGIN* keyword followed by a delimiter and *VCALENDAR* keyword.

END: The end of the calendar entry is identified by an *END* keyword followed by a delimiter and a *VCALENDAR* keyword

VERSION: This value determines the version of the iCalendar used.

VEVENT: It is a calendar component that defines a specific event. An event is an operation that is performed periodically over time. It starts with a *BEGIN* keyword and ends with an *END* keyword. In short, a *VEVENT* is a daily remainder. Within the *VEVENT*, the following values are used to define the event properties:

DTSTART: This value is used to identify the date from which the new schedule needs to be applied.

RRULE: This value is used to repeat an operation periodically forever based on certain conditions identified by frequency and repetition time. The *RRULE* has number of sub values that can be used to finitely define the frequency of event repetition. We use *FREQ* and *BYMINUTE*. The *FREQ* defines how frequent the operation needs to be repeated determined by a *value*. The *value* can be *SECONDLY*/*MINUTELY*/ *HOURLY*/*DAILY*/*WEEKLY*/*MONTHLY*/*YEARLY*. The *BYMINUTE* allows the system to define how often the operation needs to be repeated within the specified frequency. For example, perform sensing every 20 minutes daily.

DESCRIPTION: This is used to define the sensor operation.

We use *VEVENT's* to define multiple time intervals for various sensor operations. As mentioned previously, the proposed framework facilitates easy exchange of activation schedule between the sensor and mobile data mule. It also provides protocols and algorithms for activation schedule update and exchange. Figure 3.18 is an example of an activation schedule using the iCalendar format. The activation schedule has rules for radio listen and sense operation. The rule for *radio_listen* specifies that sensor needs to enter *listen* state every 30 minutes. The radio needs to stay in listen state for a period of 100 seconds. The *COUNT* parameter is used to indicate duration. A similar observation can be extended to the sense rule.

The mobile data mule receives activation schedule updates from the sink for an individual/group of sensors. The proposed system framework allows upload/download of activation schedule to and from the sensor node. The sink can take advantage of the availability of computing resources and complex data mining algorithms to compute the activation schedule. Classical sensor self-adaptation approaches (Somasundara et al., 2004, Younis, 2004, Bandyopadhyay et al., 2003) depend on cluster heads to compute the activation schedule. The self-adaptation requires the sensor to collect additional adaptation information in addition to its intended sensing operation (e.g. environmental monitoring). Moreover, the sensor may not be efficient for the sensor to perform complex computations given its resource-constrained operation. Our proposed approach eliminates such requirements reducing the workload on individual sensors. Using the proposed approach the sensor only needs to have the capability to decode the activation schedule (iCalendar). The algorithm presented in Figure 3.19 is used by the sensor to decode the activation schedule. The proposed activation schedule only specifies when and how long the sensor needs to perform a specific operation. The default state transition after performing a specific operation is the sleep state.

```
BEGIN:VCALENDAR
VERSION:1
BEGIN:VEVENT
DTSTART:20100511T21000Z
RRULE:FREQ=DAILY;BYMINUTE=30;COUNT=100
DESCRIPTION: RADIO_LISTEN
END:VEVENT
BEGIN:VEVENT
DTSTART:20100511T21000Z
RRULE:FREQ=DAILY;BYMINUTE=20
DESCRIPTION: SENSE
END:VEVENT
END:VCALENDAR
```

Figure 3.18: An Activation Schedule using iCalendar

3.5.3 Dynamic Activation Schedule: An Example

The proposed dynamic activation schedule can be employed in real-world scenarios. The sink computes activation schedules and the mobile data mule updates the sensors with the newly computed activation schedule. We further provide a comprehensive example of the dynamic activation schedule operation. We use the scenario depicted in Figure 3.20. The scenario is a traffic intersection where sensors are deployed to measure environmental phenomena. The sensors do not measure the flow of traffic.

```
Algorithm 1: Decode Activation Schedule
Input: iCal.ics (Activation Schedule)
Output: start, period, duration, operation
BEGIN Decode_Activation_Schedule
   1. open iCal.ics
   2. while not EOF
   3.       $e_i$ = read line
   4.       switch (SecondElement $e_i$)
   5.             case “VEVENT”
   6.             $e_j$ = read line
   7.             while FirstElement ($e_j$) not equal “END”
   8.                   switch (FirstElement($e_j$))
   9.                         case “DSTART”
   10.                                    Set start
   11.                              case “RRULE”
   12.                                    Set period
   13.                                    Set duration
   14.                              case “Description”
   15.                                    Set operation
   16.                         end
   17.                   do
   18.             end
   19.             setActivationSchedule(start, period,
      duration,   operation)
   20.      do
END
```

Figure 3.19: Algorithm to Decode Activation Schedule

We model the mobile mule arrival as a Poisson process (Cogill, 2009, Virtamo, 2010) where, a large population of independent mobile data mules arrive between time interval $(0, t_1)$. Using the previously described Poisson arrival process, the probability of seeing *m* mobile data mules can be computed using (3-8).

$$\boldsymbol{p(t_1, m) = \frac{(\lambda t_1)^m}{m!} e^{-\lambda t_1}} \,) \qquad \textbf{(3-8)}$$

Where, λ is the arrival rate, *m* is the expected number of mobile data mule’s and t_1 is the time interval. The function *p* (t_1, m) gives the arrival probability of exactly *m* data mules within the time interval $(0, t_1)$. To determine the activation schedule it is important to compute the arrival probability of the mobile data mule. This can be computed by first determining the probability that no arrival occurs *(m=0)*. Using the computed probability for *(m=0)* the probability that a mobile data mule will arrive within t_1 is given by:

$$p(t_1, m = 0) = 1 - \left(e^{-\lambda t}\right) \tag{3-9}$$

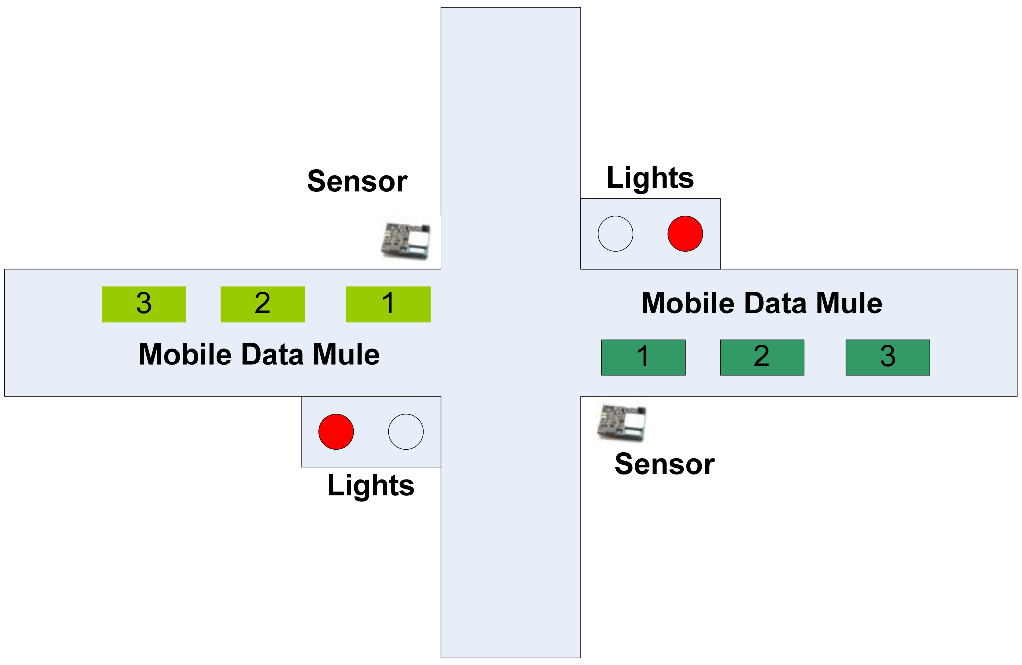


Figure 3.20: Dynamic Activation Schedule: An Example

For example, consider the arrival rate of mobile data mules at an intersection (Figure 3.20) is 50 every 60 minutes (1.666 mules/minute). The graph in Figure 3.21 presents the mobile data mules arrival probability for the time periods (1, 10) minutes. From the graph, it can be inferred that for the given data mule arrival rate, the probability of data mule arrival every 10 minutes is 0.99. Hence, the sensor can be adapted to listen to the communication channel every 10 minutes when frequent data delivery is required. By contrast, the sensor can be adapted for a 30 minute listen interval when data delivery latency is acceptable. In both cases, the Poisson arrival process is used to adapt the sensors listen state. This approach results in extended sensor lifetimes and guarantees sensor discovery with high probability. Further, the adaptation algorithm employed at the sink can use additional parameters like variable arrival rates during day and night to enhance the adaptation outcome.

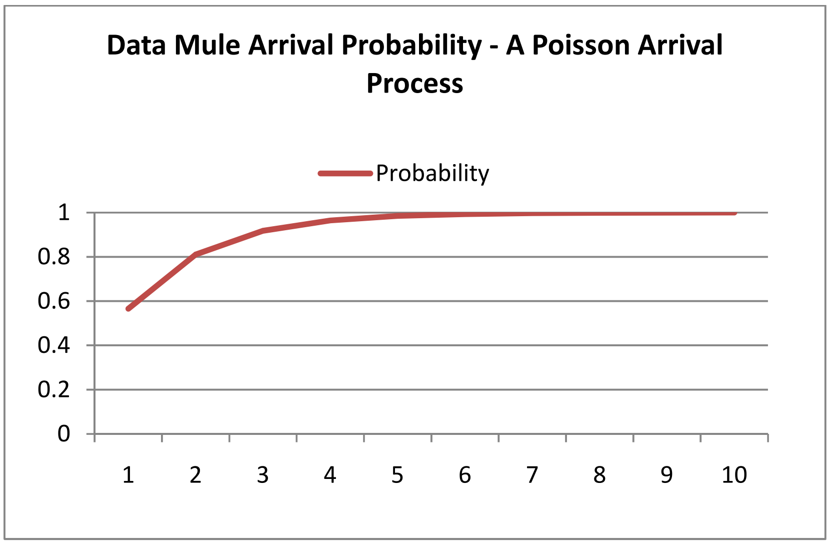


Figure 3.21: Probability of Data Mule Arrival - Poisson Arrival

3.6 Real-World Applications Scenarios using Mobile Data Mules as Sensor Data Collectors

In this section, we discuss two applications where the proposed mobile data mule based data collection can be employed without any special infrastructure set up.

iRoad (iRoad, 2010): iRoad is an on-going research project at *Luleå University of Technology, Sweden*. The project aims in making roads intelligent by integrating road surfaces with low-powered, autonomous wireless sensor nodes. The sensors embedded into the roads enable sensing parameters that can only be measured physically on the road surfaces. The use of sensor networks allows the sensed data to be transferred wirelessly for further analysis that would form the backbone of future traffic management systems. The sensor used in the iRoad project is the Mulle (EISLab, 2010) sensor node that uses Bluetooth for communication. The current Mulle used in the iRoad prototype implementation measures road temperature and uses vibration sensor to detect vehicle movement in the surroundings. Currently, this approach uses a fixed gateway to relay data to the centralised server for further processing and analysis. The Mulle sensor installed as a part of the iRoad system is equipped with solar cells allowing the sensor to harvest energy from the environment. Though energy is harvested from the environment installing a gateway over current road network is a painstaking and expensive task. Further, if the sensor is not able to harvest enough energy for sustained functioning (typical in countries like Sweden e.g. solar panel covered with snow) it needs to survive on battery to transmit the data. Our proposed system is a potentially advantageous cost-efficient alternative to the existing iRoad data collection architecture where vehicles can act as mobile data mules. Figure 3.21 is a typical example scenario of iRoad system with the proposed mobile data mule-based data collection technique. In the iRoad system the mobile data mule can be a mobile phone, a laptop or as a futuristic vision a car with the capability to use its inbuilt Bluetooth radio hardware. Our approach makes the assumption that data collection happens during times when availability of mobile data mules is abundant. During non-peak times, the sensor buffers data waiting for mobile data mule arrival. Our proposed approach can

1 Reduce infrastructure and installation cost by using existing mobile devices as mobile data mules. These devices are part of the existing infrastructure.

2 Allow adaptation of sensors using dynamic activation schedule. This enables sensors to adapt its operation based on mobile data mules' arrivals (e.g. traffic information).

3 Adapt the sensors based on climatic conditions e.g., the sensors can dynamically increase its sensing rate if it can infer with the assistance of the mobile data mule the probability that next day is a sunny day. This information can be used by the sensor to increase sensing rate and reduce data delivery latency as with certain probability it is guaranteed to recharge itself the following day.

District Heating (Deventer et al., 2009): District heating is another key application area currently studied at the *Luleå University of Technology, Sweden*. District heating enables heating of house spaces and provides hot water from a single energy source to many buildings. The energy from the primary source is routed using pipes to apartments and buildings. A heat exchanger installed within the building, called the secondary circuit, is used to transfer the energy (Deventer et al., 2009) from the primary source. At the secondary circuit the amount of energy is controlled using a control valve which calculates the energy used. The primary and the secondary unit in the current system are disconnected. Deventer et al. (Deventer et al., 2009) propose a wireless sensor network-based architecture to efficiently control the amount of energy used by the secondary circuit.

A key requirement with the district heating system (currently used in Sweden) (Deventer et al., 2009) is to provide online energy usage statistics to end users. Deventer et al. (Deventer et al., 2009) have used Mulle sensor nodes to connect the primary and the secondary circuits. This integration is built into the secondary circuit which controls energy requirement in real-time. The Mulle used in the current approach requires a fixed base-station to communicate with the rest of the world. This is feasible in test deployments but may not scale well in real-world scenarios. We propose the mobile data mule-based data collection approach as a suitable cost-efficient alternative. The mobile data mule within the scope of this application may be a mobile phone. The mobile phone when within the house can communicate with the Mulle collecting data and deliver it to the centralised sink using in-house communication capabilities e.g. wireless local area network (WLAN). The proposed dynamic activation schedule technique can be further employed to match sensor operation with the mobile device user's presence. This approach further enhances the energy-efficiency of the sensor node.

3.7 Summary

This chapter has presented our proposed system framework for sensor data collection using independent intelligent mobile data mules. Our framework targets the use of *day-to-day* mobile devices available within pervasive environments as sensor data collectors. This architecture is made feasible with the advent of Bluetooth-based sensor networks (BSN) and the widespread acceptance of Bluetooth technology. The proposed approach employs a restricted Bluetooth discovery protocol to reduce Bluetooth inquiry time used to discover sensor nodes within the surrounding. A sliding window based data collection algorithm is proposed to achieve sensor data collection using multiple independent mobile data mules. Our proposed approach does not require the introduction of any special hardware within the sensor network environment/infrastructure. The proposed system architecture can be implemented on any *day-to-day* mobile device without any specific hardware modifications. Moreover, our proposed approach computes data collection decisions on the mobile data mule hence reducing the computations on the sensor. Finally, we have proposed the use of dynamic activation schedule to adapt the sensor's duty cycle to increase discovery rate. The activation schedule is uploaded from/downloaded to the sensor on-the-fly. A detailed description of the activation schedule format and algorithms to supports its exchange between sensor, mobile data mule and sink was presented.

The proposed system framework can be easily adapted in real-world with *iRoad* and *District Heating* applications as examples. Summing up, our proposed approach is a cost-efficient unified architecture for sensor discovery, data collection and delivery using *day-to-day* mobile devices. The proposed system framework works in a decentralised fashion making use of the centralised sink resources when available. The proof-of-concept implementation and evaluation of the sGaRuDa framework is presented in Chapter 6.

4

3D-KNN: Sensor Data Collection using Nearest Neighbour Search in 3D Space

4.1 Introduction

In this chapter, we propose 3D-KNN, a *k*-Nearest Neighbour based approach for sensor data collection employed on the mobile data mule. Traditionally, *k*-Nearest Neighbour query has been used to compute a set of nearest object to a given location with the assumption that the object are on the same place (two-dimensional). Hence, most *k*-Nearest Neighbour approaches employ only distance to compute the nearest neighbour. Different from current approaches, we propose the 3D-KNN algorithm that can account for sensor distribution in three-dimensional spaces. Moreover, the use of *k*-Nearest Neighbour queries to collect data from sensors has not been used extensively. Further, the sGaRuDa framework and the corresponding algorithms proposed in Chapter 3 focus on Bluetooth-based sensor networks. Due to large growth and acceptance of Bluetooth, we see a potential for the proposed architecture in current pervasive environments. We leverage the availability of Bluetooth-based devices to achieve cost-efficient sensor data collection. Though we see Bluetooth-based sensor networks as one of the future enablers of pervasive environments, in this chapter we propose data collection algorithms that suit sensor networks that have broadcasting capabilities. We identify certain limitations of our previously proposed data collection system framework before introducing the 3D-KNN algorithm. The limitations of the sGaRuDa framework proposed in Chapter 3 are:

1) The use of Bluetooth solves the problem of sharing a frequency between multiple sensors and mobile data mules. This is good in cases where the sensor network is sparsely deployed. In dense sensor network deployments Bluetooth can only support a maximum of seven simultaneous connections. This restricts data collection in densely deployed large-scale sensor networks.

2) The use of Bluetooth also restricts multi-hop data communication. The use of multi-hop data communication requires additional network setup involving maintenance of master and slave nodes. This introduces the challenge of identifying the master node within the data collection coverage area.

3) The advent of Zigbee-based sensors (CrossbowTechnology, 2010a, ZigBee, 2009a) and Zigbee based mobile devices (ZigBee, 2007) have created new opportunities for deploying sensors in smart spaces. For example, sensor nodes with dual radios (ETH-Zurich, 2007, EISLab, 2010) i.e. Bluetooth and Zigbee, have opened the doors for techniques that use Zigbee based multi-hop data routing for control information and Bluetooth for data transfer.

The proposed sensor data collection approach employs independent mobile data mules to facilitate cost-efficient multi-hop data collection. Our data collection approaches are still driven by notion of mobility. The proposed 3D-KNN algorithm uses *k*-Nearest Neighbour (*kNN*) queries for cost-efficient data collection. The use of *kNN* allows the mobile data mule to select a subset of sensors to collect data from. The data collection algorithm efficiently uses the underlying sensor network's broadcasting capability to facilitate multi-hop data collection. Our proposed data collection philosophy is further supported by the development and availability of Zigbee-based sensors (CrossbowTechnology, 2010a, ZigBee, 2009a) and Zigbee-based mobile computing devices (ZigBee, 2007). The rest of the chapter is organised as follows. Section 4.2 provides an overview of *kNN* query-based data collection with focus on sensor networks. Section 4.3 presents a theoretical investigation of Voronoi-based *kNN* queries motivating the need for the 3D-KNN algorithm Section 4.4 presents in-depth discussion of the 3D-KNN algorithm. Section 4.5 presents a chapter summary. The 3D-KNN algorithm and the Voronoi-based *kNN* queries presented in this chapter are from the following published papers (Jayaraman et al., 2008b, Jayaraman et al., 2010a, Jayaraman et al., 2010c, Jayaraman et al., 2010b).

4.2 *kNN* Query-based Data Collection

The proposed data collection approach based on *k*-Nearest Neighbour queries is used to determine a subset of sensors within the sensor network around the mobile data mule. We term this subset that encompasses the *k* nearest neighbours as "*collection area*". The collection area used in the context of this thesis refers to the area (surface area or volume) of the three-dimensional space that encompasses the *k* nearest sensor nodes. The collection area is computed on-the-fly by the mobile data mule. Hence, the boundary and the size of the collection area shrinks/expands based on the mobile data mule's movement. Figure 4.1 presents an illustration of the collection area identified by the spheres CA_1 and CA_2 with radius R_1 and R_2.

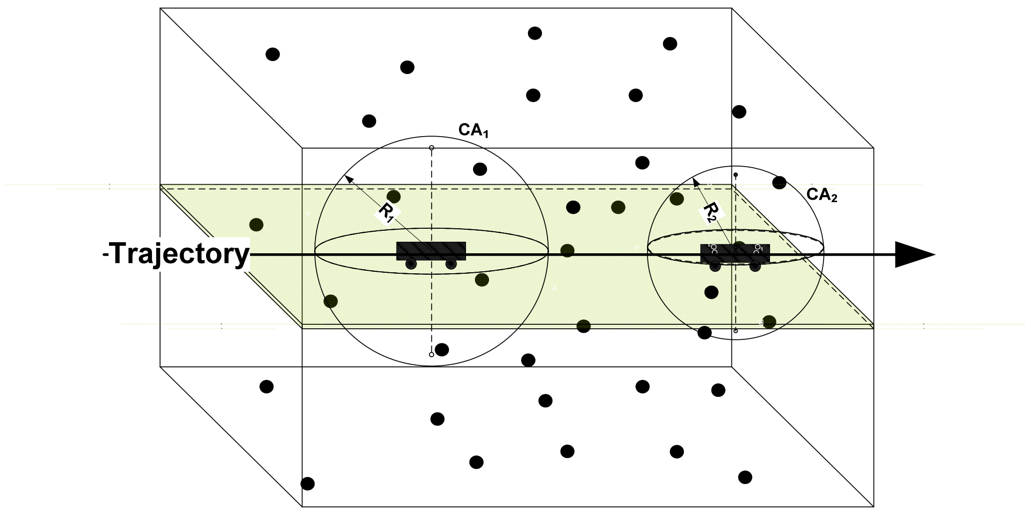


Figure 4.1: Collection Area Illustration

The line marked as *trajectory* represents the movement path of the mobile data mule. As illustrated in the figure the radius of the spheres vary depending on sensor coverage and mobile data mule location. The sensor data collection approaches using mobile data mules discussed in the literature (Shah et al., 2003, Jain et al., 2006, Somasundara et al., 2006, Kansal et al., 2004, Chakrabarti et al., 2003) make specific assumptions that sensor network deployment is two-dimensional. Some approaches (Shah et al., 2003, Jain et al., 2006) assumes an error and obstacle free grid-based environment. The assumption of three-dimensional space introduces the challenge of energy-efficient data collection across different planes given the omni-directional characteristics of radio transceivers. The illustration in Figure 4.1 is a classic example of a three-dimensional space. For example in Figure 4.1 at the first location the collection area encompasses sensors that are above, below and on the mobile data mule's movement plane. This omni-directional communication property of radio hardware requires approaches that need to compute nearest neighbours, taking into consideration radio communication metrics rather than just distance. By introducing radio characteristics with distance, we state that collection area comprises a subset of sensor nodes that are not just close but are also energy-efficient (good communication channel characteristics) to communicate with.

4.2.1 *k*-Nearest Neighbour Query Processing in Sensor Networks

We discuss the theory of *k*-Nearest Neighbour (*kNN*) queries more specifically in the context of sensor networks without mobile data mules. *k*-Nearest Neighbour queries have been used in sensor networks to retrieve data from sensors surrounding a point of interest. The point of interest is a specific location within the sensor network around which sensed data is collected. This location is computed by the application running at the sink. Chapter 2 discussed the classification of *kNN* query processing in a sensor network based on available network topology information. This infrastructure information is maintained using indices as used in traditional databases. Each sensor encompasses a set of child nodes grouped by their geographical location. A typical *kNN* query processing in sensor networks illustrated in Figure 4.2 involves the following steps:

Query Creation: This is the stage where the query request is generated by the application. The query contains the point of interest *(P)* location.

Query Propagation: The generated query is propagated within the sensor network until it reaches a *correspondent node* adjacent to the point of interest *P*.

Boundary Computation: The sensor node *s* is now responsible for processing the *kNN* query. To process the query, it first needs to compute a *kNN* boundary around *P*.

Data Collection: Once the boundary is computed, sensor data within this boundary is collected and the result is routed to the correspondent sensor node.

Query Result Propagation: The query result is propagated by the sensor node to the sink.

The *correspondent node* in Figure 4.2 is depicted in orange. The *correspondent node* is responsible for processing the query on the sink's behalf. Its operations include boundary estimation, query propagation, nearest neighbour selection and result propagation. The arrow marks represent the flow of data within the sensor network.

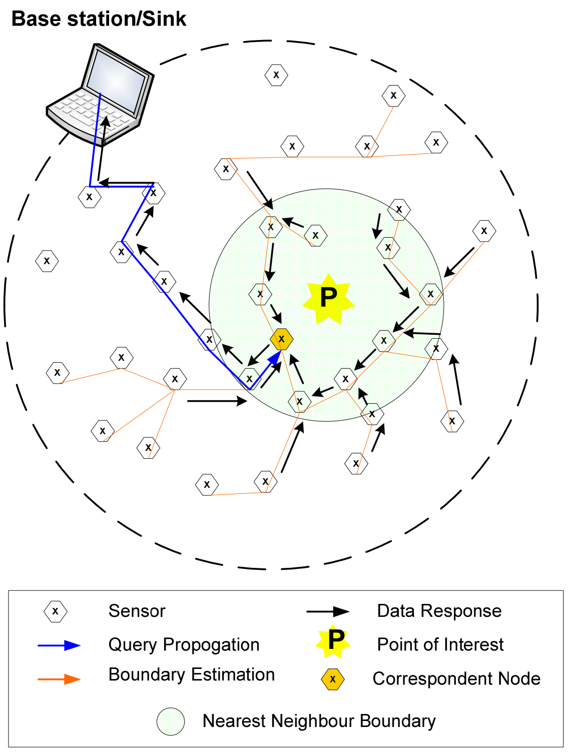


Figure 4.2: *kNN* Query processing - Overview

4.2.2 Overview of Proposed *k*-Nearest Neighbour Approach

In this subsection, we present an overview of the proposed multi-hop data collection approach using *kNN* queries. We employ *kNN* queries to a) compute the collection area that comprises a set of sensors around the mobile data mule and b) collect data from a subset of sensors within the collection area. We explore multi-hop data collection using *kNN* queries in a sensor network where mobile data mules collect sensor data.

Classic broadcast-based *kNN* query processing approaches discussed in the literature employ static sinks to process *kNN* queries. Some approaches (Soheili et al., 2005, Demirbas et al., 2003, Guttman, 1984) depend on the availability of spatial indices to process *kNN* queries while others assume physical clustering (Abbasi et al., 2007, Younis, 2004, Bandyopadhyay et al., 2003). The mobility-based data collection (Somasundara et al., 2006, Kansal et al., 2004, Jea et al., 2005) approaches that explore multi-hop sensor data collection also require physical clusters. Further, other mobility-based data collection approaches (Chakrabarti et al., 2003, Shah et al., 2003) presented in the literature do not deal with multi-hop data collection. They work under the assumption that mobile data mule comes in direct contact with sensor nodes. We present a short summary to identify the key differences between the proposed data collection approach and (Shah et al., 2003, Jain et al., 2006, Somasundara et al., 2006, Kansal et al., 2004, Chakrabarti et al., 2003). The summary is an analysis of mobile and static sink-based approaches presented in chapter 2.

1) The mobile data mule functions independently. Each data mule has the capability to compute data collection area dynamically. Our proposed system does not allocate mobile data mules to specific groups of sensors i.e. fixed sensor collection area based on geographical location.

2) The proposed *kNN* algorithm does not depend on network infrastructure (spatial indices) information to process *kNN* queries. This reduces the overheads introduced by index maintenance.

3) The proposed *kNN* algorithm does not require the existence of cluster heads within the sensor network infrastructure. Current approaches assume the existence of high-powered sensor nodes which act as cluster heads (Somasundara et al., 2006, Kansal et al., 2004, Jea et al., 2005). The use of

clusters introduces the need for a training phase during which the mobile data mule (Jea et al., 2005, Somasundara et al., 2006) collects cluster information. This requirement introduces additional overheads caused by cluster maintenance. Moreover, this approach is advantageous in the existence of a dedicated sensor data collector. In our scenario, we assume the data mules are not controlled and act independently. The probability that a data mule may visit the same set of sensor nodes is less likely.

4) The proposed algorithm functions within three-dimensional sensor network spaces which is typical of current day sensor network environments e.g. buildings. Our approach to investigate a three-dimensional sensor network space is motivated by recent research that concludes that sensors are spatially distributed (Ganesan et al., 2004). To this end, we propose data collection metrics that take into consideration energy consumed while communicating in the presence of obstacles (e.g. ceilings, walls). To the best of our knowledge our project is a pioneering effort in addressing sensor data collection in three-dimensional spaces.

The proposed data collection approach based on *kNN* queries can be further elaborated using the illustration in Figure 4.3. In our proposed approach, the point of interest is the mobile data mule. The aim of the query is to compute a set of sensors that are around the mobile data mule from which data is collected cost-efficiently. We use the term "*cost-efficiency*" to represent a function of cost parameters including: 1) communication (energy), 2) query processing latency (performance) and 3) overall network lifetime (total energy consumed). The proposed approach avoids the need for cluster heads/correspondent nodes. The mobile data mule performs the tasks of the cluster head/correspondent node. The introduction of the mobile data mule removes the need for query propagation and result propagation stages as the mobile data mule is responsible for collecting and delivering the data to the sink. The mobile data mule has the capability to dynamically compute the collection area on-the-run using available sensor network infrastructure information. As sensor infrastructure information is collected on-the-fly the proposed approach adapts well for changing sensor network topologies. The capability of the mobile data mule to compute the collection area is presented in detail in section 4.4.

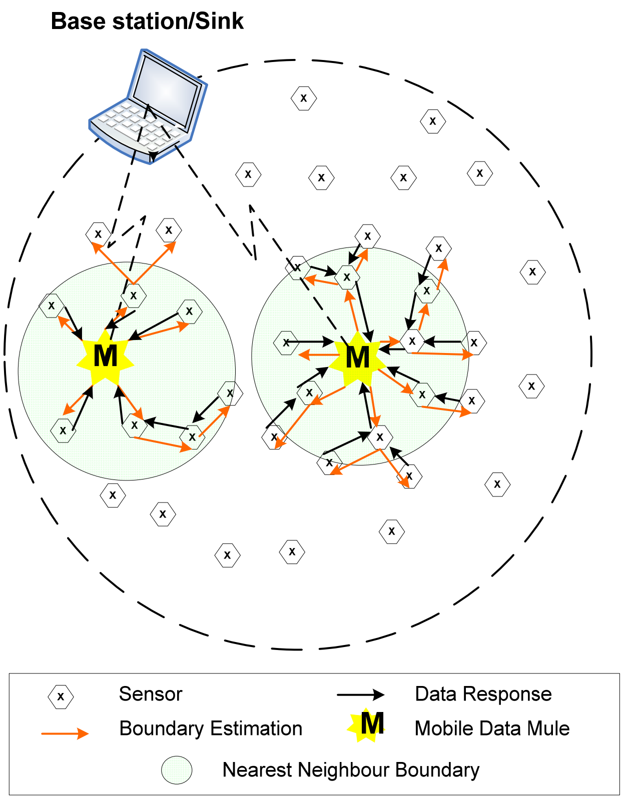


Figure 4.3: Proposed *kNN* query-based Data Collection Approach

4.3 A Theoretical Investigation of Voronoi Diagram-based *k*-Nearest Neighbour Search

In this section, we present a computational geometry (Mulmuley, 1993) inspired approach to process *kNN* queries. The technique we employ is Voronoi diagrams (Aurenhammer, 1991, Sen et al., 2010) and its Delaunay Triangulation. Voronoi diagrams have been used to efficiently cluster static sensor networks (Bandyopadhyay et al., 2003, Ghiasi et al., 2002) and have been applied to solve (Butler et al., 2003) sensor network coverage problems. The sensor network coverage problem addresses placement of sensors within the environment such that environmental changes are detected uniformly. We explore the use of Voronoi diagram and Delaunay triangulation for Bluetooth-based sensor networks.

4.3.1 Voronoi Diagram and Delaunay Triangulation

Definition 4-1: (Voronoi diagram): Given a set P with p_1 ... p_n points on a plane, a Voronoi diagram partitions the plane into n convex polygons confining one point each such that any point comprising the polygon is always closer to the point p_i in the within polygon than any other point in the set P.

The Delaunay triangulation (Aurenhammer, 1991, Sen et al., 2010) is a dual to Voronoi diagram and is obtained by connecting the points that share the same vertices in the polygon. The Delaunay triangulation, once computed from Voronoi diagram, can be used to compute the nearest neighbours and the path connecting them. Figure 4.4 shows an example of a Voronoi diagram and the corresponding Delaunay triangulation generated in MATLAB.

4.3.2 *k*-Nearest Neighbour using Voronoi Diagrams

We employ Voronoi diagrams to compute the *k* nearest neighbours around the mobile data mule. In the proposed data collection approach the mobile data mule is responsible for computing the Voronoi diagram.

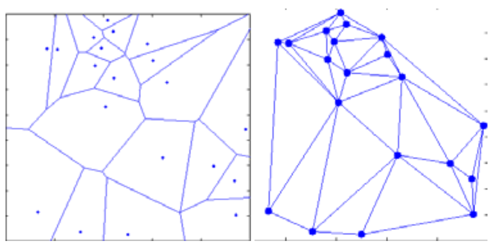


Figure 4.4: Voronoi diagram and Delaunay Triangulation

The Bluetooth inquiry process is used to discover the nodes around the mobile data mule. Once this information is available, a Voronoi diagram is computed by the mobile data mule. The sensor characteristics collected during the discovery phase are used to convert the Voronoi diagram into a weighted Delaunay graph. The weight of the Delaunay graph is assigned to the edge that connects the sensor node and the mobile data mule. This value is computed using the threshold approach proposed in Chapter 3. The threshold takes into consideration the sensor's distance, its residual energy, amount of data, etc. The Voronoi diagram and the Delaunay graph are used to compute the collection area that comprises the *k* nearest sensor nodes. The data collection is performed using the previously proposed multi-part data collection algorithm (refer Chapter 3). The Voronoi diagram is not stored at the sensors due to the space complexity of Voronoi diagrams (Aurenhammer, 1991). The proposed approach is a theoretical investigation of the feasibility of Voronoi diagrams for *kNN* computation. Our proposed Voronoi-based data collection approach is illustrated in Figure 4.5. The Voronoi diagrams were computed using MATLAB. Figure 4.5 (left) is an illustration of a sensor network with a mobile data mule. The red line depicts the mobile data mule's trajectory. Figure 4.5 (right) shows the corresponding Voronoi diagram. We assume the mobile data mule computes collection area at specific locations within the sensor network. These locations are marked as red arrows in Figure 4.5.

In our proposed system, we do not assume a dedicated mobile data mule. Hence, when computing the Voronoi diagram two types of sensor nodes need to be taken into consideration. They are sensors that are pre-known (node repository) and sensors that are newly discovered. Newly discovered nodes can fall into two categories: newly deployed and sensors that were passive/sleeping/inactive during previous visits by the mobile data mule. Given the dynamic nature of sensor networks and unavailability of network topology information, the Voronoi diagram needs to be recomputed at each instance. The cost in re-computing the Voronoi diagram is highlighted later in this section.

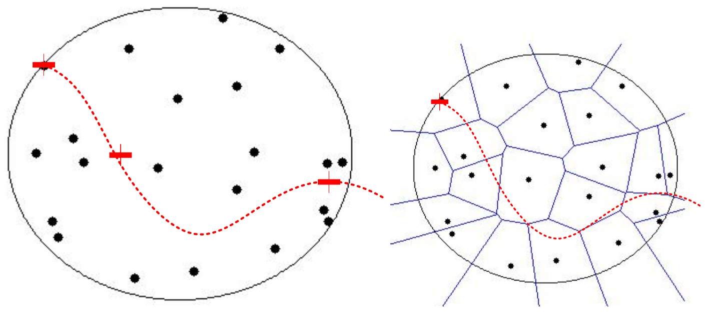


Figure 4.5: Dynamically Generated Voronoi diagrams for different Mobile Data Mule Locations

The Voronoi-based data collection approach satisfies a static sensor network that has the availability of a dedicated mobile data mule. With multiple independent mobile data mules, constant re-computation of the Voronoi diagram is expensive. This issue worsens when the size of the sensor network increases. Moreover, the complexity of the Voronoi diagram is $O(n^2 \log n)$ with the best-case complexity using a divide-and-conquer approach being $O(n \log n)$ (Sen et al., 2010). Also, the Voronoi diagram has a space complexity of $O(n)$. The above observations are based on Voronoi computation using Fortune's algorithm (Fortune, 1986). Based on these observations, we identify that Voronoi diagrams are more suitable for cases where the re-computation involved is minimal e.g. to partition the network (Bandyopadhyay et al., 2003, Ghiasi et al., 2002). The use of Voronoi in mobile sensor networks can be justified by the argument that mobile sensors are a permanent part of the sensor network, reducing re-computations. In our proposed approach the mobile data mules that perform data collection are not necessarily part of the sensor network infrastructure. Moreover, there is no guarantee that a mobile data mule might periodically visit a specific set of sensors. Further, a single mobile data mule might not have information on the entire sensor network. It is hence essential to compute the collection area dynamically during every data collection process.

We investigated the feasibility of Voronoi-based approach for a Bluetooth-based sensor network as the number of nodes that mobile data mule can communicate simultaneously is restricted to seven. This feature of Bluetooth-based sensor networks reduces time and space complexity in computing Voronoi diagrams. In densely deployed broadcast-based sensor networks, there is no restriction on the number of simultaneous connections. Hence, the time and space complexity of computing Voronoi diagrams increase by the order of n. Due to the space and time complexity of Voronoi diagrams for changing sensor network topologies, we propose a *kNN* query processing algorithm that uses simplified search and sort techniques with a time complexity of $O(\log n)$. In the next section, we discuss our proposed *kNN*-based data collection algorithm within the scope of a broadcast-based wireless sensor networks.

4.4 3D-KNN: A *kNN* Query-Based algorithm for Sensor Data Collection

In this section, we propose our novel three-dimensional *k*-Nearest Neighbour algorithm, namely, 3D-KNN, for a broadcast-based sensor network. The 3D-KNN computes a dynamic collection area using sensor information collected on-the-fly. Our proposed algorithm addresses real-world distribution of sensors by adopting a three-dimensional sensor network model. The proposed 3D-KNN algorithm considers the following:

1) The mobile data mule is centred about the collection area which encompasses a group of sensor nodes
2) The mobile data mule has no prior knowledge of the sensor network topology required to process the *kNN* query
3) Query dissemination and data collection is done by the mobile data mule
4) Sensors employ in-network aggregation to collect data from neighbouring nodes. Further, sensors do not have prior knowledge of neighbours as this is learned during query execution
5) Use of real-world radio signal propagation characteristics while computing nearest neighbours i.e. taking effect of obstacles and planes in three-dimensional space

In the following sections, we first present our sensor network model. Secondly we present the proposed 3D-KNN algorithm. Finally, we extend the proposed algorithm by introducing a prediction approach based on mobile data mule's path to improve the data collection efficiency.

4.4.1 Network Model and Assumptions

An illustration of the 3D sensor network model with the presence of a mobile data mule is depicted in Figure 4.6. We segment the three-dimensional space into planes. For example, in a building environment the planes correspond to the floors/levels. We assume that mobile data mule moves along a specific plane and sensor nodes are distributed within the *three-dimensional* space separated by levels (ceiling/walls). Sensor nodes are assumed to be location aware. Since radio range has no boundary, the discovered nodes can belong to different planes i.e. planes above the mobile data mule or planes below the mobile data mule. These planes are represented as *plane 1*, *plane 2* and *plane 3* in Figure 4.6. For illustration purposes, we consider three planes while the approach can be extended to *p* planes. The 3D-KNN algorithm employed by the mobile data mule uses a metric to determine the set of cost-efficient groups of sensor nodes. This metric is modelled taking into consideration real-world radio characteristics of sensors collected during the discovery process. To simplify our discussion, we assume that sensors deployed within the three-dimensional space are homogeneous i.e. they are identical, having the same hardware and radio range. However using heterogeneous sensors requires no modification to the 3D-KNN algorithm.

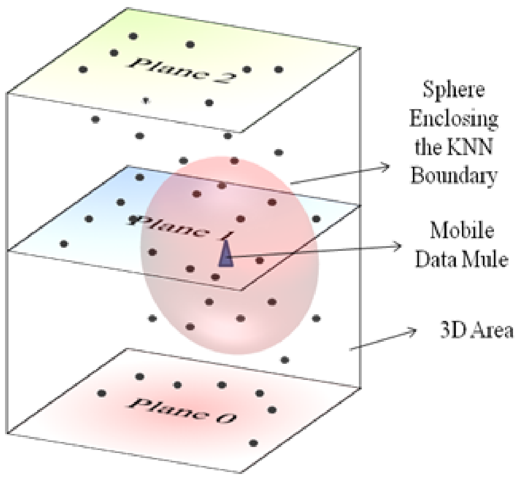


Figure 4.6: A Three-Dimensional representation of the KNN Boundary Estimation Algorithm

The mobile data mules in the proposed approach are independent of each other and the availability of a sink is not ubiquitous i.e. mobile data mules have the ability to work in disconnected mode. In the rest of the chapter, we focus on "*Single-Shot" kNN* queries. *Single-Shot kNN* queries are issued once and the results are computed based on the response. A subsequent *kNN* query is not issued from the same physical location. This differs from *continuous kNN* queries where multiple queries are broadcasted into the network addressed to the same location. This satisfies our initial claim of mobile data mules being independent of each other i.e. result of a query is not shared with other mobile data mules in the system. The lifetime of these *single-shot KNN* queries is determined by the amount of time the mobile data mule stays at that location. The proposed 3D-KNN algorithm functions in three phases:

Boundary Estimation: This phase handles estimating the search boundary that encompasses the *k* nearest neighbours. The boundary is computed as a sphere with the mobile data mule in the centre.

Pre-Routing: The pre-routing phase handles broadcasting initial request messages within the estimated *kNN* boundary. Sensors employ in-network data aggregation to reduce the amount of information transmitted.

Neighbour Selection: This phase selects the cost-efficient *k* neighbours within the estimated *kNN* boundary. The output of the neighbour selection phase is the cost-efficient set of sensors.

4.4.2 3D-KNN: Boundary Estimation Phase

Definition 4-2: (KNN Boundary Set S): Given a set N of i sensor nodes, location L of mobile data mule, a subset S of j sensor nodes represents the KNN Boundary set where $S \subseteq N$ *and* $|S| \leq |N|$ *such that for each sensor* $s_j \in S$ *and each sensor* $s_i \in N - S$, $DIST(s_j, L) \leq DIST(s_i, L)$ *and* $DIST(s_j, L) \leq KNN\_BOUNDARY$.

The set *N* is the set of all nodes within the sensor network. The subset *S* of *j* sensors that fall within the *kNN* boundary is computed from *N*. The *kNN* boundary determines the perimeter of the data collection area. In short, the radius of the collection area is given by the *KNN_BOUNDARY*. It is within the *kNN* boundary, the search for the *k n*earest neighbours is performed. The subset *S* introduced in definition 2 is depicted as a coloured sphere in Figure 4.6. The straightforward approach to compute the *kNN* boundary is to employ flooding-based techniques (Hedetniemi et al., 1988). By flooding the network with broadcast packets, the mobile data mule will have network information about the entire sensor network which can be further used to determine the *k n*earest neighbours. Though this approach is reliable, it is expensive, as the entire network is flooded with broadcast packets from the mobile data mule. Hence, a more efficient approach is to broadcast packets to a subset of sensors (*S*) within the sensor network. In our model, the initial query *Q* is propagated by the mobile data mule. In our scenario the mobile data mule is the point-of-interest. The 3D-KNN boundary estimation algorithm computes a sensor boundary around the data mules' location. We term our boundary estimation approach "*inside-out*" method i.e. the boundary estimation is performed by the mobile data mule which is the point-of-interest. Hence, the broadcast query originating at the point-of-interest (inside) propagates outwards towards surrounding sensors. This is different from traditional sink approach where the broadcast query originates outside the network (from the sink).

The *kNN* boundary estimation is one of the challenging steps in an infrastructure-less sensor network. Since an assumption of uniform distribution over the entire sensor network is not appropriate, we assume that nodes are uniformly distributed within the computed *kNN* boundary. This assumption aids in accurately computing the number of nodes within the *kNN* boundary. With this assumption we can determine the density of nodes (N_D) within the sensor network. The density of the network is the total number of sensors within the network and not just active sensors. Hence, to determine the *kNN* boundary that encompasses at least s_j sensors, we use (4-1) to compute the surface area covered by s_j sensors such that $\forall s_j \in S\ (S \subseteq N)$ satisfies the condition $loc\ (s_j)$ is within *A* (area covered by the 3D space). *A* is the area of the 3D collection space (collection area) and $loc\ (s_j)$ is the location of the sensor node s_j. Since radio range is represented as a sphere, we use equation (4-2) to compute the radius *R* of the sphere covering at least s_j sensors. The *KNN_BOUNDARY* is given by the radius *R*. Sensors node within the *KNN_BOUNDARY* comprise the set *S*.

$$\boldsymbol{Surface\ Area\ of\ the\ 3D\ Space\ (A)} = \frac{\boldsymbol{Node\ Density\ N_D\ (per\ m^3)}}{\boldsymbol{No\ of\ nodes\ n}} \qquad \textbf{(4-1)}$$

$$\boldsymbol{KNN\_BOUNDARY\ =\ Radius\ R} = \sqrt[3]{\frac{\boldsymbol{Area\ A * 3}}{\boldsymbol{4 * \pi}}} \qquad \textbf{(4-2)}$$

4.4.3 3D-KNN: Pre-Routing Phase

Our key assumption is that no network topology information is required to process the *kNN* query. Hence, node discovery needs to be performed within the computed *kNN* boundary. The pre-routing phase performs the following functions:

- Collect sensor information which includes node location, signal-to-noise ratio (a cumulative function for each hop from mobile data mule to destination sensor node) and hop distance (from mobile data mule).

- Dynamically compute route information to each sensor within the *kNN* boundary
- Compute perimeter nodes that form the boundary of the collection area.

To achieve the above functions, a broadcast message is sent by the mobile data mule with its location *l* and *KNN_BOUNDARY* that determines the radius of the collection area. Each broadcast message sent by the mobile data mule has an associated broadcast-id which is derived from the mobile data mule's MAC address to maintain uniqueness. Each broadcast message is also associated with a time to live (TTL) to avoid delayed arrival of data packets which is typical of real-world communications. The pre-routing phase function is depicted in Figure 4.7. Note that while we have used a single mule representation, this approach can be scaled to multiple mobile data mules. The mobile data mule depicted in Figure 4.7 is represented as the yellow star and sensors as *X*. The sensors highlighted in green identify the border nodes i.e. nodes that form the perimeter of the *kNN* boundary. We use a two-dimensional illustration to represent the working of the pre-routing phase.

A sensor node *X* on receipt of a broadcast message checks if the broadcast message was received from the mobile data mule or other sensors. This operation ensures that nodes that are in one hop distance from the mobile data mule do not use neighbouring nodes to communicate. If the message source is other sensors, the first arriving message is saved and the rest of the messages are discarded. The sensor uses the TTL value of the broadcast message to identify delayed message arrivals.

On receipt of a new broadcast message, the sensor nodes use the information in the message to determine its distance *(D)* from the mobile data mule. This distance *(D)* is used to determine if the sensors are within the *kNN* boundary. Sensors within the *kNN* boundary re-broadcast the message to neighbouring sensors. The re-broadcast message is appended with the node's identification. This information is useful to compute the dynamic route. The route information is used by sensors to determine a reverse path to the mobile data mule and vice-versa. In order to save energy over multi-hop communications, the sensors employ in-network data aggregation. Data aggregation is performed using timers. The timer value is determined as a function of the *kNN* boundary and the node's distance given by:

$$\boldsymbol{Wait\ Timer\ (W_x) = \frac{KNN\_BOUNDARY}{D_x}} \tag{4-3}$$

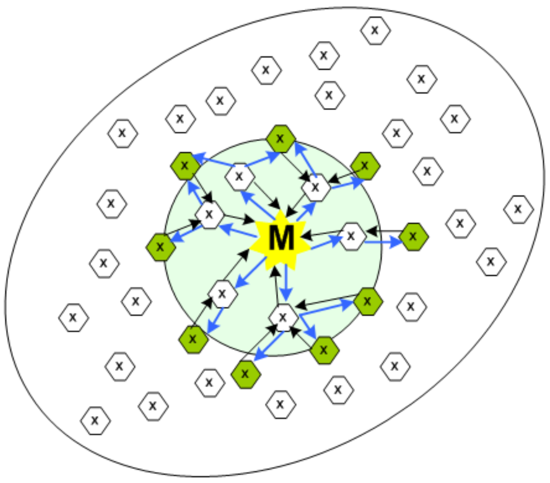


Figure 4.7: Pre-Routing phase - Illustration

Sensor nodes that are outside the *kNN* boundary cease broadcasting messages further into the network. These sensors which are at the border or just outside the *kNN* boundary form the perimeter of the data collection area. The perimeter nodes are shaded nodes in Figure 4.7. These nodes initiate response to the broadcast message (query) by sending their location and signal-to-noise ratio to the neighbouring node through which it received the broadcast message. The same process is performed at each node until the requested information reaches the mobile data mule. Sensors that do not receive any response from neighbours on timer expiry, forward their information back to the mobile data mule. For example, consider the illustration presented in Figure 4.8. For easier illustration, only a section of the collection area is shown. In this case, the sensor node *X1* receives the broadcast message from the mobile data mule and forwards it to the neighbouring nodes *X2, X3, X4*. X1 computes a timer value using (4-3). From (4-3), we can deduce that timer value of *X1* will be greater than the timer values of nodes *X2, X3* and *X4* respectively. The node *X2, X3* and *X4* identifies itself to be outside the *kNN* boundary. Hence, it stops re-broadcasting the message and returns the requested data to node *X1*. Since *X4, X3* and *X2* are at varying distances from the mobile data mule their timer expiry will result in a tiny offset that helps to avoid contention while communicating with *X1*. The boundary estimation and pre-routing phase pseudo code is presented in Figure 4.9 and Figure 4.10. The algorithm is divided into two parts, the mobile data mule part and the sensor node part. The sensor node implementation is developed taking into consideration their low-powered and resource-constrained nature. The 3D-KNN avoids the necessity to run complex algorithms on the sensor nodes. We present a description of the boundary estimation and pre-routing pseudo code employed by the mobile data mule in the paragraph following Figure 4.8. The description of the sensor node pseudo code is presented in the paragraph following Figure 4.10.

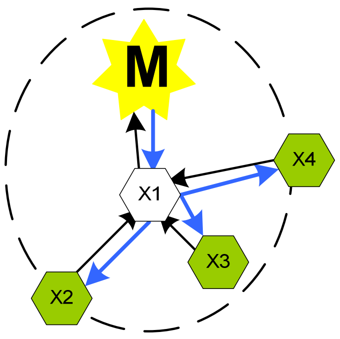


Figure 4.8: Illustration of Pre-Routing

```
3DKNN – Boundary Estimation and Pre-Routing Phase- Mobile Data Mule
Pseudo Code
Input:  Surface Area of the 3D space (A), total number of sensors
(n), l location of mobile data mule
Output: SNR, location and hop information of node
BEGIN
   1  Compute $N_D = N/A_T$
   2  Get k
   3  Compute A such that $A = k/N_D$
   4  Compute $R = \sqrt[3]{\frac{Area\ A*3}{4*\pi}}$
   5  Compute maxKNNDIST = DIST(l ± R)
   6  Send initial broadcast message(msgID, route, TTL, maxKNNDIST,
      l)
   7  Repeat
   8        listen to channel
   9  Until all data received or TTL reached
END
```

Figure 4.9: Boundary Estimation and Pre-Routing Phase- Mobile Data Mule Pseudo Code

Step 1, 2 and 3 is used to compute the *kNN* boundary using sensor network density and required *k*. In step 4, the radius (R) of the sphere enclosing the nearest neighbours is computed. Step 5 computes the maximum distance a sensor can be from the mobile data mule. This value determines the *kNN* boundary. Step 6 performs the operation of preparing and sending the broadcast messages. Step 7, 8 and 9 perform the operation of listening to the channel until all responses from neighbouring nodes are received. This operation expires when the TTL of the broadcast message is reached.

```
3DKNN – Boundary Estimation and Pre-Routing Phase- Sensor Node
Input: Broadcast Message m
Output: SNR, location and hop count
BEGIN
   1  For each broadcast message m
   2        if m not from source and m is received
   3              discard m
   4        else
   5        Compute distance D = DIST(n, l)
   6        If D < maxKNNDIST
   7              set timer = maxKNNDIST / distance
   8              add nodeID to route
   9              hopcount ++
   10             if neighbourList ($nec_i$) available
   11                   forward broadcast message $nec_i$
   12             else
   13                   forward broadcast message
   14             end if
   15       Else
   16             mark perimeter node
   17             hopcount ++
   18             add nodeID to route with a END mark
   19             return location, snr to next node in route
   20       End if
   21 End For
   22 On Timer Expiry
   23       aggregate results of location and snr
   24       return result to next node in route
   25 End
END
```

Figure 4.10: Boundary Estimation and Pre-Routing Phase- Sensor Node Pseudo Code

Step 1 handles new broadcast messages arriving at the sensor node. Step 2 checks if the received broadcast message already exists. Step 3 discards the message if it exists. Step 5 computes the distance of the sensor node from the mobile data mule. Step 6 checks if this distance is within the *kNN* boundary. Step 7 sets a timer for data aggregation. Steps 8 and 9 add the node's identification to the broadcast message and increments the hop counter. Step 10 checks if a neighbour list exists at the sensor node. This is not a requirement for the functioning of the 3D-KNN algorithm. Steps 11 and 13 handle rebroadcasting of the message to neighbouring nodes. If the sensor node is outside the *kNN* boundary, step 16 adds the sensor as a perimeter node. Steps 17 and 18 increment the hop count and adds the node identification to the route part of the broadcast message. Finally step 19 initiates return of data (SNR and the location information). Steps 22, 23 and 24 describe the timer expiry procedure. When the sensor timer expires, it returns aggregated data to its parent node (obtained from the route part of the broadcast message). The parent node is the sensor from which the broadcast message was received. This process is repeated until the mobile data mule is reached.

4.4.4 3D-K NN: Neighbour Selection

The outcome of the pre-routing phase provides the mobile data mule with sensor information within the estimated *kNN* boundary (collection area). To determine the set *K* of *k* sensors, we propose a sensor selection metric based on sensor information collected during the pre-routing stage.

Definition 4-3: (k-Nearest Neighbours): Given a set S of j sensor nodes, location L of mobile data mule, a subset K of k nodes where $K \subseteq S$ *and* $j \leq k$ *such that for each sensor* $s_k \in K$ *and each sensor* $s_j \in S - K$, *KNN-METRIC* $(s_k, L) \leq$ *KNN-METRIC* (s_j, L).

The *KNN-METRIC* is used to map sensors on different planes onto mobile data mule's plane. We term this approach *"plane rotation"* i.e. we virtually rotate sensors on various planes to a single plane based on the *KNN-METRIC* metric.

4.4.4.1 Mapping Technique

The mapping algorithm uses sensor parameters signal-to-noise ratio (SNR) and distance collected during the pre-routing phase to compute a single-valued metric. This single valued metric is used to effectively map sensors on different planes (as depicted in Figure 4.6) to a reference plane based on real-world radio signal propagation characteristics. The reference plane is the plane in which the mobile data mule moves.

This plane is the zero reference plane in which, the value of *z* axis is zero while *x* and *y* change based on mobile data mule's movement pattern. Sensors above the reference plane take a positive value for *z* while sensors below the reference plane take a negative value for *z*. The sensors themselves are calibrated based on the reference point governed by the deployed space. For example, in a building, sensors *z* value represents the level in which they are present.

We introduce the term *"KNN-METRIC"*- a metric computed using sensor nodes characteristics including channel quality, interference caused by obstacles and distance. The distance (*D*) is Euclidian distance computed as the distance between the mobile data mule and the sensor node given by (4). The location of the mobile data mule and the sensor node are given by the coordinates *(X_1, Y_1, Z_1)* and *(X_2, Y_2, Z_2)* respectively. The *KNN-METRIC* applied in sensor mapping is computed using (4-5)

$$\boldsymbol{Distance\ D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2)}} \qquad \textbf{(4-4)}$$

$$\boldsymbol{KNN - METRIC = c * \frac{\alpha * SNR}{\beta * Distance\ (D)}} \qquad \textbf{(4-5)}$$

The SNR value is collected during the pre-routing phase. The SNR for sensors that are more than one hop away is computed as a cumulative sum of SNR values between each hop. The SNR and the distance parameters are inversely proportional i.e. higher SNR represents better channel quality while greater distance increases retransmission reducing reliability. These parameters are taken into consideration as they are good indicators of energy consumed during communication. For example, with poor SNR the sensor needs to increase its radio power to successfully complete the data transfer. The same can be observed at the mobile data mule even though energy is not a primary concern for the mobile data mule. With the aim to reduce the energy-consumed during communication, it is important to select a set of sensors that are energy-efficient to communicate. The value *c* is a constant and *α*, *β* are pre-assigned weights. The weights are assigned by the sink based on application requirements. To compute a value for the constant *c,* we assume ideal case (computed from simulation outcomes) values for SNR and distance such that *KNN-METRIC* = 1. The ideal case values and hence the value of the constant *c* can be changed for different deployment environments. We determine *c* from (4-6).

$$\boldsymbol{c = 1 * \frac{0.6 \ * 100}{0.4 * 60} = 2.5} \qquad \textbf{(4-6)}$$

The *KNN-METRIC* provides a value that is used to map sensors around the data mule based on sensor characteristics. At the end of the mapping stage, the *k* cost-efficient sensor neighbours are computed by using sorting techniques, based on the computed *KNN-METRIC*. Current work discussed in Chapter 2 dealing with *kNN* queries assumes two-dimensional sensor distribution with no consideration for communication channel characteristics. They only rely on Euclidian distance for computing nearest neighbours. Though the extension of distance-based approaches is straight-forward in 3D planes, we argue it is not necessarily energy-efficient. For example, consider a sensor *A* at a distance D_1 in a plane below the mobile data mule and sensor *B* at distance D_2 in the same plane as the mobile data mule such that $D_2 > D_1$. Given the channel quality (computed using SNR) for *B* is better than *A*, the two-dimensional distance-based approach would neglect the channel quality parameter influenced by plane separation. It would select *A* as the nearest neighbour. The proposed *KNN-METRIC* based sensor mapping approach selects sensors as a function of both distance and channel quality (SNR), hence selecting sensor *B* as the energy-efficient nearest neighbour. Moreover, current approaches (Yao et al., 2009, Yao et al., 2006, Winter et al., 2005, Wu et al., 2007) consider error-free communication channel which does not hold true in real-world scenarios. Our approach can be easily extended to incorporate additional sensor metrics that can improve cost-efficiency of data collection. For example, introducing the sensor residual energy parameters into the *KNN-METRIC* will result in a set of sensors with the following characteristics: 1) distance from the mobile data mule is least, 2) has good communication channel quality (SNR) and 3) sufficient energy to successfully complete data collection. Further, the selection parameters can be correlated to improve sensor selection accuracy i.e. a correlation between the SNR and distance metric can be used to dynamically recompute the weights α and β respectively. The algorithm for the *k*-Nearest Neighbour selection phase is presented in Figure 4.11.

```
3DKNN – k-Nearest Neighbour selection
Require: S subset of nodes, k required number of neighbours, mobile
data mule reference plane and location P (X,Y,Z) and for each node I
in S its SNR and location X, Y, Z
BEGIN
   1  For each node I in S
   2        determine current plane
   3        compute Distance D with reference to P
   4        I_D = $\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2}$
   5        compute KNN-METRIC = 2.5 * (0.4 * I_SNR / 0.6 * I_D)
   6        add KNN-METRIC to S
   7  End For
   8  Sort S based on KNN-METRIC
   9  if k < S
   10       return top k results from S
   11 end if
END
```

Figure 4.11: *k*-Nearest Neighbour Selection- Pseudo Code

4.4.5 3D-KNN: Neighbour Prediction

The *k* nearest neighbours computed during the selection phase form the set of energy-efficient sensors within the collection area. To further improve the energy-efficiency of the data collection process, we propose an extension to the 3D-KNN algorithm. We propose the introduction of neighbour prediction. The neighbour prediction technique further filters the set *K* to identify the most cost-efficient set of sensor neighbours.

Definition 4-4: (Predicted Set of Nearest Neighbours P_T): The predicted set of nearest neighbours P is a set of p nodes where $P \subseteq K$ and $p \leq k$ such that for every sensor $s_p \in P$ and every sensor $s_k \in K-P$, DIST (s_p, L_T) $\leq$ DIST (s_k, L_{Ti}) where i = 1 to x and hopcount (s_p, L_T) > 2.

L_T represents the mobile data mule's current location at time *T* and L_{Ti} represents future locations at time $T_1, T_2 ... T_x$. The *hopcount* represents the number of hops between the sensor node and the mobile data mule. A node s_p is said to belong to the set *P* if and only if its distance *D* from the data mule's current location L_T at time *T* is the less than its distances $D_1, D_2, D_3 ... D_x$ at time $T_1, T_2, T_3 .. T_x$. We follow the following rules to compute the set *P* from set *K*.

1) Add all sensor s_k of set *K* to set *P*

2) Consider a sensor s_p as a future neighbour if and only if

    a. Hop distance between s_p and mobile data mule's current location is greater than two

    b. Hop distance between s_p and mobile data mule's future locations at time $T_1, T_2, T_3 ... T_x$ is less than two hop distance.

The above conditions can be satisfied as it is feasible to estimate the maximum hop count between the mobile data mule and the sensor node at time $T_1, T_2, T_3 .. T_x$ using the sensor's radio range and computed distance. For example, if the radio range is 30 meters and distance is 50 meters then the sensor is at least 1 hop away from the mobile data mule. Further, we use a hop count value of 2 as it is less likely that a sensor at a future location would be an energy-efficient choice if it is more than two hops away. Moreover, the probability of reaching a sensor that is two hops away is much less. The radio range is given by the maximum distance between the sensor node and the mobile data mule at which direct communication is possible.

For example, consider the illustration in Figure 4.12. The mobile data mule is identified by the yellow coloured star and the sensors are identified by *X*. The trajectory provides the path of the mobile data mule. The two locations represented by the yellow star are the locations *T* and T_1 where the mobile data mule stops to compute nearest neighbours.

We consider two cases identified by the node in green and red respectively. In case 1, the sensor node *X* in *green* is closer at time T_1 rather than at time T. Hence, the prediction algorithm selects it as a candidate for data collection at time T_1. In case 2, the previously stated rules are applied. The sensor node *X* in *red* is at the same hop distance from locations *T* and T_1. Assume the distance between the sensor node and the mobile data mule at time *T* and T_1 are 90 and 80 respectively. In terms of distance, the sensor is closer at time T_1 but the mobile data mule would still require two intermediate hops. Hence, it would be more advantageous to collect data from that sensor at time *T*. The nodes that satisfy the two rules are added to the mobile data mule's predicted next hop collection set K_{LTx} where LT_x represents future locations and time combinations.

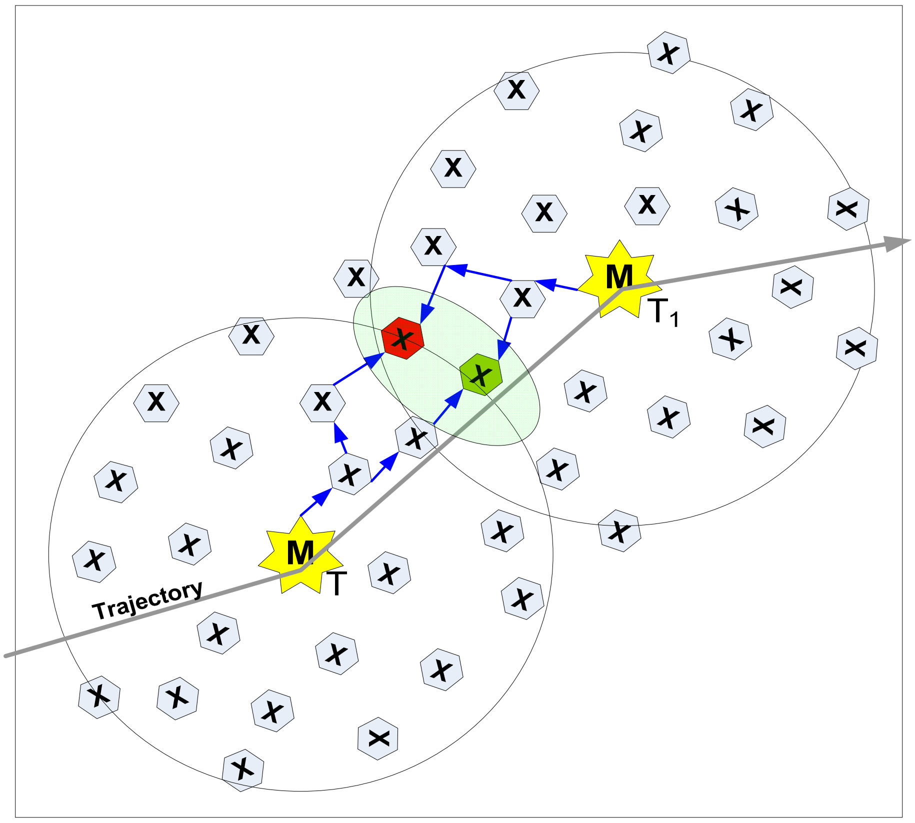


Figure 4.12: Nearest Neighbour Prediction- An Illustrative Example

The set P is an optimised energy-efficient set of sensor nodes from which data is collected by spending least overall energy. The prediction algorithm computes distances between sensor locations (obtained from *kNN* query) and the mobile data mule's future locations. Hence, actual SNR values may not be available. Once the mobile data mule arrives at location *L* at time T_x, it piggy-backs the list of nodes whose distances were pre-computed with the new *kNN* broadcast message. Each node receiving the message checks its node-id against the node list in the broadcast message. If the node-id does not exist, the *kNN* pre-routing phase continues by broadcasting the message to other neighbours. If the node-id matches the sensor's id, it checks for any neighbours that are part of the list. In scenarios where a neighbour list does not exist, the sensor node re-broadcasts the message and sets a timer based on the *kNN* boundary. If a neighbour exists, it forwards the message to the selected neighbour. On timer expiry, the aggregated data is returned to the mobile data mule. The 3D-KNN neighbour prediction algorithm is presented in Figure 4.13. A description of the algorithm is presented in the following paragraph.

Step 1 iterates through each node s_k in the set K which comprises the selected k nearest neighbours. Step 2 computes the hop count between s_k and the mobile data mule. Step 3 checks if the hop count is greater than 2. The nodes that have hop count greater than 2 are considered to be possible future neighbours. Step 4 and Step 5 computes the distance between the sensor node and the mobile data mule. The *DIST* function computes the Euclidian distance. Step 6 iterates through subsequent locations (*L*) of the mobile data mule given by the vector $\overrightarrow{\text{mov}}$. Step 7 computes the distance between the mobile data mule's subsequent location and sensor s_k. Step 8 is used to determine the hop count between the mobile data mule's future location and sensor *k*. Step 9-11 uses the function minimum to find the least of the two computed distance metrics. Step 13 checks if the sensor node's distance from the mobile data mule's current location is the minimum compared to the mobile data mule's subsequent locations. Step 15 adds the node to the current location's predicted set if the condition is true. Step 17 appends the sensor *k* and the mobile data mule's future location (*location)* to the predicted set P_{Ti}.

```
3DKNN – Energy Efficient Set of Nearest Neighbour Prediction P
Input: K set of $s_k$ nearest nodes, mobile data mule future Locations
(L) in a Vector $\overrightarrow{mov}$, mobile data mule current location loc
Output: Predicted set P
BEGIN
   1  for each node $s_k$ in K
   2        h = hopcount(k, data mule)
   3        if h > 2 then
   4              current_metric= DIST ($s_k$, loc)
   5              min_metric = current_metric
   6              for each L in $\overrightarrow{mov}$
   7                    $k_{DL}$ = DIST ($s_k$, L)
   8                    if ($k_{DL}$ / radioRange <= maxHop)
   9                          min_metric = MINIMUM (current_metric,
                                                  min_metric)
   10                         set location = L
   11                   end if
   12             end for
   13       end if
   14       if (min_metric = current_metric)
   15             Add node to P
   16       else
   17             Add node to $P_{Ti}$ (location, $s_k$)
   18       end if
   19 end for
END
```

Figure 4.13: 3D-KNN Neighbour Prediction- Pseudo Code

4.5 Summary

In this chapter we have applied the theory of nearest neighbour search in our mobile data mule-based sensor data collection. We proposed a novel *k*-Nearest Neighbour based multi-hop sensor data collection algorithm namely, 3D-KNN. The 3D-KNN has the capability to account for channel quality, distance and sensor parameters in a three-dimensional environment. Our proposed 3D-KNN algorithm is an extension to the mobile data mule-based data collection architecture presented in Chapter 3. More specifically, we focused on extending our data collection approach to a connection-less broadcast-based sensor network. The 3D-KNN approach incorporates algorithms to discover, collect and deliver sensor data on-the-fly. The 3D-KNN algorithm employed on the mobile data mule has the following features:

a. Employs energy-efficient techniques for sensor node discovery within the scope of a connection-less sensor network

b. Employs a sensor mapping algorithm that maps sensors in different planes to a reference plane using the proposed *KNN-METRIC*

c. Compute and process kNN queries on-the-fly

d. Incorporates sensor prediction to further improve the energy efficiency of the sensor data collection process.

5

R-CS: Modelling Smart Spaces using Smart Sensing

5.1 Introduction

Space is defined by Dictionary.com as "*the unlimited or incalculably great three-dimensional realm or expanse in which all material objects are located and all events occur.*[13]"A space may be an enclosed area e.g. *meeting room,* or an open area e.g. *courtyard*. *Smart space* is the term used to represent spaces embedded with computing infrastructure (Satyanarayanan, 2002). Embedding physical spaces with computing infrastructure enables sensing and controlling of one world by the other. This computing embedded physical space can be visualised as a fully connected information space where data flows from one point to another (Kumar, 2005). Applications that work within smart spaces depend on contextual information to perform various tasks. These smart space applications are also called *context-aware* applications (Loke, 2006, Hähner et al., 2004). Context is defined as "*that which surrounds, and gives meaning to, something else*[14]". Smart spaces are reactive and proactive i.e. they have the capability to react to environmental changes by monitoring and analysing the data generated within the physical space. For example, controlling the air-conditioning system depending on occupants profiles or controlling the software on the occupant's mobile phone depending on his/her location.

The low-level data required to compute contextual information within smart spaces originate from individual or multiple software/hardware-based sensors. Sensor

[13] Space: http://dictionary.reference.com/browse/space
[14] Context: The Free On-line Dictionary of Computing

networks play a vital role in smart spaces as they provide sensing capabilities. Our smart spaces philosophy is driven by the availability of sensor data which is used to infer high level context. For example, using data collected from a *temperature sensor*, high level context like "*comfortable temperature*" can be reasoned.

In this chapter, we present a situation-based context reasoning approach. The situation-based context model is used to reason about real-world situations based on available contextual information. Situation awareness is a flavour of context-aware computing that allows applications to adapt to changing environmental situations. For example, a situation *meeting* in the real-world can be defined in the virtual world as a collection of context attributes *light status*, *noise* and *projector status* (Padovitz, 2006). The values of the context attributes (at a specific time), may be used to infer the occurrence of the situation *meeting*. The possible values for the occurrence of the situation *meeting* can be *bright* for light, *5* for noise (noise ranges from 0-silent to 5-noisy) and *off* for projector status. The values *bright*, *5* and *off* are computed from hardware/software sensors.

In previous chapters of this thesis we proposed techniques for cost-efficient sensor data collection. The data collection framework, namely, sGaRuDa, facilitates data collection from sensors embedded within smart spaces using mobile devices. We term our data collection approach *smart sensing*. The data collected from sensors is dynamic i.e. it changes over time. Further, the widespread adoption of sensor networks leads to more sensor data generated, stored, processed and used. To address the dynamic characteristic of sensor data, we propose a dynamic situation modelling approach that composes situation definitions at run-time, based on available sensor data. Proposing/exploring a new context model in itself is a big challenge. Hence, in this chapter, we choose to extend an existing situation-based context modelling framework, namely, *Context Spaces* (Padovitz, 2006). We propose extensions to Context Spaces to enable situation composition on-the-run. This chapter is divided into two main sections. The first section provides an overview of Context Spaces. The second section presents the proposed Context Spaces extensions published in the following papers (Jayaraman et al., 2008c, Jayaraman et al., 2009b, Jayaraman et al., 2009a).

5.2 Context-Situation Pyramid

Our notion of a situational context model can be best captured by the context situation pyramid (Padovitz, 2006, Padovitz et al., 2004) depicted in Figure 5.1.

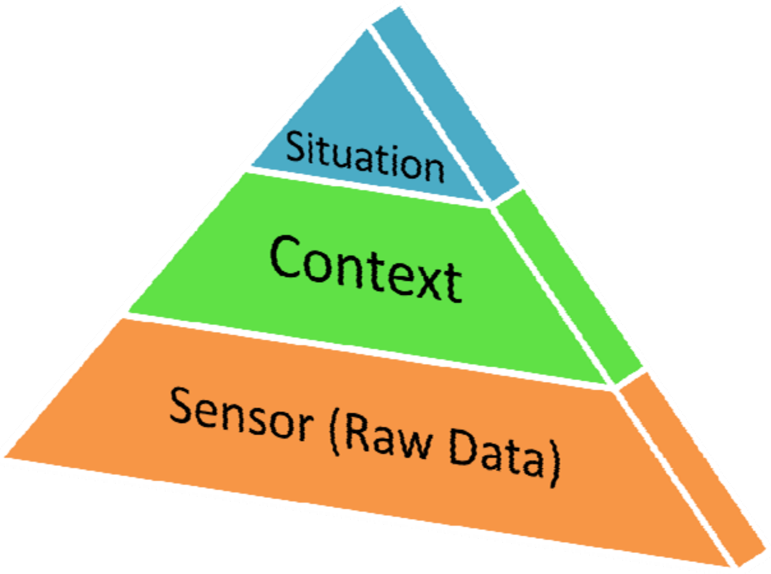


Figure 5.1: Context-Situation Pyramid

Sensor (Raw Data): This is the lowest level of the context model representing the raw data collected from sensory sources. Data at this level does not convey any meaning. e.g.: a temperature sensor reporting a value 24.5^0 C.

Context: The middle level represents a collection of information, facts and predictions combined with algorithms to interpret and process context. The contextual information can be used to represent the state of the system. The state of the system changes as context information changes over time. The context is used to compute a constrained view of the physical world.

Situation: The meta-level notion above context which is derived by analysing available contextual information. A situation is a more detailed view of the physical world situation inferred by reasoning about context.

5.3 Theory of Context Spaces

The theory of Context Spaces (Padovitz, 2006, Padovitz et al., 2004, Padovitz et al., 2005) has been proposed on the philosophy of the context situation pyramid. Context Spaces use state-space models and geometrical metaphors to represent situations and system states in a multi-dimensional space. Context spaces provide the following definitions:

Context Attribute (a): A context attribute is any type of data with a corresponding value. A virtual or physical sensor determines the value of the context attribute, e.g. temperature is a context attribute. A context attribute is a virtual world representation of a hardware/software sensor available within the smart space.

Application Space ($\Re$): A universal set of disclosure (UOD) contains all possible context attributes with associated value ranges for any specific application domain.

Context State (C^T = (a_i^T)): A context state is a set of context attributes and their corresponding values at time *T*. For example, the context state at time *T* might comprise context attributes temperature and humidity with sensor values 20^0 C and 70% respectively.

Situation Space (S =A_i^j)): A situation space comprises a set of context attributes and their corresponding acceptable attribute region values. The context attributes are used to define the situation such that it best represents its real-world counterpart. For example, a situation *meeting* can be defined using context attributes *light* and *noise* with corresponding attribute region values *(on, mild)* and *(5 - 8)* respectively.

Attribute Region (A_i^j): The attribute region defines the possible values the corresponding context attribute a_i can have within the defined situation. For example, the attribute region values for the context attribute *noise* in a situation *meeting* can range from 5 to 8.

Figure 5.2 illustrates the Context Spaces approach. In this example, the situation is defined using three context attributes (***a_1, a_2, a_3***). Each context attribute corresponds to a specific dimension. The context state at time t_1 is represented as C^1 and at time t_2 as C^2

respectively. Situation reasoning is performed by comparing the context state at time *T* against pre-defined situation definitions.

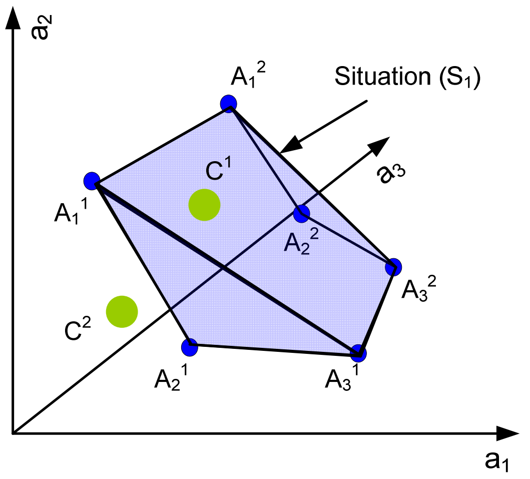


Figure 5.2: Three Dimensional (3 context attributes) illustration of Situations and Context States

In Figure 5.2, at time t_1 the context state $C^1$is within the situation definition S_1 while at time t_2, the context state C^2 does not conform to any pre-defined situations. For example, suppose a situation *presentation* is defined over the context attribute *projector status* with corresponding attribute region value *on*. At time *T*, if the context state value of the *projector* (computed from sensor attached to the projector) is *on,* then the context state matches the situation definition. This information is used to reason the occurrence of the situation *presentation*. Context spaces proposes Context Spaces Fusion (ConSpaF) (Padovitz, 2006), an algorithm to fuse sensor data with context. This algorithm extends the geometry-based state-space reasoning approach by incorporating additional heuristics that increase Context Spaces ability to reason under uncertainty. We focus on the following two heuristics:

1) Incorporating sensor error caused by sensor inaccuracies
2) Relevance and contribution functions that identify importance of context attributes within a situation.

The first heuristic identified allows the Context Spaces model to adapt to sensor inaccuracies. The second heuristic has two parts, the relevance and the contribution function. The relevance determines the importance of a context attribute within the situation i.e. is the attribute important enough to determine the outcome of the reasoned situation? Context Spaces uses weights to denote the relevance of a context attribute within the situation. The contribution function is used to determine if the context attribute's current context state value falls within the situation's attribute region definition. The contribution function computes a contribution value *(0 or 1)* for a context attribute a_i over the region A_i^j defined in the situation space S_j. To fuse context attributes' relevancies and contributions, Context Spaces propose the use of multi-attribute utility theory (MAUT) (Winterfeld et al., 1986). MAUT combines different contributions into a single utility value. The single utility value, computed by fusing multiple context attributes, determines the certainty of situation being inferred. This certainty is termed *confidence*.

Example: Consider two contextual attributes a_1 and a_2, a situation S_1 with the attribute region values A_1 and A_2. Let the weights w_1, w_2 and contributions c_1, c_2 represent the context attribute's importance and contribution in situation S_1. The weights are any values between 0 and 1 such that $w_1 + w_2 = 1$. The MAUT based utility, *confidence* for situation S_1, is computed using $w_1 * c_1 + w_2 * c_2$.

In the next section, we present the proposed extensions to Context Spaces.

5.4 Proposed Extensions to Context Spaces

The first proposed extension to Context Spaces is the introduction of partitioned situation spaces. The situation space is partitioned based on the relevance of the context attributes. At runtime, a temporal situation is computed from the partitioned situation space based on available contextual information (context state) within the smart space.

The second extension to Context Spaces is the use of flexible attribute regions. The attribute region determines the contribution factor of a corresponding attribute. This contribution factor influences the overall confidence in the situation being reasoned. The proposed extension adds flexibility to the crisp attribute region definition used in Context Spaces. Additionally, we introduce sensor quality metrics, namely, data freshness in the error computation algorithm of Context Spaces.

The third proposed extension to Context Spaces is the introduction of hierarchical situation space definitions. The hierarchical situation space provides a way to define relations between attribute regions within a situation.

We call our proposed system R-CS depicting the use of ranking and partitioning in Context Spaces. A detailed description of the proposed extensions is presented in the following sections.

5.5 Dynamic Situation Modelling

The situation space in Context Spaces (CS) (Padovitz, 2006) is defined using a fixed set of context attributes (a_1, a_2… a_n) and corresponding attribute region values (A_1, $A_{2\dots}$ A_n). The relevance of each context attribute within the situation is given by weight w_i such that

$$\sum_{i=0}^{n} w_i = 1 \quad 1 \leq i \leq n \tag{5-1}$$

We argue that in smart spaces, the application must cope with changing contextual information. The fixed situation space definition of Context Spaces makes it difficult to incorporate new context attributes on-the-fly. The dynamic situation composition arises from our argument that fixed situation-based reasoning may not cater to smart space applications where new context information appears and disappears over time.

We define a generic situation space called the universal situation space (S_U) that consists of all possible context attributes and their corresponding attribute region values. The universal situation space is an extension to Context Spaces situation space definition. The difference is that universal situation space has all possible context attributes that may be used to infer the situation while Context Spaces (CS) situation space contains a restricted set of the most relevant context attributes. For example, suppose a situation *meeting* is defined in CS over the highly relevant context attributes *light*, *people,* and *noise*, the same situation is defined in R-CS over the range of context attributes *light*, *people*, *noise*, *temperature*, *door status, humidity, floor pressure (sensor),* etc. Some of the context attributes defined in the universal situation space may not have high relevance but may be used to determine the situation outcome. E.g. a possible reason for the situation car accident could be acute stomach pain for the person driving the car. This possibility is rare but cannot be excluded. The universal situation space can be considered the universal set of all possible context attributes that represent the situation. The proposed R-CS approach does not use the universal situation space during the reasoning process as opposed to CS approach. A temporal situation is composed by selecting context attributes that are available within the context state and has a corresponding attribute region defined in the universal situation space definition. The temporal situation space *(S_T)* definition changes over time based on the context state value. For example, suppose at time *T*, the context state value for the context attributes *light*, *temperature* and *humidity* are only available, the temporal situation space is composed with attribute region definitions corresponding to *light*, *temperature* and *humidity*.

The temporal situation space computed from the universal situation is used during the reasoning process. The context attribute's relevance given by weight w_i is defined as a value between a lower and an upper boundary. The lower and upper boundary is application specific, e.g., a value 0 to denote least relevance and a value 10 to denote high relevance. At runtime, when the temporal situation space (S_T) is computed, the weights associated with the context attributes are normalised.

Definition 5-1: Universal Situation Space (S_U) – A universal situation space consists of all possible sets of context attributes and corresponding attribute region values with their associated relevance (importance).

$$\mathbf{S_U} = \{\mathbf{w_1 A_1}, \mathbf{w_2 A_2} \dots \mathbf{w_n A_n}\} \tag{5-2}$$

$$where\ min \leq w_i \leq \max and\ 1 \leq i \leq n$$

Definition 5-2: Temporal Situation Space (S_T) – A temporal situation space at time T represents the real world situation that has m context attributes derived from the universal situation space S_U that matches the context state at time T.

$$\mathbf{S_T} = \{\mathbf{w_1 A_1}, \mathbf{w_2 A_2} \dots \mathbf{w_m A_m}\} \tag{5-3}$$

$$\boldsymbol{such\ that\ m < n, \sum_{i=0}^{m} w_i = 1,\ 1 \leq i \leq n\ and\ S_T \subseteq S_U}$$

The temporal situation space S_T provides the situation definition at time T. It is a subset of the universal situation space S_U . The context attributes of S_T are derived from the universal situation space. The temporal situation space changes as context state evolves. The use of a universal situation space allows context-aware applications to define all possible context attributes that best describe the situation. Our proposed dynamic temporal situation composition is illustrated in Figure 5.3.

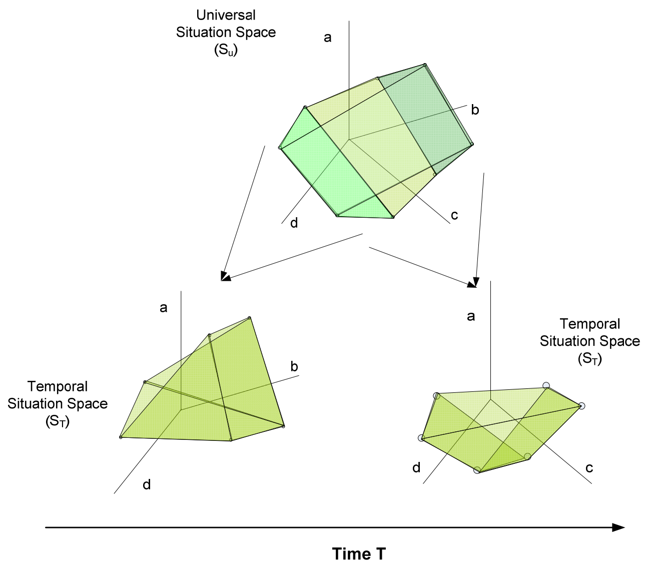


Figure 5.3: Dynamic Situation Composition - Illustrative Example

For example, consider a situation definition with attributes *a, b, c* and *d*. Suppose, at time *T*, if the context state value for the context attribute *c* is unavailable, CS considers the contribution to be 0 hence influencing the overall confidence computation. This is attributed to the fact that CS works on fixed situation definitions. The proposed temporal situation will dynamically compose a temporal situation based on available context state values for context attributes *a*, *b* and *d*. This is illustrated in Figure 5.3 (bottom left) using spatial representation. The temporal situation adapts to the smart environment by computing a situation with a set of context attributes that best represents the real-world situation.

The universal situation space S_U comprises n number of context attributes and corresponding attribute region values. To compute the temporal situation space, a straightforward though expensive approach is to use brute force, in other words, search all available context attributes defined within the universal situation space. As the number of context attributes increase the search complexity increases. Hence, to handle this problem, we propose a partitioned universal situation space technique. The partitioned approach divides the universal situation space into multiple partitions based on the context attribute's relevance.

5.5.1 Situation Partition

Relevance given by weight (w_i) determines the importance of the context attribute in the universal situation space. The universal situation space is partitioned over the context attribute's relevance. We use a partition function to determine the context attribute's partition membership. The partition function is defined as follows:

Definition 5-3: A partition function determines the partition membership of a context attribute a_i with associated relevance given by w_i.

The partition function provides an approach to split the universal situation space into multiple partitions. The partition function is defined as a set of conditions δ over the context attribute's relevance. For example, suppose a situation S_1 is defined using context attributes a_1, a_2 and a_3 and associated relevance (w_i) *55*, *20* and *60* (*0* identifying least significant and *100* identifying maximum significant). The partition function defined with the conditions (δ) $50 < w_i \leq 100$ and $0 < w_i \leq 50$ will partition the universal situation space into two partitions, *partition 1* comprising a_1, a_3 and *partition 2* comprising a_2.

The temporal situation space is computed by initially considering the context attributes in the first partition. The confidence for the situation being reasoned is computed using the temporal situation space definition. If the computed confidence is below the confidence threshold (defined as the minimum confidence value used to infer the occurrence of the situation), the search continues into subsequent partitions. Hence, the temporal situation space shrinks and grows dynamically. The pseudo code to partition the universal situation space is presented in Figure 5.4.

Step 1 iterates through the attribute regions within the situation space. Step 2 and 3 checks if the relevance associated with the context attribute region satisfies the conditions defined in δ. Step 4 adds the partition number to the attribute region defined in the universal situation space.

```
Pseudo Code: Universal Situation Space Partition
Input: Universal Situation Space ($S_U$), Partition Condition $\delta$
Output: Partitioned Situation Space
Begin
   1. for each $w_i \in A_i$ in $S_U$
   2.       for each condition C in $\delta$
   3.             if $w_i$ satisfies C
   4.                   add partition number to $A_i \in S_U$
   5.             end if
   6.       end for
   7. end for
End
```

Figure 5.4: Universal Situation Space Partition - Pseudo Code

The condition for the partition function is determined by the application designer while its value can be improved by learning reasoning outcomes at runtime. Genetic algorithm (Mitchell, 1998) based techniques may be employed to aid this process. The reason we chose a partitioned situation space approach is to reduce the complexity in iterating through all context attributes defined within the situation when the required confidence can be reached by considering some partitions. The confidence threshold is application defined constant and can be varied. The higher the threshold value the higher the confidence in the inferred situation.

To further improve the reasoning ability of the R-CS approach, we propose a weight (relevance) re-distribution technique. The weight re-distribution recomputes weights among context attributes within the temporal situation space. The proposed weight re-distribution approach is applicable in cases where situations being reasoned cannot co-exist.

Definition 5-4: Orthogonal Situations (Padovitz, 2006): Situation spaces S_i and S_j are orthogonal denoted by $S_i \neq S_j$ if they do not overlap.

Orthogonal situations cannot co-exist i.e. they cannot occur in parallel. Orthogonal situations S_i and S_j may share context attributes but the context state value of the same context attribute will not satisfy (contain) respective context attribute regions (region of acceptable values) in S_i and S_j at the same time. The *contain* operation refers to the context state value of a context attribute falling within the respective attribute region in S_i. For example, suppose two situations are *studying* and *presentation* in a smart room. The situation *presentation* would require the projector to be turned on (attribute region) while the situation *studying* may not require the projector (*off*). In this case, the situations share the context attribute *projector* but the context state value for projector at time T which can either be *on or off* satisfies only one situation. Given that a *studying* activity is occurring, the situation *presentation* cannot occur.

The use of dynamic weight re-computation is to reduce the relevance of context attributes whose context state value satisfies (contained) respective attribute regions in orthogonal situations (cannot co-exist). For example, suppose situations *running* and *walking* are defined over the context attribute *speed* with corresponding attribute region values *1 to 3 km/h* and *3 to 6 km/h*. At time T the context state value for the attribute *speed* is 3. Given, the situations are orthogonal the context state value satisfies attribute regions of both situations. Going by definition 4 this is not possible as orthogonal situations do not overlap. Since the overlap influences the confidence computation (equally satisfying both situations), the proposed weight re-computation aims to reduce the effect of overlapping context state values for orthogonal situations. The weight reduction factor v is computed using:

$$\boldsymbol{v} = \frac{\boldsymbol{Max}\ (\boldsymbol{w_i}) - \boldsymbol{Min}\ (\boldsymbol{w_i})}{\boldsymbol{Max}\ (\boldsymbol{w_i}) + \boldsymbol{Min}\ (\boldsymbol{w_i})} \qquad (5\text{-}4)$$

w_i denotes the relevance of the context attributes in the temporal situation space (S_T). The *max* and the *min* functions compute the maximum and minimum weights of the available context attributes. The weights assigned to the context attribute in the temporal situation space S_T is derived from the universal situation space S_U but, the re-computation of weights are performed only within the temporal situation space. The re-computed weight is not stored in the universal situation space. Hence, the original weight distribution of context attributes in S_U is preserved. Further, this approach ensures that outcome of the temporal situation space weight re-computation adapts to changing context states.

For example, consider two situations $S_1$and S_2 with the context attributes regions A_1^1, A_2^1, A_3^1 and A_1^2, A_2^2, A_3^2 with respective relevance (weights) w_1^1, w_2^1, w_3^1 and w_1^2, w_2^2, w_3^2 are defined as orthogonal situations. Suppose the context state C^T for the shared context attributes a_1^1 and a_1^2 satisfies the corresponding attribute regions A_1^1 and A_2^1. The outcome of the reasoned situations will be influenced by the relevance of the shared context attribute. Since the situations are orthogonal, we argue that by reducing the relevance of shared context attribute, the reasoning outcome can be improved. Hence, we compute the value v using the weights w_1^1, w_2^1, w_3^1 and w_1^2, w_2^2, w_3^2. The value v is then added to the context attributes that do not have overlapping context states (satisfying attribute regions of both situations).

The pseudo code for the temporal situation space computation and the weight re-computation is presented in Figure 5.5 and Figure 5.6. The use of partition introduces flexibility by which the application developer can define a range of context attributes based on historic information. In most cases, context attributes with the highest relevance determine the situation outcome. In cases when the highly relevant context attribute does not produce sufficient confidence, other partitions which are less relevant are explored. The paragraph following Figure 5.5 gives a description of the temporal situation space computation algorithm.

```
Pseudo Code: Temporal Situation Space Composition
Input: Universal Situation Space $S_U$
Output: Temporal Situation Space $S_T$
Begin
   1. Declare temporal Situation $S_T$
   2. for each partition p in $S_U$
   3.       for each $A_i$ in p
   4.             if Context State $C^T$ of $a_i$ satisfies $A_i$
                        add region to $S_T$
   5.             end if
   6.       end for
   7.       computeConfidence($S_T$)
   8.       if confidence > threshold
   9.             break
   10.            end if
   11.      end for
   12.       return $S_T$
End
```

Figure 5.5: Situation partition and temporal situation space - Pseudo Code

Step 1 declares a temporary situation space. Step 2 iterates through each partition of the universal situation space S_U. Step 3, 4 and 5 add the context attribute and the corresponding region to the temporal situation space if a matching context state value is found. Step 7 computes the confidence for the current situation being reasoned using the temporal situation space. Step 8 checks if the computed confidence has reached the required threshold. If the threshold value is not reached, the temporal situation space expands into the next partition. This process is repeated till the required confidence is reached.

```
Pseudo Code: Weight Re-Computation
Inputs: Temporal Situation Space S, Temporal Situations SS
Begin
   1. for each situation S' in SS
   2.       if S and S' are orthogonal
   3.             for each attribute-region $A_i$ in S
   4.                   for each attribute-region $A_j$ in S
   5.                         if compare($A_i$ , $A_j$ )
   6.                               overlap ($A_i$)= true
   7.                         end if
   8.                   end for
   9.             end for
   10.            end if
   11.            compute $v = \frac{\text{Max}(w_i) - \text{Min}(w_i)}{\text{Max}(w_i) + \text{Min}(w_i)}$
   12.            for each $A_i$  in S
   13.            if (overlap($A_i$) = false))
   14.                  $w_i = w_i + v$
   15.                  update $w_i$ for $A_i$
   16.            end if
   17.      end for
   18.      return S
End
```

Figure 5.6: Temporal Situation Weight Re-Computation - Pseudo Code

We present a description of the weight re-computation algorithm depicted in Figure 5.6. Step 1 and 2 determine if the situations *S* and *S'* are orthogonal (cannot co-exist). Steps 3 -10 iterate through every attribute region of the situation spaces *S* and *S'* to check if the attribute regions overlap. The *compare* function in step 5 checks if the current context state value for a corresponding context attribute overlaps i.e. does the current context state value lies within the attribute regions range of the context attribute. Step 11 computes the reduction factor *v* taking into consideration the relevance (weights) associated with the context attribute regions in the situation *S*. Step 12-16 re-computes the weights of each context attribute region in Situation *S* that are not overlapping. Step 18 returns the new temporal situation space *S*.

5.5.2 Situation Composition using Dynamically Discovered Context Attributes

One of the key challenges of context-aware systems is to manage uncertainty. Uncertainty is the result of incomplete or inadequate information to reason about context. Uncertainty can be reduced by discovering additional evidences (contextual information) that help the reasoning process. In the previous section, we proposed temporal situation spaces that best represent the current situation based on available contextual information. As an extension to the partitioned situation spaces, we consider the problem of introducing dynamically discovered context into the situation space definition at runtime. Dynamically discovered context is defined as information that is available within the smart space situation at time *T* with no corresponding attribute region definition in the universal situation space. The challenge involved in such scenarios is:

1. Computing the relevance of the newly discovered attribute in the current situation
2. Computing the attribute region, also called region of acceptable values, used to compute the contribution of the context attribute

For example, suppose the situation *presentation* is defined over the context attributes a_1, a_2 and a_3. At time *T*, if a new context attribute a_4 is discovered within the smart space, the challenge is to determine the attribute region A_4 and relevance w_4 of the context attribute. We investigate a theoretical solution to this problem using situation relationships defined in Context Spaces (Padovitz, 2006).

Context Spaces defines situation relations *containment*, *equi-containment and partial containment.* We provide a definition of each situation relationship. We refer to the two situations as *S*, *S'* and the corresponding attribute regions as A_i, A_j' where $1 \leq i \leq n$ and $1 \leq j \leq m$.

Containment: A situation S is contained in S', iff for each attribute region (A_i) in S there is a corresponding attribute region (A_j') in S' such that $A_i \subseteq A_j'$

Equi-Containment: Two Situations S and S' are equi-contained, iff $A_i = A_j'$

Partial Containment: Two situations S and S' are partially-contained iff S and S' are not contained and some attribute regions in S are defined over the same region of acceptable value in S'.

For example, suppose a *song* is part of an *album*. The *song* is defined over the context attributes *singer* and the album is defined over the context attributes *singer* and *composer.* The *song* is contained in *album* since its region of acceptable values (*singer*) is contained within its respective region in *album*. If we remove the context attribute *composer* from *album*, the situations will become equi-contained. Alternatively, by adding the context attribute *lyricist* to *song*, the *song* and *album* become partially contained as there is not matching attribute region for *lyricist* in album.

Our proposed approach performs a search over related situations which have matching regions of acceptable values. We argue, by searching related situations it might be feasible to compute the attribute region value and relevance (weight) of a context attribute. Suppose *song* and *album* are partially-contained (refer previous example). A new context attribute *composer* in *song* will have not have a corresponding region of acceptable values. Since *song* and *album* are partially-contained, by searching *album,* a suitable region of acceptable value for *composer* can be computed. We use the terms *similar* and *co-exist* to represent the type of situation containment. Since there can be many partially-contained, contained and equi-contained situations, we use the Jaccard similarity coefficient (Doan et al., 2002) to compute similarity between situations.

$$SimilarityCoefficient(X,Y) = P(X \cap Y) / P(X \cup Y)$$
$$= \frac{P(X,Y)}{P(X,Y) + P(X,\overline{Y}) + P(\overline{X},Y)} \qquad (5\text{-}5)$$

X and Y are two situations. $P(X, Y)$ is the number of attribute regions that are similar in both situations. $P(\bar{X},Y)$ is the number of attribute regions in X similar to attribute regions in Y and $P(X,\bar{Y})$ is the number of attribute regions in Y similar to attribute regions in X. We go by the assumption that in a specific application domain, the relevance and attribute region values of a context attribute are shared among similar situations. The Jacard's coefficient provides a way to find situations in the application space that best matches the situation being reasoned. The set of situations that share one or more context attribute regions with the situation being reasoned, constitute a situation closure. The similar situations are ranked based on Jacard's similarity co-efficient which is then used to determine the best sets of relevance and acceptable region of values for the context attribute. The newly discovered attribute's relevance and attribute region values are incorporated into the temporal situation space. The confidence is recomputed using the newly computed temporal situation space. The situation composition using dynamically discovered context is illustrated in Figure 5.7. The situation defined using two attributes a_1 and a_2 evolves spatially by incorporating newly discovered context a_3.

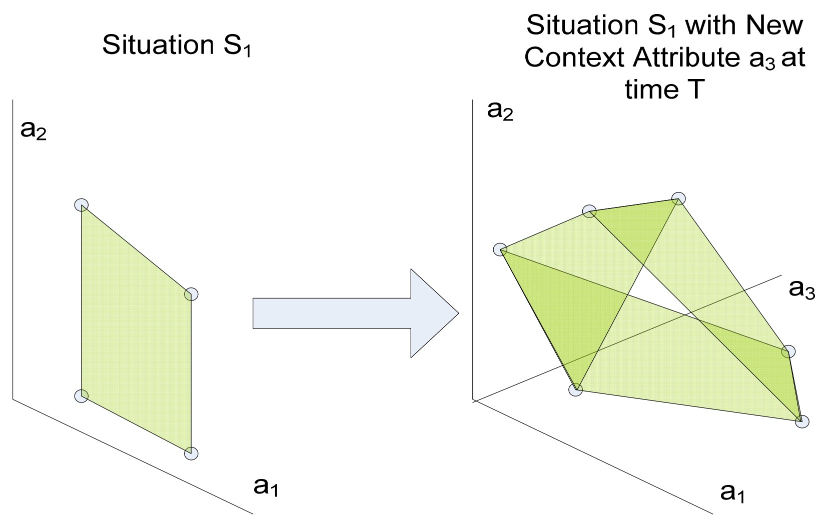


Figure 5.7: Illustration of Situation Composition with Dynamic Context

For example, consider the situations *presentation* and *lecture* in a smart room. These two situations are assumed to have some form of containment (co-exist). The situation *presentation* is defined with the following context attributes and corresponding attribute region values *projector* (*on*), *noise* (*4 - 7*), *people* (*5 - 30*) and *light* (*dim*). The situation *lecture* is defined with the following context attributes and corresponding attribute region values *projector* (*on*), *noise* (*1 - 2*) and *light* (*dim, bright*). At time *T*, the context state values for the following context attributes *projector, noise, light* and *people* are available. Using the context state values at time *T* the situation *lecture* can be reasoned within the smart space. While reasoning the situation *lecture*, the context attribute *people* is the newly discovered attribute with no corresponding context attribute region definition. Using the proposed similarity-based reasoning, the situation *presentation* closely matches the definition of situation *lecture*. Hence, the attribute region and the relevance for the context attribute *people* are derived from the similar situation *presentation*. The new context attribute is then added to the temporal situation space computed at time *T*. As an extension to the above technique, at a much higher level, when sufficient similarity is not reached between situations, ontology-based (Gu et al., 2004) matching may be used to strengthen the reasoning process. The pseudo code that computes the relevance and the corresponding attribute region value, for the dynamically discovered context attribute, is presented in Figure 5.8. A description of the pseudo code is presented in the paragraph following Figure 5.8.

```
Pseudo Code: Compute relevance and attribute region for newly
discovered context attributes
Input: New Context Attribute $a_i$, Reasoned Situation (S)
Begin Compute Similar Situations (C(S))
   1. for each situation s in Application Space($\Re$)
   2.       if context attribute $a_i \in$ s
   3.             similarity_coefficient = $\frac{P(S,s)}{P(S,s) + P(\bar{S},s) + P(S,\bar{s})}$
   4.       end if
   5. end for
   6. rank (C(S))
   7. obtain attribute relevance and attribute region
   8. return relevance, attribute region
End
```

Figure 5.8: Compute newly discovered context attribute's relevance and attribute region value - Pseudo Code

Step 1 computes the list of situations that can co-exist with the situation being reasoned. Step 2 and 3 iterates through each situation in the application space. Step 4 computes the similarity coefficient between the reasoned situation and the situation in the application space. This is denoted as *C(S)* containing the list of situations similar to the reasoned situation S. Step 7 - 9 sorts the situation list *(C(S))* based on the similarity coefficient. The relevance and attribute region values of the highest ranked situation are returned.

5.6 Sensor Data Quality and Flexible Attribute Region

The second extension proposed to Context Spaces (CS) is the introduction of sensor data quality and flexible attribute regions. These parameters help in computing the context attribute's contribution in the reasoned situation. The contribution determines the influence of the context attribute on the outcome of the reasoned situation. For example, suppose a situation *presentation* is defined over the context attribute *light* with attribute region values *(dim, bright)*. If at time *T*, the context state value for the context attribute *light* is dim, then the contribution of the attribute *light* in the situation *presentation* is 1. Alternatively, if the context state value is *off*, then the contribution is 0. The contribution of a context attribute is used as evidence to infer the situation.

To compute the contribution, CS takes into consideration the context attributes region and its context state value at time *T*. If the context state value for the context attribute is within the attribute region, it results in a contribution value of 1, otherwise 0. The contribution of a context attribute *(a_i)* within the situation *S* in CS (Padovitz, 2006) is computed as

$$\boldsymbol{c_i} = \mathbf{Pr}(\hat{\boldsymbol{a}}_i \in \boldsymbol{A_i}) \tag{5-6}$$

In equation (5-6), P_r is the probability that context attribute a_i is within the corresponding attribute region A_i. a_i represents the sensed value (context state) for the context attribute at time *T*. $\hat{a}_i$ represents the error corrected value given by

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a_i} \pm \boldsymbol{e_i} \tag{5-7}$$

e_i is the error in the sensor reading. CS assumes that error is obtained from the sensor or from a probability distribution function based on historic sensor data collected. The probabilistic approach ensures error/deviation in sensor data does not influence the reasoning process. The function used by CS to compute the contribution value is a crisp/rigid function i.e. it follows the following criteria

$$\boldsymbol{c_i} = \begin{cases} \mathbf{1}, & \widehat{\boldsymbol{a_i}} \in \boldsymbol{A_i} \\ \mathbf{0}, & otherwise \end{cases} \tag{5-8}$$

This assumption of CS reduces its reasoning capability in situations where the context state value for the context attribute a_i at time T is marginally outside the attribute region value. For example, suppose the context attribute *temperature* is defined in the situation *meeting* with a corresponding attribute region value *24 - 27.* CS crisp boundary approach computes a contribution of 1 if the context state value of *temperature* is between *24* and *27* and *0* if it is outside this range (*e.g. 27.01*). An illustration of the Context Spaces crisp boundary is provided in the earlier example is depicted in Figure 5.9.

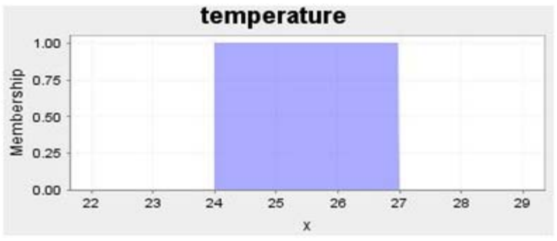


Figure 5.9: CS crisp boundary illustration

One approach to convert the crisp context attribute region definition into a more flexible approach is to use Fuzzy logic and Fuzzy membership functions (Mendel, 1995). A typical fuzzy logic-based membership function is depicted in Figure 5.10. Though fuzzy logic can be used to compute varying membership values, it is not very useful in many application scenarios where a decision has to be made with certain confidence. For example, it is not very helpful for a context-aware application to conclude that it is 0.6 *hot* and 0.4 *cold*. Further, the use of fuzzy logic requires precise definition of fuzzy sets and membership functions to determine the context attribute's membership. Moreover, fuzzy does not provide a likelihood of occurrence of an event or condition but merely provides a value between 0 and 1 where 0 represents no membership, 1 represents complete membership and a value between 0 and 1 represents partial membership.

We propose a fuzzy like approach by extending the crisp region of Context Spaces to a flexible-crisp region i.e. defining extended boundary regions around the existing attribute region definition. We define outer regions to mean existing attribute region definition but allowing certain deviation while computing the contribution. Our contribution function is computed using the following rule

$$\boldsymbol{c_i} = \begin{cases} \mathbf{1}, & \widehat{\boldsymbol{a_i}} \in \boldsymbol{A_i} \\ \varpi, & \widehat{\boldsymbol{a_i}} \in (\boldsymbol{A_i} \pm \widehat{\boldsymbol{A_i}}) \\ \mathbf{0}, & otherwise \end{cases} \tag{5-9}$$

The value ϖ is the contribution if the context state value of the sensor is within the outer attribute region ($\widehat{A_i}$). The value ϖ is application defined which gives the application developer the ability to define contributions for context attribute when its context state value is just outside the CS defined crisp attribute region.

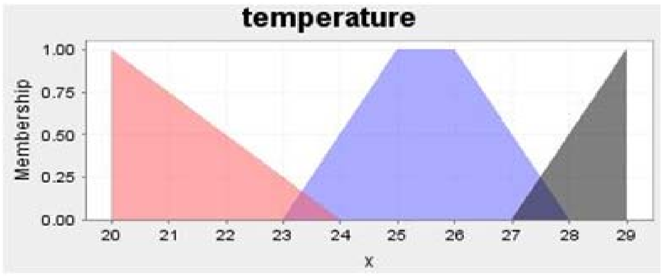


Figure 5.10: Fuzzy Membership Function

Example: Consider a context attribute *temperature* defined with a context attribute region value of 22 - 25^0 C for the situation "*GOOD OPERATION*". If the context state value for the context attribute *temperature* is *25.05*, CS approach will return 0. This results in the context-aware application concluding that operating condition is not good. This is primarily attributed to the crisp attribute region definition of CS. We apply the proposed approach to the same scenario. We define an outer context attribute region ($\widehat{\boldsymbol{A_i}}$) of ±0.5. In this case, the context state value of *25.05* for the context attribute temperature would return the value ϖ rather than 0 as in the case of CS. The proposed outer region-based approach allows the application to incorporate deviations in the context attribute region definitions.

The value ϖ is computed using a fuzzy inspired approach. Instead of using fuzzy membership functions the context state value in the outer region can be fitted into a simple distribution function which can be used to determine the values ofϖ. Figure 5.11 is an illustrative example of how ϖ can be computed using the different distributions. In case 1, the outer region is defined as a crisp region while in case 2, the outer region is characterised as a normal distribution. Figure 5.12 illustrates the computation of ϖ for different context state values. We have assumed a value of 0.5 for ϖ.

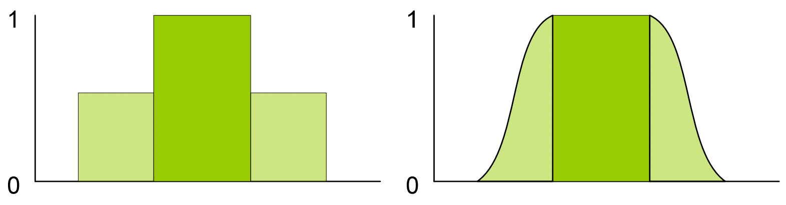


Figure 5.11: Distribution function to compute ϖ

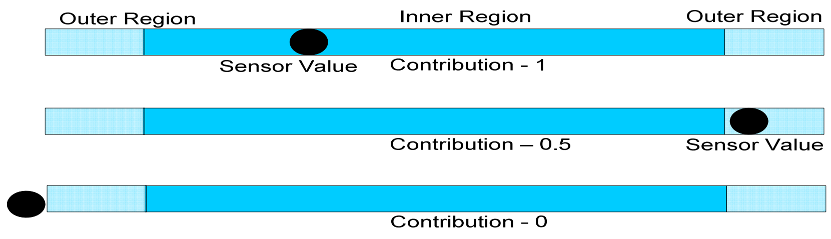


Figure 5.12: Contribution c for a crisp outer region

The next proposed extension incorporated in CS is sensor data quality. The sensor error heuristic employed by CS relies on pre-defined error probability to compute sensor errors given by (6) and (7). The error value e_i is adjusted with the sensed value for the context attribute a_i to compute the error corrected value $\widehat{a_i}$. We introduce sensor data freshness parameter to determine the quality of the data collected. By data freshness we try to answer the question *how recent the data is*? We do not change our focus into quality of context (Buchholz et al., 2003) as that is a topic of research by itself. Our approach uses sensor data freshness to identify old sensor data. Sensor data quality is important in pervasive environments especially in cases where sensor data is collected using mobile data mules. The mobile data mules have their own schedule to deliver data to the sink. Hence, the sensor data may not be always recent. By using data freshness parameter, the context-aware application obtains the ability to distinguish between old and current data. Incorporating the flexible attribute region value and sensor data freshness, the new contribution function is given by

$$\boldsymbol{c_i = \widehat{\Pr}(\hat{a}_i \in A_i) * Freshness} \qquad (5\text{-}10)$$

The function $\widehat{\Pr}$ represents the modified sensor error probability that takes into consideration outer attribute region definitions. The *Freshness* metric is defined using a predicate $F_p(a_i)$. The outcome of the *Freshness* metric is given below.

$$\boldsymbol{Freshness} = \begin{cases} \mathbf{1}, & \widehat{a_i}\, \boldsymbol{satisfies}\ \boldsymbol{F_p} \\ \mathbf{0}, & otherwise \end{cases} \tag{5-11}$$

For example, let the predicate $F_p(a_i)$ be defined as $0 \leq hour \leq 3$. The predicate *hour* is the data freshness metric's unit. Sensor data for the corresponding context attribute a_i will result in a contribution value 0 if the data is more than *3* hours old. The contribution function given by (5-10) makes the sensor inaccuracy heuristics and the sensor data freshness dependent. The sensor inaccuracy heuristic also incorporates flexible attribute regions. To handle scenarios where high degree of independence is required between sensor error and data quality, we define a modified contribution computation function below

$$\boldsymbol{c_i = d_1 * Pr(\widehat{a_i} \in A_i) + d_2 * Freshness} \tag{5-12}$$

d_1 and d_2 are weights assigned to each sensor heuristics, making them independent of each other. Hence, an application can decide which heuristic needs to have a major impact on the contribution computation function. Making $d_2 = 0$ will remove the influence of sensor data freshness heuristic and vice-versa.

5.7 Hierarchical Context Attribute Regions

The final extension proposed to Context Spaces (CS) (Padovitz, 2006) is the introduction of hierarchical context attribute regions. CS defines single level context attribute regions. This approach falls short in exactly capturing real-world situations where context attribute regions are related. By relation, we create the notion of dependence i.e. the outcome of one context attribute is based on another. Since situations in CS are defined using context attribute regions, we propose a hierarchical approach that facilitates relationships between context attribute regions.

For example when reasoning the situation *person running* using the context attribute *speed*, CS approach uses context attribute region to define *speed* for the running person which can range from *5 to 10 km/h*. This reasoning approach is sufficient when situations are defined at high-level i.e. *person running*. When situations are required to be defined at much finer granularity, the outcome of the *person running* situation becomes highly dependent on the person type i.e. the speed of the person is dependent on the age of the person. The value for speed would vary for a person in the age group 15 to 25 and a person in the age group 45 to 50. Current CS approach does not provide a way to reason situations based on such relationships.

The proposed hierarchical context attribute regions allow defining relationships between attribute regions. The proposed hierarchical attribute region approach has the following merits: 1) define situations with finer granularity helping the virtual world to model the real-world situation with improved accuracy, 2) improve the reasoning ability of the context-aware system. To incorporate multi-level context attribute relationships, we introduce the terms "*parent*" and "*child*" context attribute regions. Every parent attribute region has a corresponding range for its child attribute regions. For example, if the *parent* context attribute region *age* is between *15 and 25*, then the *child* context attribute region *speed* may have a value 5 to 10 km/h, alternatively, if the *parent* context attribute region *age* is between *45 and 60*, then the child context attribute region *speed* may have the value range *3 to 5* km/h.

The similar situation can be defined in CS but independently i.e. a situation for a person between the age *15 and 25* and a situation for a person between the age *45 and 60*. Suppose *age* and *speed* have a weight of *0.5* assigned to them respectively and the attribute region definition defined independently as *45 to 60* and *3 to 5 km/h*. For a context state value of *speed = 6* and *age = 50,* CS will compute a confidence of *0.5* for the situation *running*. The 0.5 computed by CS can be misleading as the *context-aware* application may not be able to completely exclude the occurrence of the situation *running*. Using the proposed hierarchical attribute region relationships, the contribution of the parent context attribute is computed as a function of the child context attributes. Hence, the overall contribution using R-CS would be 0. The R-CS approach aids in negating the occurrence of the situation *running* with higher certainty.

The proposed approach allows definition of multiple corresponding attribute regions for a context attribute within the situation space. Hence, based on the primary context attribute's predicate, the child context attributes predicate changes within the situation definition. Further, the overall contribution of the parent attribute is computed by computing the cumulative contribution of the child attributes. A real-world situation definition using the proposed hierarchical context attribute regions approach is presented in Table 5.1.

Situation:	**Running**			Relevance
Context Attribute	Context Attribute Regions			(weights) 0 to 5
Location	"=GYM"	"=PARK"	"!=OFFICE"	2
Age	Sub Regions	"20 - 30"	"40 - 50"	5
	Speed	≥ 6 & ≤ 8	≥ 2 & ≤ 4	5
	Heart Rate	≥93 & ≤146	≥83 & ≤131	5
	Systolic BP	≥108 & ≤122	≥112 & ≤130	5

Table 5.1: Definition of situation "Running" illustrating hierarchical context attribute regions

Consider the primary context attribute a_i associated with attribute regions A_i^d and its related sub-attributes a_{dk} with associated attribute regions A_{dk}. The notation d represents the number of sub attribute regions for the context attribute a_i. Each A_{dk} is associated with a weight w_k that represents the relevance of the context attribute a_{dk} within the situation. The contribution of each context attribute a_{dk} within the situation is computed using (5-12). The total contribution of the primary context attribute (a_i) is given by

$$c(a_i) = \sum_{k=0}^{r} w_{dk}\, c(a_{dk}) \quad 1 \leq d \leq n \tag{5-13}$$

The contribution $c\ (a_{dk})$ determines the contribution of the k^{th} context attribute whose corresponding context attribute region is given by the dimension d of the parent context attribute a_i. As illustrated in Table 5.1, the context attribute (a_i) has A_i^d

corresponding attribute regions. *d* takes the value 2 in the example given in Table 5.1. In a generic form, we can represent the attribute region relationship in a matrix given by

$$\begin{array}{lll} \boldsymbol{A_i^1} & \boldsymbol{A_i^2} \ldots & \boldsymbol{A_i^d} \\ \boldsymbol{A_1^1} & \boldsymbol{A_1^2} \ldots & \boldsymbol{A_1^d} \\ \boldsymbol{A_k^1} & \boldsymbol{A_k^1} \ldots & \boldsymbol{A_k^d} \end{array} \tag{5-14}$$

The header row represents the number of context attribute region dimensions along with their predicates for the context attribute a_i. The rows represent the number of dependent context attribute regions for the parent attribute a_i with corresponding attribute region values.

5.8 Summary

Sensors both physical and virtual are the key source of data in smart spaces. Hence, we require approaches that are capable of modelling real-world situations (smart spaces) based on available sensor information. Moreover, these models need to have the capability to represent the real-world situations with great detail, allowing context-aware applications to reason situational context, i.e. reasoning with the notion of situations. In this chapter, we proposed extensions to Context Spaces incorporating dynamic reasoning based on available sensor data. Context Spaces' is a generic context modelling approach that uses situations-based reasoning. Since sensor data changes over time and not all data are available ubiquitously, it is important for a context model to have some level of adaptation. We have proposed the following extensions to Context Spaces with the goal to introduce dynamic situation adaptation.

(1) A dynamic situation-composition approach that models real-world situations based on available sensor data. We proposed a technique that employs similarity measures to compute relevance and attribute regions for context attributes that are dynamically discovered within the smart space. Our dynamic situation composition employs situation partitioning, temporal situation composition and dynamic weight re-computation algorithms to improve reasoning capability under uncertainty.

(2) The use of flexible attribute region approach against a crisp attribute region definition employed by Context Spaces. This approach aids the reasoning process by adding more flexibility and tolerance to sensor values that lie within the border of attribute region definition.

(3) Additional sensor inaccuracy metrics, namely, data freshness. This attribute addresses the problem of out-of-date sensor data, increasing the accuracy of the reasoning process.

(4) Hierarchical (multi-level) context attribute regions that facilitate defining relationships between contextual attributes. The multi-level context attribute provides applications with the ability to define situations with finer granularity.

6 Implementation and Prototyping of sGaRuDa, 3D-KNN and R-CS

6.1 Introduction

In the previous chapters, we proposed a system framework, namely, sGaRuDa that enables mobile devices to collect data from sensors distributed within pervasive environments. We extended the proposed architecture to suit a wider range of sensors by proposing 3D-KNN, a *k* nearest neighbour-based sensor data collection algorithm. Finally, we proposed a situation-based smart spaces modelling approach, namely, R-CS. The modelling approach allows dynamic adaptation of situations based on available sensor data.

In this chapter, we present prototype and algorithm implementations of the proposed sensor data collection and smart spaces modelling approaches. This chapter is divided into three major sections focusing on the three main contributions of this thesis. In each section, we initially present an overview of the development tools followed by details of system/algorithm implementations. This chapter is organised as follows: Section 6.2 presents the implementation details of the proposed data collection system framework sGaRuDa. Section 6.3 presents the implementation of the *k*-Nearest Neighbour based data collection approach namely, *3D-KNN*. Finally section 6.4 presents the implementation details of the R-CS system.

6.2 sGaRuDa: Proof-of-Concept Implementation

Chapter 3 proposed sGaRuDa, a system framework for mobile data mule-based sensor data collection. As stated in Chapter 3, our proposed approach does not rely on prior network infrastructure information for sensor data collection. Moreover, the proposed system framework does not require any specialised hardware to communicate with the underlying sensor network infrastructure. sGaRuDa achieves sensor data collection without the need for a fixed data collection infrastructure. The proposed data collection algorithms were targeted at Bluetooth-based sensor networks. The use of Bluetooth-based sensor networks leverage on the abundant existence of Bluetooth-based mobile devices in current real-world environments. This is primarily attributed to the wide acceptance of Bluetooth (BluetoothSIG, 2010a) technology across a multitude of mobile device platforms (Eliasson et al., 2008, Nachman et al., 2005, Leopold et al., 2003).

In this section, we present details of our system prototype implemented on real-world mobile devices that belong to smart spaces. The implementation on mobile devices (PDA) proves the functional feasibility of our proposed sensor data collection architecture without the need for specialised hardware for data collection and delivery. The implementation scenario is presented in detail in Chapter 7. The detailed discussion of implementation and prototyping of sGaRuDa, 3D-KNN and R-CS presented in this Chapter is extended from the following published papers (Jayaraman et al., 2007, Jayaraman et al., 2008a, Jayaraman et al., 2008c, Jayaraman et al., 2009b, Jayaraman et al., 2009a, Jayaraman et al., 2008b, Jayaraman et al., 2010a, Jayaraman et al., 2010c, Jayaraman et al., 2010b)

6.2.1 Development Tools

6.2.1.1 Mulle Sensor Node

The underlying sensor network used in our implementation is the Mulle sensor node platform. Mulle is a Bluetooth-based sensor node developed at EIS Labs, Lulea University of Technology Sweden (EISLab, 2010). The Mulle is a low-powered wireless sensor node with the capability to transmit sensor data over Bluetooth (v3.1) or/and 802.15.4 (Zigbee) (v5.2). Figure 6.1 presents a Mulle sensor node.

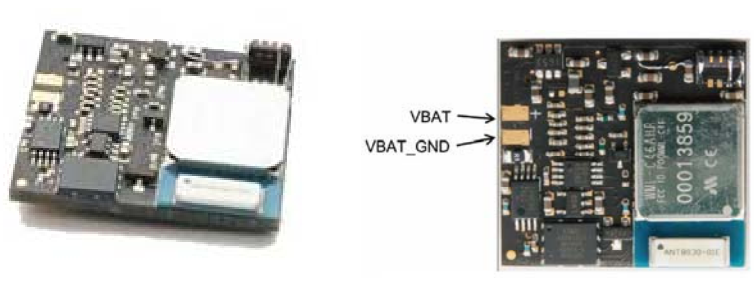


Figure 6.1: Mulle Sensor Node (adapted from (Eistec, 2009))

Hardware: The physical size of the sensor node is 24x26x5mm weighing 5 grams. It has a 10MHz Renesas M16 (Renesas, 2008) microcontroller. The radio communication hardware available on the Mulle is a Mitsumi Bluetooth module [4] and/or an 802.15.4 Zigbee module. The Mulle requires DC power in the range of 3.5V - 5.5V. The Mulle is equipped with a real-time clock (RTC) which serves the following purposes: 1) provides the MCU with a sub-clock and 2) generates timer interrupts. The Mulle is also equipped with an onboard flash memory of 2 MB. The Mulle has an onboard Dallas DS600 temperature sensor and provides an optional 26 pin connector to connect a multitude of sensors. The batteries used to power the Mulle sensor node are lithium or lithium-ion with capacity ranging from 120 mAh to 2200 mAh. The Mulle hardware platform architecture is presented in Figure 6.2.

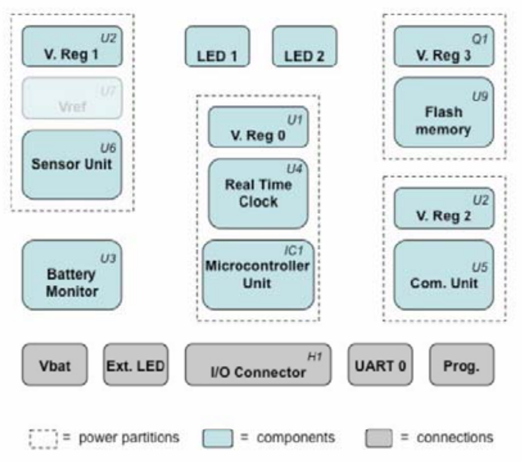


Figure 6.2: Mulle Hardware Architecture (adapted from (Eistec, 2009))

Software: The Mulle comes with open-source software written in C programming language. The current available firmware on the Mulle is developed using the IAR workbench (IARSystems, 2008). The software architecture of Mulle allows it to efficiently use the Bluetooth low-power modes, Park and Sniff to reduce the overall energy consumption hence greatly improving the lifetime of the sensor node. The software module provides the capability to change the MCU clock speed from 10MHz to 1MHz. The Mulle is provided with an expansion board that is used to program the Mulle to application requirements. The expansion board is connected by USB to the computer running the IAR workbench. A M16C-Flasher (M16CFlasher, 2008) is used to reprogram the Mulle using the expansion board. Figure 6.3 shows the Mulle expansion board (left) and a screenshot of the M16C Flasher application used to reprogram (right) the Mulle. The Mulle software stack uses standardised TCP/IP over Bluetooth to transfer sensor data. The use of TCP/IP allows Mulle to become part of existing internet infrastructure. This feature though requires the availability of a permanent Bluetooth access point to relay sensor data from the Mulle to the internet. The Bluetooth stack used by the Mulle is a lightweight Bluetooth stack implementation, namely, lwBT (Ohult, 2006). The stack supports the following Bluetooth profiles, namely: Dial-up Networking (DUN), Serial Port Profile (SPP), Personal Area Networking (PAN) and Local Area Network Access Profile (LAP).

We chose the Bluetooth-based Mulle sensor node platform for prototype implementation and experimentation because:

1) Bluetooth sensor nodes have been proved feasible for low-powered operations hence making them suitable for sensor network applications (Leopold et al., 2003). The Mulle sensor platform belongs to the category of low-powered sensor nodes. The operational power characteristics of Mulle is much efficient (Lundberg et al., 2005) than other popular Bluetooth-based sensor nodes like BTnode (ETH-Zurich, 2007) and iMote (Nachman et al., 2005). The Mulle's sleep time energy consumption is 0.012mW compared to 9.9mW for BTnode and 9mW for iMote sensor platforms respectively. A detailed analysis of Mulle's power consumption is presented in the evaluation section of the thesis.

2) Mulle uses C programming with open source stack implementation making it feasible to extend/develop new functionality.

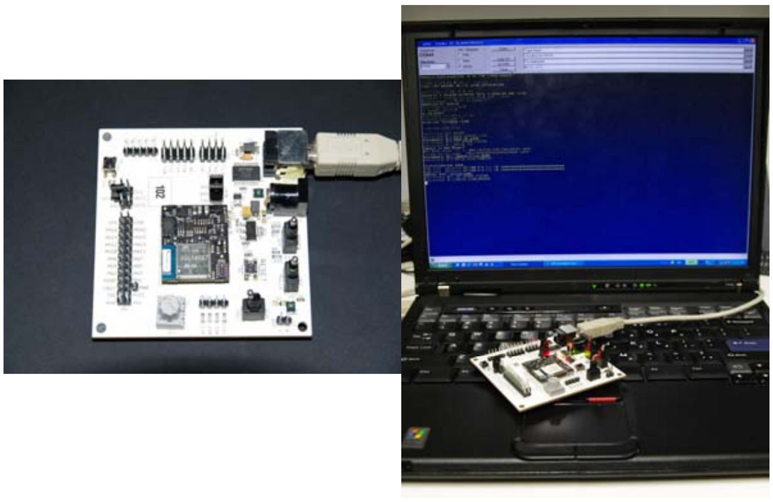


Figure 6.3: Mulle Expansion Board and Mulle Development Environment

3) The use of standardised Bluetooth stack and Bluetooth profiles provides interoperable capabilities. This feature allows Mulle to communicate with current-day consumer mobile devices with no additional hardware/software requirements. The BTnode and the iMote software architectures use customised Bluetooth stack to communicate with other sensors and the base station. Hence, inter-operability with current day mobile device platforms may not be directly possible.

6.2.1.2 Software and Hardware Toolkits

The data collection framework presented in Chapter 3 has been implemented on two classes of mobile devices: 1) A personal digital assistant (PDA) which exemplifies the class of current day smart mobile device platforms and 2) A mobile robot that represents the class of mobility-enabled devices that could be part of future pervasive environments. The mobile robot used in our implementation is assumed to be part of the pervasive environment and requires no special hardware to communicate with the sensor. Further, the mobile robot implementation is also used as an emulator to emulate smart mobile devices movements in real-world scenarios. The software and hardware development tools/kits used in the development of the system prototype includes:

1) Microsoft.NET Framework and Microsoft .NET Compact Framework (CF) (Microsoft.NET, 2010): We have used the Microsoft.NET and .NET CF platforms to develop our system prototypes. The .NET CF allows development of managed applications for resource constrained devices like PDA. We have used the .NET framework to simulate a simple sink operation. The collected sensor data is delivered by the mobile data mule to the sink. The .NET CF based application can be ported to a multitude of current-day smart mobile device platforms. Our proposed system architecture is not restricted by the development platform and can be migrated to any other mobile device platforms e.g. Symbian (Java) (SymbianFoundation, 2010), Android (Google, 2010), etc. The development in .NET CF was done using VisualBasic.NET platform.

2) BTAccess.NET (BTAccess, 2008) and BlueCove (BlueCove, 2007): The BTAccess.NET software development kit (SDK) provides mobile devices running the .NET CF platform the ability to communicate and control the Bluetooth hardware. The BTAccess.NET provides, application programmable interfaces (API) to managed applications hiding the underlying un-managed Bluetooth code functions. We have used BlueCove library (BlueCove, 2007) for the robot-based sensor data collector implementation. BlueCove is a Java library (JSR-82) for Bluetooth that works on most devices that support Java and has a supporting Bluetooth stack implementation. BlueCove stack is used to communicate and control the Bluetooth hardware present on the robot platform.

3) ER1 Robot Platform(Evolution-Robotics, 2010): The ER1 is an off-the-shelf robot platform developed by Evolution Robotics. The ER1 build is completely customisable based on application requirements. The robot hardware is equipped with infrared sensors for obstacle avoidance, a web-camera for vision and two motors controlling the robot's movement. ER1 comes equipped with a software development kit, the Evolution Robotics Software Platform (ERSP). ERPS allows software based control of the ER1 hardware. The heart of the ER1 is a laptop that controls the entire system functionality.

4) Ekahau Positioning Engine (Ekahau, 2010): Ekahau position engine (EPE) is a software-based location system that uses wireless access point to triangulate device locations. In our implementation, we use EPE as an indoor global positioning system (GPS) providing the mobile data mule its current location at any instance in time. The EPE is completely software based and only requires a wireless network interface card to be present on the device being tracked.

5) JAVA: We use JAVA to implement the data collection framework functions on the mobile robot. The factor behind the choice of language was attributed to the fact that ER1 and EPE provide java API's for robot control and location information. Hence, we decided to migrate sGaRuDa to JAVA taking advantage of platform independence and availability of robot control API's. The JAVA implementation also proves the platform independent nature of the proposed data collection architecture.

6.2.2 sGaRuDa: A Practical System Prototype

The implementations of sGaRuDa, the proposed data collection architecture are divided into three parts:

1) Mulle Sensor Node Implementation
2) Mobile Device Implementations
3) Sink Implementation

The sensor node implementation deals with the software implementations on the Mulle sensor node platform (Eistec, 2009). The mobile device implementations deal with implementation on the mobile device and mobile robot platforms. The sink implementation deals with simple sink functions. The sink implementation is used to test some of the features that are part of the sGaRuDa architecture e.g. the dynamic activation schedule. A complete sink implementation may not fit within the scope of this thesis. Hence, our sink implementation is a restricted to functionality that support sGaRuDa framework.

6.2.2.1 Mulle Sensor Node Platform Implementation

We stated in Chapter 3, the implementation on the sensor node is developed with sensor characteristics in mind aiming to reduce the workload and hence, improve the sensor's life time. The operation performed by the Mulle sensor node is depicted in Figure 6.4. As depicted in the flowchart, the Mulle sensor node is only expected to perform its regular functions. No additional task is enforced on the Mulle sensor node.

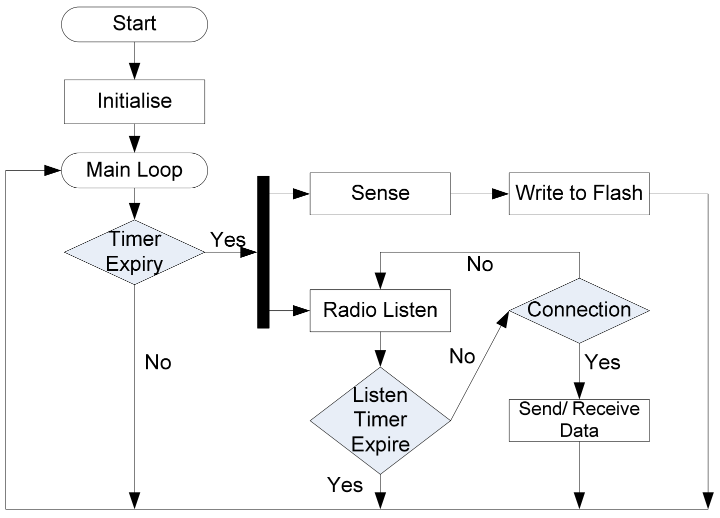


Figure 6.4: Mulle Sensor Node- Data Flow

Mulle programming was done using C language. We have developed new modules and modified existing modules to incorporate the functions required by the sGaRuDa framework. The primary functions that have been implemented on the Mulle are:

1) A scheduler function that periodically turns on the Bluetooth radio allowing the Mulle to participate in data collection.
2) An incoming data processing function that performs the following
    a. Dynamically receive and update the sensor's activation schedule
    b. Synchronise the sensor's time with the mobile data mule
    c. Exchange data with the mobile data mule using the requested window size by consecutively reading the data from the flash memory

3) Module to update the sensor node's name fields based on available energy and data to be transmitted.

For prototype implementation, we have used the Bluetooth serial port profile (SPP). One major reason to use SPP is attributed to its ease of connection establishment. Using other Bluetooth profiles like LAN Access Profile (LAP) introduce TCP/IP overheads. To keep the implementation on the Mulle simple, we stick to SPP profile. Figure 6.5 depicts the various modules that have been implemented on the Mulle to achieve the aforementioned functionality. The yellow coloured boxes represent Mulle's functional groups. The boxes in light green represent modules whose existing functions have been modified or new functions have been added as a part of sGaRuDa framework. The modules in dark green represent the newly implemented modules.

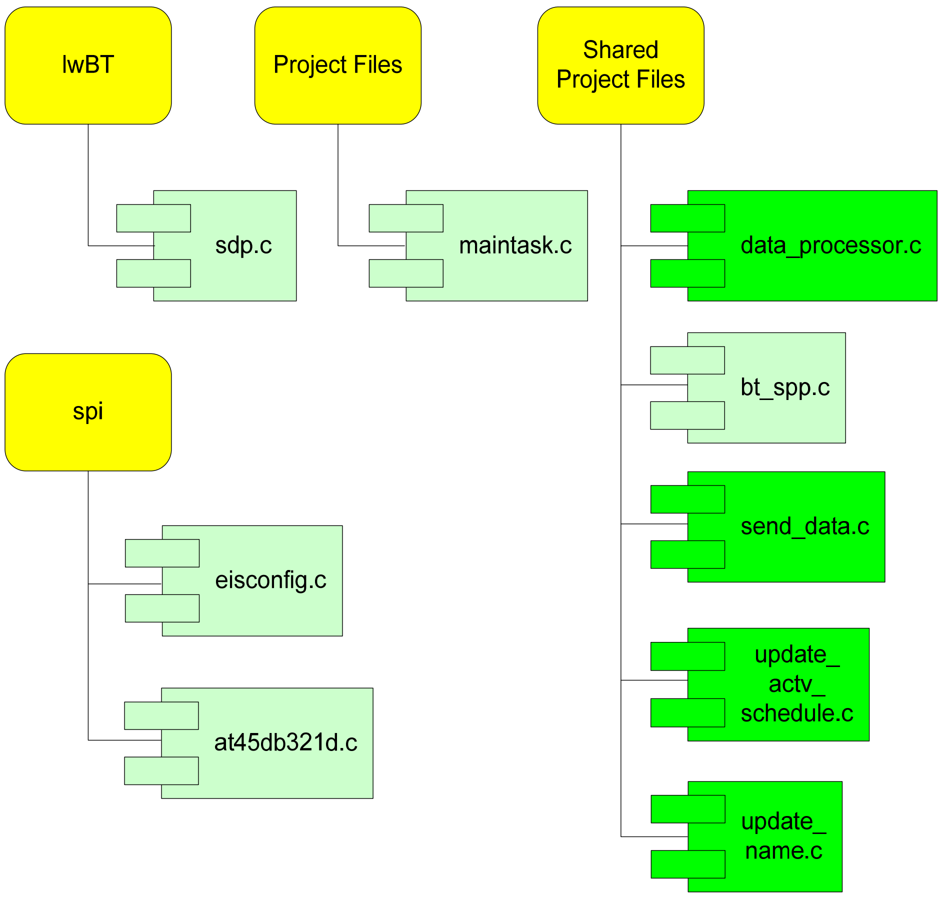


Figure 6.5: Mulle Modules Implemented

The lwBT folder contains the files corresponding to the Bluetooth stack. The file *sdp.c* has been modified to add a serial port service. The *spi* folder has files that relates to the flash memory on which the Mulle configuration is stored. We have modified the configuration files to introduce a timer value to control the Bluetooth radio and the sensing function. The *eisconfig.c* file defines the configuration and the *at45db321d.c* handles reading and writing from the onboard flash memory. Figure 6.6 presents a sample code of the *eisconfig.c* file.

```
void eis_config_default (void) {
 eis_config. bt_interval_s = 60 * 1;
  eis_config.bt_on_time_s = 60 * 3;
  eis_config.windowSize = 0;
  eis_config.bak_r_ptr = 0;
  eis_config.transferDone = 0;
  eis_config.nr_reboots = 0;
       eis_config.max_nr_files = 8000;
  eis_config.samples_per_file = 6;
  eis_config.sample_interval_s = 60 * 1;
       eis_config.w_ptr = 3;
  eis_config.r_ptr = 1;
  eis_config.nr_samples = 0;
  eis_config.min_volt_BT = 3.4;
 }
```

Figure 6.6: eis_config.c - Code Snippet

The *project_files* folder has application dependent files primarily the *maintask.c*. This module defines the program's entry point and has the code that functions within the main loop as depicted in Figure 6.4. The primary purposes of this file are: 1) Enable timers and interrupts; 2) Read configuration from flash memory during boot; 3) Call appropriate functions/modules when timer expires; 4) Run a watchdog timer to reboot the Mulle after execution failure. A code snippet of the *maintask.c* file is presented in Figure 6.7.

The shared_project_files contains the newly implemented modules related to sGaRuDa framework. The four main modules that have been implemented are data_processor.c, send_data.c, update_actv_schedule.c and update_name.c. The bt_spp.c file handles all serial port related functions i.e. establishing connection, tearing down a connection, relaying data from the RFCOMM layer to the data_processer.c and sending data using the send_data.c module over serial port. Figure 6.8 presents code snippets for the module bt_spp.c.

```
void maintask(void) {
 /* read system configuration to be used by all subsystems */
 read_config();
  /* turn on the timer output  and enable interrupts*/
 tm_set_state(TM_RTC, TM_ACTIVE);
 int2_start();
 int4_start();
 for (;;) {
  /* Handle timers */
 if ( KS_inqsema(INT2_SEM) == SEMA_DONE) {
      bt_period++;
      sense_period++;
      //code to check if data needs to be sensed
      if (sense_period == eis_config.sample_interval_s / 60){
            sense_period = 0;
            sensord_sample();
      }
      //code to check if Bluetooth needs to be turned on
      if (bt_period == eis_config.bt_interval_s / 60) {
            bt_period = 0;
            if (lwbt_active !=1){
            //Setting a task to start the Bluetooth radio immediately
                  titask_add(TI_RUN_ONCE, 1, lwbt_start, NULL);
                  //add task to turn bluetooth off
                  titask_add(TI_RUN_ONCE, eis_config.bt_on_time_s * 4,
                  lwbt_disconnect, NULL); }}
      }  } // for(;;)
} // maintask
```

Figure 6.7: maintask.c - Code Snippet

The *data_processor.c* module handles all incoming data request. This module can also be called the controller as it decides the next corresponding action to be performed by the Mulle. For example, if the request is for data transmission the module instructs the *send_data.c module* to begin data transfer for the specified window size. Figure 6.9 presents code snippets of *data_processor.c* module.

The *update_actv_schedule.c* handles the following functions: 1) un-parse the activation schedule received from the mobile data mule; 2) Update the Mulle with the new activation schedule. The above functions performed by *update_actv_schedule.c* are presented as code snippets in Figure 6.10 and Figure 6.11.

```
err_t com1_recv(void *arg, struct rfcomm_pcb *pcb, struct pbuf *p,
err_t err) {
 char *tempdata;
 u8_t *data;
 struct pbuf *returnp;
 int counter = 0;
 if (p->len > 0){
 while (q) {
     data = (u8_t *)p->payload;
      if (counter >=1) {
            tempdata = realloc(tempdata,strlen(tempdata) +
                                          strlen(data));
            strcat(tempdata,data);        }
      else{
            tempdata = malloc(strlen(data));
            strcpy(tempdata,data);        }
      counter ++;
      p = p->next;            }
     putchar('\n');
     returnp = processIncomingData(tempdata, pcb);    }
 return ERR_OK;   }
```

Figure 6.8: bt_spp.c - Code Snippet

```
struct pbuf* processIncomingData(char* data, struct rfcomm_pcb *pcb1) {
 char *temppayload, *copiedpayload, *result;
 struct pbuf *returnp;
 struct Date *now;
 char w_size[2];
 if (strncmp(data, "DATE", 4) == 0) {
  now = getDatafromDate (data);
  setDatatoRTC(now);     }
 else if (strncmp(data,"BEGIN",5) == 0) {
  printf("Recevied New Activation Schedule\n");
  readActivationSchedule(data);      }
 else if (strncmp(data,"STARTD", 6) == 0){
  result = strtok(data," ");
  result = strtok(NULL," ");
  w_size[0] = result[0];
  w_size[1] = '\0';
  if (atoi(result) != 0){
      eis_config.windowSize = atoi(w_size);
      returnp = sendData(data,pcb1);            } }
 else if (strncmp(data,"ACK", 3) == 0){
  eis_config.r_ptr = eis_config.r_ptr + eis_config.bak_r_ptr;
  returnp = sendData(data,pcb1);          }
 return returnp;
}
```

Figure 6.9: data_processor.c - Code Snippet

```
void readActivationSchedule(char *aSchdule) {
 int boolVCALEVENT = 0;
 char *date, *rrule, *desc, *result = NULL;
 char *result1 = NULL, *tempresult = NULL;
 result = strtok(aSchdule,"\n");
 while (result!=NULL){
  if (strncmp(result,"BEGIN:VCALENDAR", 15) == 0){
       result = strtok(NULL,"\n");
       result = strtok(NULL,"\n");   }
  if (strncmp(result,"BEGIN:VEVENT",12) == 0){
       boolVCALEVENT = 1;            }
  else if (strncmp(result,"DTSTART",7) == 0){
       date = result;          }
  else if (strncmp(result,"RRULE",5) == 0){
       rrule = result;         }
  else if (strncmp(result,"DESCRIPTION",11 )==0){
       desc = result;          }
  else if (strstr(result,":VEVENT") != NULL){
       if (boolVCALEVENT == 1){
             processNewEvent(desc,rrule,date);
             boolVCALEVENT = 0;            }
  }
  result = strtok(NULL,"\n");
}}
```

Figure 6.10: update_actv_schedule.c - Code Snippet 1

```
void updateEISConfig(int descrip, int period, int duration) {
 //description =0 then its RADIO_LISTEN
 if (descrip == 0){
 //represents the time when the radio will turn on-> periodic
 eis_config.bt_interval_s = 60 * period;
 //represents the time the radio will be on
 eis_config.bt_on_time_s = 60 * duration;
 reboot();  }
 //description = 1 then its sense
 else if (descrip == 1){
 eis_config.sample_interval_s = 60 * period;
 reboot();  }
}
```

Figure 6.11: update_actv_schedule.c - Code Snippet 2

6.2.2.2 Mobile Data Mule: Device Implementations

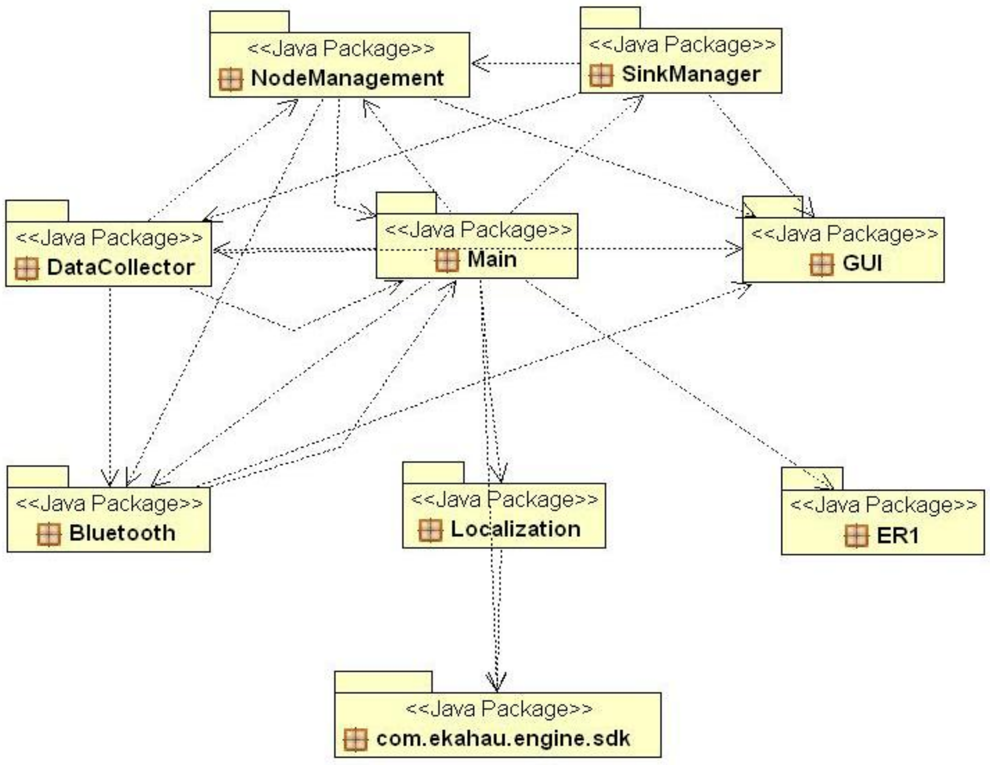
The following key functions have been implemented on the mobile device platforms.

1) Sensor Discovery

2) Sensor Data Collection

3) Sensor Data Delivery

Apart from these key functions, the mobile device platform implementation performs the task of interfacing with the mobile device's communication and location sub-system. The two mobile device platform used for the proof-of-concept implementations are Personal Digital Assistant (PDA) and mobile robot (ER1). The PDA ran Microsoft Pocket PC 2003. Microsoft.NET framework was used to develop code for the PDA platform and JAVA was used for the mobile robot platform. The primary functional modules/classes for both platforms are identical with device-specific functionality added to each device-specific implementation. For example, the device-specific implementation for the mobile robot is a module that controls the robot movement. Figure 6.12 and Figure 6.13 illustrates the package/class diagram for the two platform implementations. A detailed description of each package implementation is presented in Appendix A, Figure A2.



<<Java Package>> NodeManagement
<<Java Package>> SinkManager
<<Java Package>> DataCollector
<<Java Package>> Main
<<Java Package>> GUI
<<Java Package>> Bluetooth
<<Java Package>> Localization
<<Java Package>> ER1
<<Java Package>> com.ekahau.engine.sdk

Figure 6.12: Package Diagram - Mobile Robot Implemented in JAVA

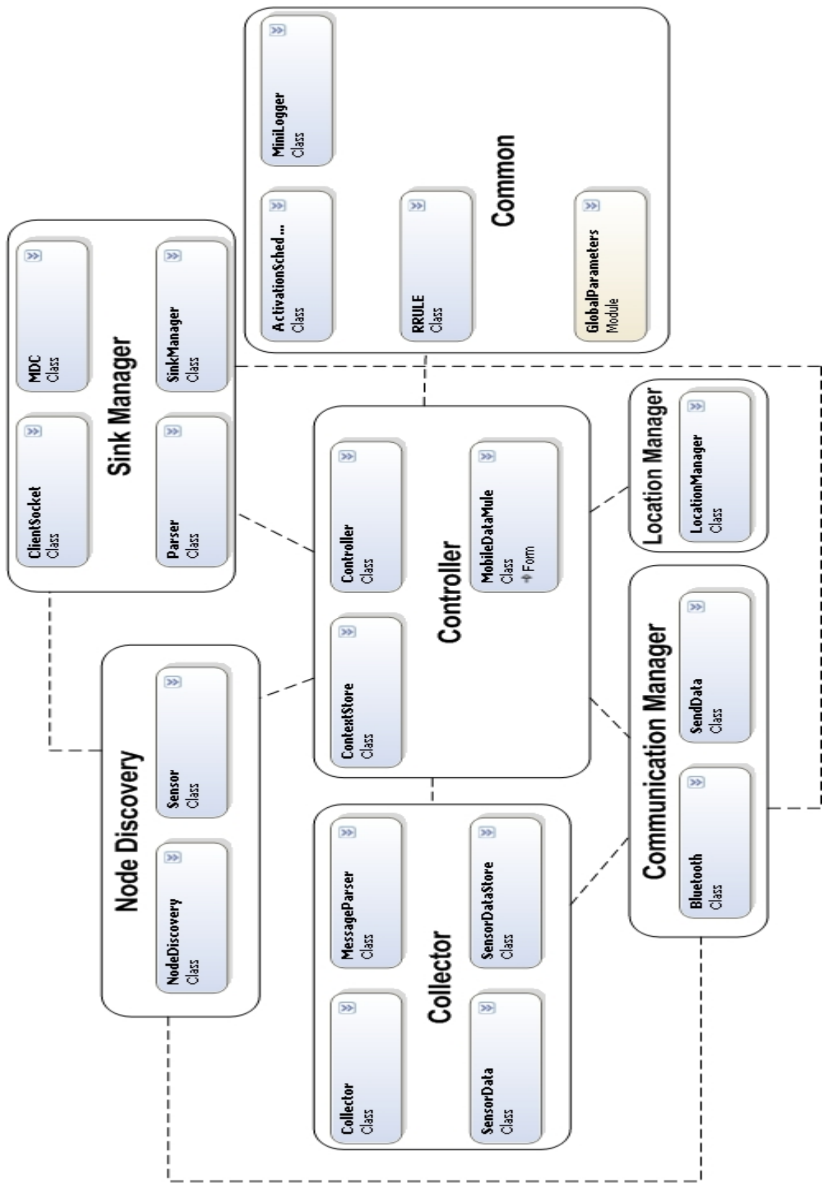


Figure 6.13: Class Group (Package) Diagram - PDA Implemented in .NET CF

Node Discovery Module: The node discovery module comprises classes that perform the operations sensor discovery, sensor information storage and sensor node management. Sensor discovery is performed by connecting to the mobile device's Bluetooth stack. The node discovery module also handles parsing and storing of the sensor information (metadata). The sensor naming convention follows the format presented in Chapter 3. The sensor repository information is synchronised with the sink periodically. Figure 6.14 presents code snippet for the sensor node discovery process.

```
Public Function discoverNodes(ByVal _bluetooth As Bluetooth, ByVal
duration as Integer) As ArrayList
 Dim _sensor As Sensor, _sensorAdd As String
 Dim availableNode As New ArrayList
 done = _bluetooth.findDevices(duration)
 If done = True Then
  For i = 0 To deviceList.Count - 1
  Dim sense_name = _bluetooth.getSensorName(deviceList.Item(i))
  found = checkSensorinList(sense_name)
  availableNode.Add(deviceList.Item(i))
  'check if discovered node is already in the node database
     If found = False Then
     _sensorAdd = _bluetooth.getSensorAddress(deviceList.Item(i))
     _sensor = createSensorProfile(sense_name, deviceList.Item(i))
     discoveredDevices.Add(_sensor)
  Else
  For j = 0 To discoveredDevices.Count - 1
       temp = discoveredDevices.Item(j)
             sensorName = readSensorName(sense_name)
       If (sensorName = temp.Name) Then
             temp.LifeTime = 0
             temp.LifeTimeCurrent = True
       End If
  Next
  Next
 Else
  addResult("Search Complete. No Devices Found")
 End If
Return availableNode
End Function
```

Figure 6.14: Node Discovery - Code Snippet

The *_bluetooth.findDevices(duration)* function determines the amount of time the Bluetooth discovery process needs to last. This function implements the shortened interlaced Bluetooth inquiry scan presented in Chapter 3. The section of code highlighted in bold manages the dead node problem discussed in Chapter 3. The variable *LifeTimeCurrent* determines if a sensor node previously discovered at a particular location is still active. The value of the *LifeTimeCurrent* variable is periodically synchronised with the centralised sink allowing the mobile data mule to determine dead sensor nodes. The *discoveredDevices* variable is an array list which is used to represent the node repository maintained by the mobile data mule.

Communication Manager: This module performs the communication manager functions of the sGaRuDa framework presented in Chapter 3. It has a set of functions and classes that creates an interface for the components node discovery and data collection, to use the Bluetooth stack available on the mobile device. The Bluetooth software development kit (SDK) used in .NET is BTAccess.NET (BTAccess, 2008). For the mobile data mule implementation on the mobile robot in JAVA, we used BlueCove Bluetooth implementation (JSR82) (BlueCove, 2007).

Collector: The *collector* module performs the role of the data collection module of the sGaRuDa framework presented in Chapter 3. For data collection, the mobile data mule employs the proposed window-based data collection algorithm. The window size is determined based on the computed data collection threshold. In our implementation we have assumed a window size of 2 for threshold between 0.5 to 0.6, 3 for threshold between 0.6 to 0.7, 5 for threshold between 0.7 to 0.8 and 7 for threshold above 0.8. This value is application dependent and can be customised. The *collector* module uses the packet structure presented in Chapter 3 which is reproduced in Figure 6.15.

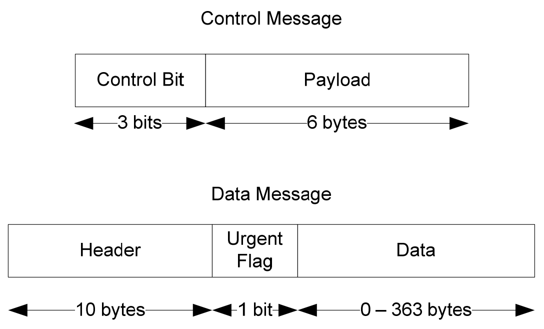


Figure 6.15: Control and Data Messages Format

The control message (refer Figure 6.15) implemented are “STARTD” for starting data transfer, “BEGIN” to upload a new activation schedule, “DATE” for time synchronisation and “ACK” for acknowledging data transmission. The Mulle response codes are “OK” for successful reception that includes control (start data transfer) and data messages (activation schedule), “NODATA” to terminate data transfer and “NK” for negative acknowledgement. The data message sent by the Mulle is wrapped in the above mentioned data structure format. The header part of message has the following information: 1) date and time when data was sensed, 2) packet number of sensed data, 3) file format used to store the data and 4) number of samples in each data packet. The urgent flag is enabled if the data packet requires immediate delivery. The Mulle used for the implementation is equipped with a temperature sensor and hence, the data part of message contains <*temperature, voltage*> pairs. Our window-based data collection approach can be easily extended for other complex data formats. Figure 6.16 presents code snippets for the data collection function.

```
Public Function CollectData(ByVal windowSize As Integer, ByVal btDev As
BtDevice, _
    ByVal _bt As Bluetooth, ByVal mydeviceName As String) As Boolean
      Dim sensordataStore As New SensorDataStore
      Dim success As Boolean = false
      Dim sendString As String
      success = _bt.establishSerialConnection(btDev)
      If success = True Then
        sendString = "STARTD " & windowSize
         _bt.sendData(btDev, sendString, logger)
         While dataReceiveComplete = False
                Application.DoEvents()
         End While
         dataReceiveComplete = False
         _bt.disconnectSerialConnection(btDev)
         temp = FinalReceivedData.Split(";")
         For i1 = 0 To temp.Length - 2
          sensordataStore = createSensorData(btDev.DeviceName,
                                              temp(i1), mydeviceName)
         SensorDataStores.Add(sensordataStore)
         Next
      End If
      Return success
 End Function
```

Figure 6.16: Data Collection - Code Snippet

In the code presented in Figure 6.16, the sensor data collected from the sensor is stored into the *SensorDataStore* class. The *SensorDataStore* definition is presented in the detailed class diagram in Figure A1 and A2 in Appendix A. The *collector* module also implements a *Message Parser* module which is responsible for parsing and un-parsing messages sent and received by the Mulle. For example, it parses the control messages into a formatted string to be dispatched to the Mulle by the communication manager and un-parses data and control messages received from the Mulle that require further processing.

Sink Manager: The sink manager performs functions related to data delivery module of the sGaRuDa framework presented in Chapter 3. The sink manager handles communication between the *sink* and the mobile data mule. The sink manager delivers data collected from the sensor based on its importance. The sink manager is also responsible for receiving activation schedule updates from the centralised sink. The activation schedule is offloaded at the sensor during data collection. The sink manager is also responsible for periodically synchronise the sensor node repository with the *sink*. The period of synchronisation is application dependent and can be increased or decreased which will directly impact the data delivery latency. Finally, the sink manager handles mobile data mule registration and de-registration. Data exchange between the mobile data mule and the centralised *sink* is message based similar to communication between the mobile data mule and the sensor node. The communication between the sink and the mobile data mule has been implemented using sockets in both Microsoft.NET framework and in JAVA. A detailed list of classes and their functions implemented within the sink manager package is presented in Figure A1 and A2 in Appendix A. Figure 6.17 presents code snippet for the sink manager module.

```
Public Function sendSensorData(ByVal sDs As SensorDataStore) As Boolean
    Dim s As String = "SENSORDATA"
  Dim buf() As Byte =
  System.Text.Encoding.ASCII.GetBytes(s.ToCharArray())
     Dim parser As New Parser, success As Boolean=false
     If connected = True Then
       NetStream = New NetworkStream(ClientSocket)
     Try
       ClientSocket.Send(buf)
       sendData = parser.parseSensorDataStore(sDs)
       Dim buf1() As Byte =
    System.Text.Encoding.ASCII.GetBytes(sendData.ToCharArray())
       ClientSocket.Send(buf1)
     End Try
     End If
    Return success
End Function
```

Figure 6.17: Data delivery to sink - Code Snippet

Controller: The controller module represents the central controller functionality of the sGaRuDa framework presented in Chapter 3. The controller is the central command centre that co-ordinates the activity of each associated module. Communication between modules is performed by the controller which controls the sensor discover, sensor data collection and delivery process. We have implemented a support package, the *Common* package that is shared among all system modules. The variables within the *Common* package are instantiated once and are declared global. In JAVA this functionality is achieved by declaring the variables static while in Microsoft.NET, the class is defined as a public module. The controller is responsible for computing the data collection threshold based on the algorithm proposed in Chapter 3. The parameters used to compute the data collection threshold in the prototype implementation are distance, received signal strength (RSSI), residual time, sensor residual energy and amount of sensor remaining data. The RSSI is obtained during Bluetooth inquiry scan. This avoids the need to make a connection with the sensor to obtain the RSSI value. The code snippet presented in Figure 6.18 shows the threshold computation algorithm implemented in .NET CF. To calculate the threshold we have taken a packet size of 1200 bits. The controller provides functions to access the data collection platform modules functionality. For example, the *controller* module has a function to invoke the node discovery function of the *node management* module. The controller module also runs task timers to periodically check for available network connectivity to the sink to deliver urgent sensor data. In our implementation, we have used Wi-Fi as the means of data delivery. For evaluation purposes, we define a cost to the Wi-Fi connection at specific locations within our test environment. Data collected at such locations will be saved on the mobile data mule for later delivery unless the urgent flag is set to high (1). The controller module also has a context manager class that handles all available context information. In the system prototype implementation, the contextual information used are the mobile data mule's current location, its future trajectory information (used to estimate residual time) and the sensor node's context information obtained from the sensor node repository. The current location of the mobile data mule is obtained using *Ekahau positioning engine (EPE)*. The localisation module is implemented as a separate class in the JAVA implementation. In the VB.NET implementation the localisation function is integrated with the context manager.

ER1: This is the additional module that has been implemented in JAVA for the mobile robot platform. The classes in this module handle all communications between the robot hardware and the data collection framework. The ER1 modules facilitate the development of application specific plug-ins requiring control over the robot hardware. In the currently implemented system prototype no additional plug-ins has been implemented. The ER1 module is used only to control the robot's movement by making use of the default application programmable interface (API) available with the SDK. Since robot navigation and control is a vast research area by itself, our implementation does not focus on those challenges. Rather, we use existing *ER1*modules to accomplish our requirement to move the robot within the building environment. A detailed class diagram of .NET CF classes implemented on the PDA developed in VB.NET and JAVA classes implemented on the mobile robot developed in ECLIPSE (Eclipse.Org, 2010) are presented in the Appendix A.

```
Public Function computeThreshold(ByVal sEnergy As Decimal, ByVal rssi
As Decimal, _
    ByVal distance As Decimal, ByVal s_rdata As Integer)
        Dim thres As Decimal = 0, rTime, rEnergy , v_max ,
                                                       tempThreshold
        v_max = s_rdata * 1200 / _contextStore.channel_speed
        rTime = (_contextStore.ResidualTime + _contextStore.StaticTime)
        If (rTime > v_max) Then
            tempThreshold = 1
        Else
            tempThreshold = rTime / v_max
        End If
        thres = thres + tempThreshold * _contextStore.wrT
        v_max = _contextStore.Energy
        tempThreshold = sEnergy / v_max
        thres = thres + tempThreshold * _contextStore.wmE
        v_max = _contextStore.Distance
        tempThreshold = 1 - (distance / v_max)
        thres = thres + tempThreshold * _contextStore.wmD
        v_max = _contextStore.RSSI
        tempThreshold = rssi / v_max
        thres = thres + tempThreshold * _contextStore.wmR
        Return thres
    End Function
```

Figure 6.18: Data Collection Threshold Computation - Code Snippet

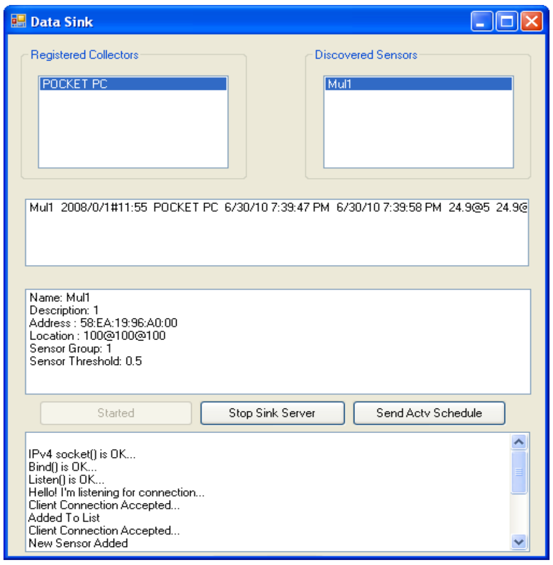
6.2.2.3 Data Sink Implementation

The sink within the scope of this thesis is assumed to be a centralised power data store that handles further processing of sensor data. In the sGaRuDa system prototype implemented, the sink implementation is to support the proof-of-concept implementation performing the following function: 1) handle mobile data mule registrations 2) receive sensor data collected and delivered by mobile data mule, 3) provide feature for sensor inactivity synchronisation among mobile data mules and 4) send new activation schedule for a single or group of sensors to the mobile data mule.

The sink has been implemented using Microsoft.NET framework developed in Microsoft Visual Basic.NET. A socket-based server has been implemented to listens to incoming client (mobile data mule) requests. Figure 6.19 presents code snippet for the incoming data processing function. The case shown in the code is when new sensor information is received by the sink. A screen shot of the sink implementation is presented in Figure 6.20 and a detailed class diagram is presented in Figure 6.21.

```
Public Sub processIncomingRequest(ByVal buf As String, ByVal nstream As
NetworkStream)
 Select Case returndata
    Case "SENSORINFO"
    sendAck(nstream)
    Do While nstream.DataAvailable
         byte_buff = nstream.ReadByte()
         buffer = buffer & Convert.ToChar(byte_buff)
    Loop
    parser.initialize(buffer)
    Dim thread2 As New System.Threading.Thread(AddressOf
                                                parser.newSensor)
    thread2.Start()
End Sub
```

Figure 6.19: Sink Incoming Data Request Processing - Code Snippet



```
Mul1  2008/0/1#11:55  POCKET PC  6/30/10 7:39:47 PM  6/30/10 7:39:58 PM  24.9@5  24.9@
```

```
Name: Mul1
Description: 1
Address : 58:EA:19:96:A0:00
Location : 100@100@100
Sensor Group: 1
Sensor Threshold: 0.5
```

```
IPv4 socket() is OK...
Bind() is OK...
Listen() is OK...
Hello! I'm listening for connection...
Client Connection Accepted...
Added To List
Client Connection Accepted...
New Sensor Added
```

Figure 6.20: Sink implemented in VB.NET – Screenshot

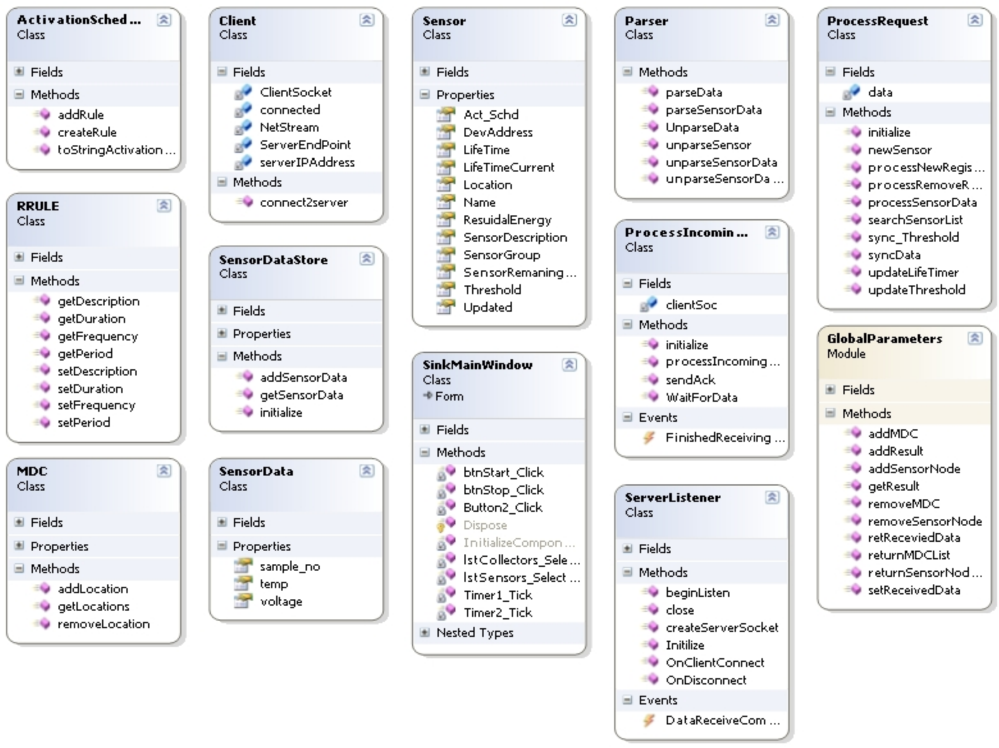


Figure 6.21: Detailed Class Diagram - Sink Implementation

6.3 *k*-Nearest Neighbour Based Sensor Data Collection - Implementation

Chapter 4 presented our proposed *3D-KNN* sensor data collection algorithm. The *3D-KNN* algorithm targets sensor nodes with broadcasting capabilities. The introduction of broadcasting channel facilitates multi-hop data collection. The *3D-KNN* data collection algorithm can be incorporated straight-forwardly into sGaRuDa. To evaluate the *3D-KNN* algorithm over a large-scale sensor network, we chose a simulator environment. The simulator environment allows us to validate the cost-efficiency of the proposed *3D-KNN* algorithm under varying system parameters, namely, number of sensor nodes, size of the area, radio range, signal-to-noise, etc. The *3D-KNN* algorithm has been simulated in the simulator *Global Mobile Information System Simulator* (GloMoSim) (GloMoSim, 2010, Nuevo, 2004). In this section, we present an overview of the simulation platform followed by *3D-KNN* algorithm's implementation details.

6.3.1.1 A Scalable Simulation Environment: GloMoSim

GloMoSim is a scalable parallel discrete event simulator for large-scale wireless and wired networks. It uses Parsec (UCLA, 2009), a C based language developed by the Parallel Computing Laboratory at UCLA. GloMoSim has the capacity to simulate wireless sensor networks with up to thousand nodes. It allows simulating multi-hop wireless communication using ad-hoc networking. The simulator is designed using the layered Open Systems Interconnection (OSI) network approach. Each layer of the OSI model is implemented by a set of C files. The functions performed at each layer can be summarised into 1) send/receive data between layers 2) perform layer level functionality i.e. physical layer simulates sending data as bits while the network layer simulates the operation of assembling/disassembling data packets and 3) facilitate data exchange between nodes within the network.

GloMoSim supports node mobility using random waypoint, random drunken and group mobility models (Nuevo, 2004). The mobile data mule's mobility has been simulated using this feature of GloMoSim. A configuration file *config.in* is used to setup the simulation environment. Figure 6.22 presents a sample of the configuration file. GloMoSim allows sensor nodes to be simulated within a three-dimensional (3D) space as defined by the terrain-dimension parameter in Figure 6.22. Each sensor node's location within the 3D space is defined using x, y, and z co-ordinates. GloMoSim uses two types of messages to facilitate communication within layers and between nodes. They are *non-packet* and *packet messages*. The *non-packet* messages are used for inter-layer event messages and self-schedule timer events. The *packet messages* are used to send data messages across layers or across various nodes.

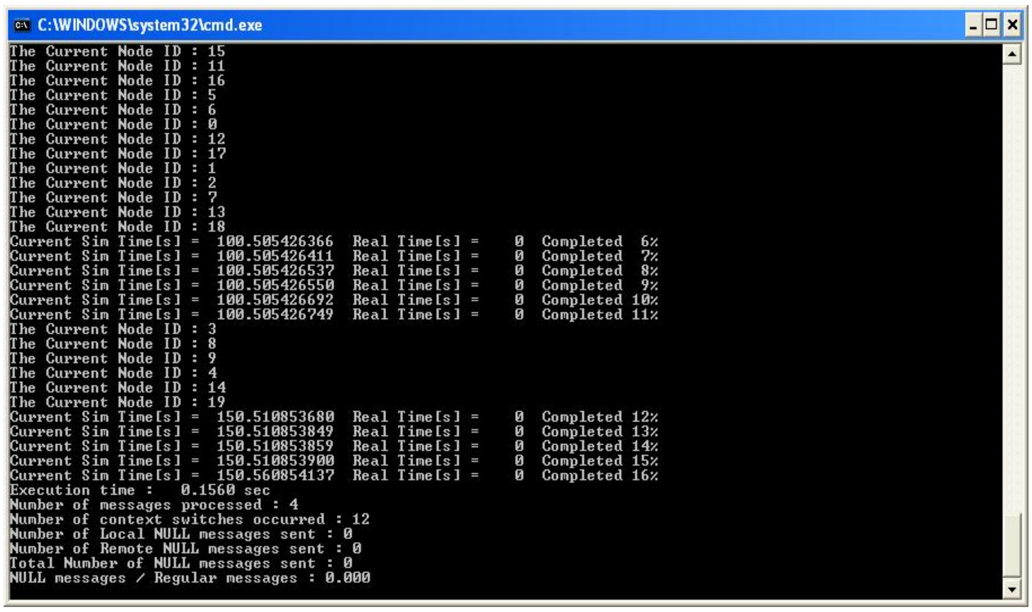
```
SIMULATION-TIME     15M
SEED                1
TERRAIN-DIMENSIONS  (2000, 2000, 2000)
NUMBER-OF-NODES     200
NODE-PLACEMENT      UNIFORM
```

Figure 6.22: GloMoSim Sample Configuration File

The *non-packet* messages used to enable self-schedule timers provide the capability to implement in-network data aggregation function. Figure 6.23 illustrates a sample message structure used during data exchange between nodes and between layers. The code snippet provided in Figure 6.23 is extracted from the *message.h* file in the *include* folder. Mobility is achieved using a mobility trace file mentioned in the *config.in* file. The trace file can be used to mention specific movement patterns for the mobile data mule. We use this feature to incorporate mobile data mule movements. Figure 6.24 presents a screen dump of the GloMoSim command prompt-based simulation environment. All simulation outputs are written to the *glomo.stat* text file.

```
struct message_str
{
short layerType; // Layer that receives the message
short protocolType, eventType;
char* packet, payLoad;
} message;
```

Figure 6.23: Message Structure used in Data Exchange



```
The Current Node ID : 15
The Current Node ID : 11
The Current Node ID : 16
The Current Node ID : 5
The Current Node ID : 6
The Current Node ID : 0
The Current Node ID : 12
The Current Node ID : 17
The Current Node ID : 1
The Current Node ID : 2
The Current Node ID : 7
The Current Node ID : 13
The Current Node ID : 18
Current Sim Time[s] =  100.505426366  Real Time[s] =    0  Completed  6%
Current Sim Time[s] =  100.505426411  Real Time[s] =    0  Completed  7%
Current Sim Time[s] =  100.505426537  Real Time[s] =    0  Completed  8%
Current Sim Time[s] =  100.505426550  Real Time[s] =    0  Completed  9%
Current Sim Time[s] =  100.505426692  Real Time[s] =    0  Completed 10%
Current Sim Time[s] =  100.505426749  Real Time[s] =    0  Completed 11%
The Current Node ID : 3
The Current Node ID : 8
The Current Node ID : 9
The Current Node ID : 4
The Current Node ID : 14
The Current Node ID : 19
Current Sim Time[s] =  150.510853680  Real Time[s] =    0  Completed 12%
Current Sim Time[s] =  150.510853849  Real Time[s] =    0  Completed 13%
Current Sim Time[s] =  150.510853859  Real Time[s] =    0  Completed 14%
Current Sim Time[s] =  150.510853900  Real Time[s] =    0  Completed 15%
Current Sim Time[s] =  150.560854137  Real Time[s] =    0  Completed 16%
Execution time :   0.1560 sec
Number of messages processed : 4
Number of context switches occurred : 12
Number of Local NULL messages sent : 0
Number of Remote NULL messages sent : 0
Total Number of NULL messages sent : 0
NULL messages / Regular messages : 0.000
```

Figure 6.24: GloMoSim Simulator Environment - Screen Dump

6.3.1.2 3D-KNN - Implementation in GloMoSim

In this section, we present the implementation details of the *3D-KNN* algorithm in GloMoSim. The *3D-KNN* has been implemented at the network layer of GloMoSim. Hence, packet processing and response happen at the network layer. This implementation can be moved to the lower layers, namely, the data link layer as our current implementation does not rely on any network layer specific functionality. Node addressing is achieved using low-level naming i.e. *node-identification number*. The following key functions have been simulated using GloMoSim.

1) A mobile data mule responsible for issuing and processing *kNN* queries in a 3D environment.
2) Mobility model to support the mobile data mule's movement within the sensor network
3) Sensor nodes that respond to incoming *kNN* queries using in-network aggregation.
4) Scripts to generate outputs for the *3D-KNN* algorithm's evaluation.

Figure 6.25 presents the *3D-KNN* algorithm's modules implemented in GloMoSim. The files in GloMoSim have a *.pc* extension due to the use of the Parsec. Though the extension used is parsec, the coding follows C syntax. The modules highlighted in dark green are the primary *3D-KNN* modules that have been newly developed and implemented. The modules highlighted in light green are existing code that has been modified for the *3D-KNN* implementation. The functions of the sensor and the mobile data mule have been implemented in the *sensor.pc* and *mdm.pc* files. The *timer.pc* is used for in-network data aggregation. The primary modules shown on the right in Figure 6.25 are invoked by both sensor and the mobile data mule to perform various operations including sending/receiving broadcast messages, processing incoming data, responding to incoming data, computing nearest neighbours using the *KNN-METRIC* and collecting data from nearest sensors. The *knn.h* is the common header file used by the sensor and the mobile data mule code to invoke functions defined in various modules. We describe the functionality of each module used by the sensor and the mobile data mule subsequently.

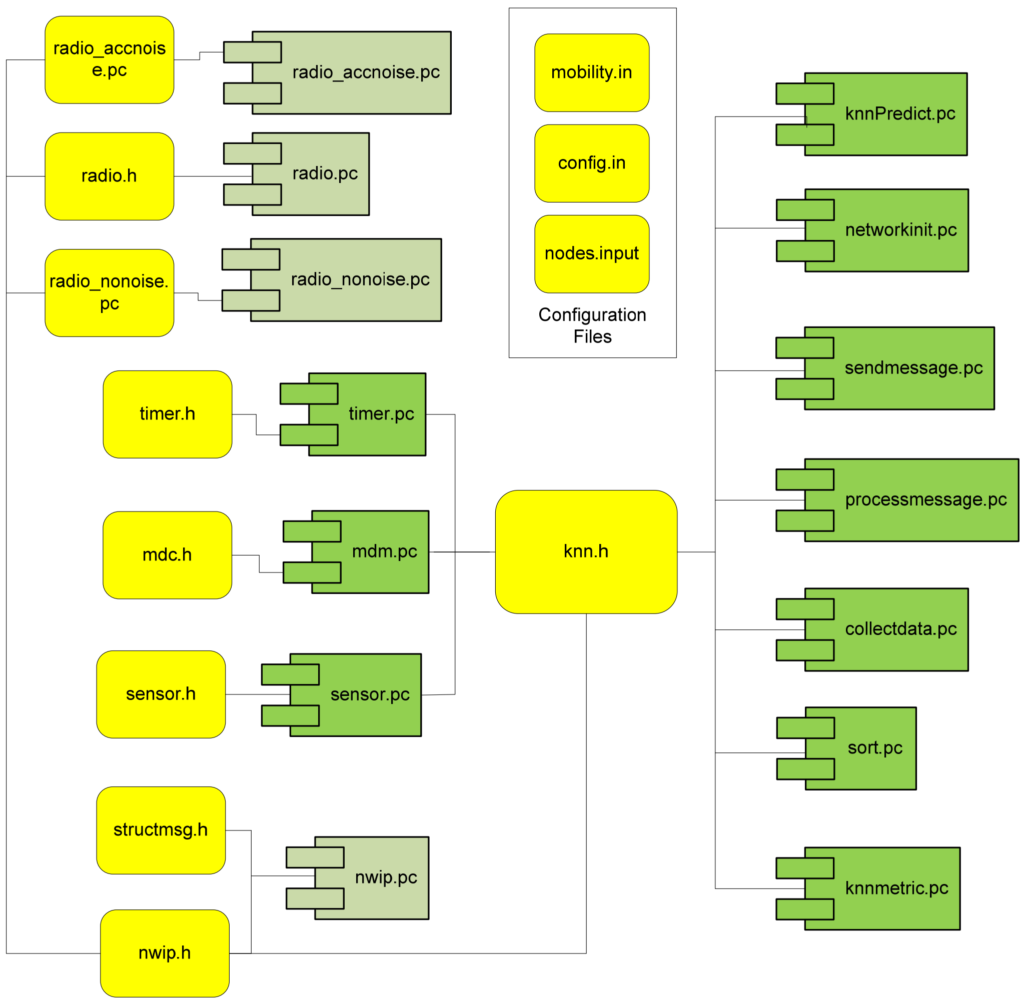


Figure 6.25: 3D-KNN GloMoSim Simulation Modules

networkinit.pc: This module contains code to initialise the sensor network. It provides functions to select the mobile data mule within the simulator environment. Every other node within the network is then initialised as a sensor node. The initial broadcast message propagated by the mobile data mule is controlled by the *networkinit.pc* module. The *networkinit.pc* is also responsible for setup files required to produce simulation outputs.

sendmessage.pc: The *sendmessage.pc* modules handle sending broadcast and uni-cast messages. The broadcast message is sent with an address to all nodes that are within the radio range. The uni-cast mode is used when the route to a particular sensor node is available. It is used during the final data collection phase. Figure 6.26 presents the broadcast message and the data packet structures. Since our approach does not require any network route information, the broadcast message is used to compute routes between sensors and the mobile data mule. Each broadcast packet has a *nodeList* variable. This variable is appended with sensor route information (route through which the broadcast information propagated). The reverse-path is used to identify the route to the mobile data mule. The data packet comprises sensor node's location and cumulative signal-to-noise ratio. Figure 6.27 and Figure 6.28 presents code snippets for sending broadcast and uni-cast messages in GloMoSim. The *NetworkIpSendNewPacketWithDelay* function is used to send broadcast/uni-cast packets. The destination type determines the type of packet i.e. *ANY_DEST* represents a broadcast packet and *dest_addr* represents a uni-cast packet addressed to a specific sensor node. The two message structures presented are used during the *kNN* boundary estimation phase. The data packet with sensor information is used by the mobile data mule to compute nearest neighbours.

```
typedef struct {
 int bcastId, maxHop, nodecount,
 int nodeList[300];
 double x, y, z;
} intial_broadcast_pckt;
typedef struct {
 int retId; //same as the intial bcastId
 int nodeCount, totalnodes;
 int route[nodeCount];
 double x[nodeCount], y[nodeCount], z[nodeCount];
 double snr;
} data_pckt;
```

Figure 6.26: 3D-KNN Implementation - Message Packet Structures - Code Snippet

```
void handleSensorBroadcast(GlomoNode *node, Message *msg)
{ msg=GLOMO_MsgAlloc(node,GLOMO_NETWORK_LAYER,NETWORK_PROTOCOL_I
 P,MSG_NETWORK_BroadcastEvent);
 GLOMO _MsgSend(node,msg,ADVT_INTERVAL);
 NetworkIpSendNewPacketWithDelay(node,ANY_DEST,REAL_TIME
 ,SENSOR_PROTO,IPDEFTTL,(char
*)bcast_pkt,sizeof(intial_broadcast_pckt),(clocktype)      DELTA_TIME
* node->nodeAddr/2);
}
```

Figure 6.27: 3D-KNN Implementation - Sending Broadcast Message - Code Snippet

```
void sendUniCast(GlomoNode *node, Message *newMsg){
 data_pckt* senddata;
 int i = 0;
 NODE_ADDR dest_addr;
 senddata = (data_pckt* )GLOMO_MsgReturnInfo(newMsg);
 senddata->nodeCount = senddata->nodeCount - 1;
 dest_addr = senddata->route[senddata->nodeCount];
 NetworkIpSendNewPacketWithDelay(node,dest_addr,REAL_TIME,SENSOR_D
 ATA,IPDEFTTL,(char *)senddata,sizeof(data_pckt),(clocktype)
 DELTA_TIME * node-     >nodeAddr/2);
}
```

Figure 6.28: 3D-KNN Implementation - Sending Uni-cast Message - Code Snippet

processmessage.pc: This module performs incoming message processing. The incoming messages can be classified into two types broadcast messages and data messages. The data message can be a response to a broadcast message or a response to a data collection request. The *nwip.pc* file handles all incoming messages from the MAC layer. Once the message is received at the network layer, (*nwip.pc*) it calls the appropriate function defined in the *processmessage.pc* module to process the message. Figure 6.29 presents code snippet of the modified *nwip.pc* file. The functions defined in the *processmessage.pc* module are shared by the mobile data mule and the sensor node. The primary functions defined are *handleSensorBroadcast, handleSensorReBroadcast* and *sendUniCast*. The first two functions perform the following operations: 1) send initial broadcast message (data mule operation) and 2) receive and forward broadcast messages (sensor operation). The *sendUniCast* function is used by both mobile data mule and the sensors to respond to *kNN* queries. These functions facilitate the *3D-KNN* pre-routing and data collection phases. Figure 6.30 provides code snippets of the incoming broadcast information handling function.

collectdata.pc*:* The *collectdata.pc* module is used by the mobile data mule to compute the nearest neighbours based on the *KNN-METRIC*. It employs the functions defined in the *sort.pc* module to sort the nearest neighbour sensor list. The *knnmetric.pc* module is used to compute the *KNN-METRIC* of each sensor. The *sendmessage.pc* module is employed by the *collectdata.pc* module to send modified uni-cast messages to the *k* nearest sensors. The modified uni-cast uses the route information collected during the boundary estimation phase to route data to specific nodes (over multi-hop channels). Since the cumulative signal-to-noise ratio is used in the *KNN-METRIC* any sensor which is part of a poor route will not be selected as the nearest neighbour. Further, the *collectdata.pc* also uses the proposed prediction algorithm to determine sensors that are future nearest neighbours. Once at the new location, the *sendmessage.pc* is employed to collect data from those sensor nodes.

```
void NetworkIpLayer(GlomoNode *node, Message *msg) {
  switch (msg->protocolType) {
  case NETWORK_PROTOCOL_IP: {
  switch(msg->eventType) {
      case  MSG_NETWORK_BroadcastEvent: {
            handleSensorBroadcast(node, msg);
            break;                  }
      case  MSG_NETWORK_ReBroadcastEvent: {
            handleSensorReBroadcast(node, msg);
            break;                  }
      case  MSG_NETWORK_DATA: {
            sendUniCast(node, msg);
            break;            }}
}
```

Figure 6.29: 3D-KNN Implementation - nwip.pc - Code Snippet

```
void ProcessIncomingMessage(GlomoNode *node, Message *msg, NODE_ADDR
sourceAddress, double snr){
 Message *newMsg;
 intial_broadcast_pckt* info;
 info = (intial_broadcast_pckt*)GLOMO_MsgReturnPacket(msg);
 nodeAddress = node->nodeAddr;
 if (broadCastReceivedID != info-> bcastId)     {
            broadCastReceivedID = info-> bcastId;
  newMsg = GLOMO_MsgAlloc(node,GLOMO_NETWORK_LAYER,
 NETWORK_PROTOCOL_IP,MSG_NETWORK_ReBroadcastEvent);
  GLOMO_MsgInfoAlloc(node, newMsg,sizeof(intial_broadcast_pckt));
  resendinfo = (intial_broadcast_pckt* ) GLOMO_MsgReturnInfo(newMsg);
  GLOMO_MsgSend(node,newMsg,ADVT_INTERVAL);           }
}
```

Figure 6.30: 3D-KNN Implementation - Checking Incoming Broadcast Message - Code Snippet

timer.pc*:* The *timer.pc* module is used for in-network data aggregation. The *non-packet* messages described earlier are used to implement timers within GloMoSim. The *non-packet* messages used in timers are self-addressed i.e. a loop-back. After the specified delay, the message is delivered back to the sensor node. The arrival of a new timer message indicates the timer expiry event. Figure 6.31 presents the timer implementation code used by the sensor node.

```
void startTimer(GlomoNode *node, clocktype timerDelay){
 Message *newMsg;
 timer_details* timer_pkt;
 newMsg=GLOMO_MsgAlloc(node,GLOMO_NETWORK_LAYER,NETWORK_PROTOCO
 L_IP,MSG_TIMER_EVENTS);

 GLOMO_MsgInfoAlloc(node, newMsg, sizeof(timer_details));
 timer_pkt = (timer_details* )GLOMO_MsgReturnInfo(newMsg);
  timer_pkt->nodeID = node->nodeAddr;
 timer_pkt->state = 0;
  GLOMO_MsgSend(node, newMsg, timerDelay);
}
void checkTimerTimeout(GlomoNode *node, Message *newMsg){
 timer_details* newTimer;
 newTimer = (timer_details*)GLOMO_MsgReturnInfo(newMsg);
 if (newTimer->nodeID == node->nodeAddr)  {
  processTimer(node, newMsg);
 }
}
```

Figure 6.31: 3D-KNN Implementation - Timer Implementation on Sensor - Code Snippet

radio.pc: The *radio.pc, radio_nonoise.pc* and *radio_accnoise.pc* have been modified to implement a radio model that accurately measures energy consumption of wireless ad-hoc sensor nodes. This model has been implemented by Margi et al. (Margi et al., 2006) and the source code is available online. The default radio model of GloMoSim is oriented towards high-powered ad-hoc mobile devices. The *mobility.in*, *config.in* and *nodes.input* files are used to configure mobile data mule's mobility pattern, simulation environment parameters and node placements within the sensor network respectively.

6.4 R-CS - Implementation

In this section, we present the implementation details of Ranked-Context Spaces (R-CS) system. We use the term *ranked* to denote the partitioning approach that uses context attribute ranks. The implementation of R-CS is built around the Context Spaces reasoning engine. The following features have been implemented in R-CS. They are:

1) The situation partitioning algorithm used to partition the universal situation space.

2) Incorporating sensor data quality parameters and flexible attribute region definitions for the context attributes. Algorithm to compute *contribution* based on flexible attribute region and sensor data quality.

3) Incorporation of hierarchical attribute region definition and algorithm to reason hierarchical attribute regions.

4) A graphical user interface that provides the user the ability to define/simulate sensor inputs, situation spaces and context attribute regions.

5) Scripts to generate outputs of the reasoning process used in R-CS evaluation.

Context Spaces has been implemented in JAVA and hence, the extended Context Spaces implementation, namely, R-CS is also implemented in JAVA. We have used the CORE (Padovitz, 2006) reasoning engine to implement R-CS features. To implement a flexible attribute region, we have used a basic logic that provides a region contribution based on the context state value of the context attribute. Our implementation defines outer regions as $\pm x$ where x is the flexible region value. For example, if the attribute region definition is 2 to 4, and the flexible region definition is ± 0.5, any context state value for the context attribute within the range 2 to 4 results in a contribution of 1 while any value within the region 1.5 to 2 and 4 to 4.5 will result in a contribution value from 0 to 1. To determine a value between 0 and 1 we used fuzzy logics.

The overall JAVA package diagram for R-CS implementation is presented in Figure 6.32. The primary Context Spaces (CS) implementation is defined by the package core. The *core.kernel, core.model* are Context Spaces implementation that have been modified to R-CS functions while the *core.gui* and *corePrxy* are R-CS specific implementations. A detailed class diagram of the implementation is presented Figure A3 in Appendix A.

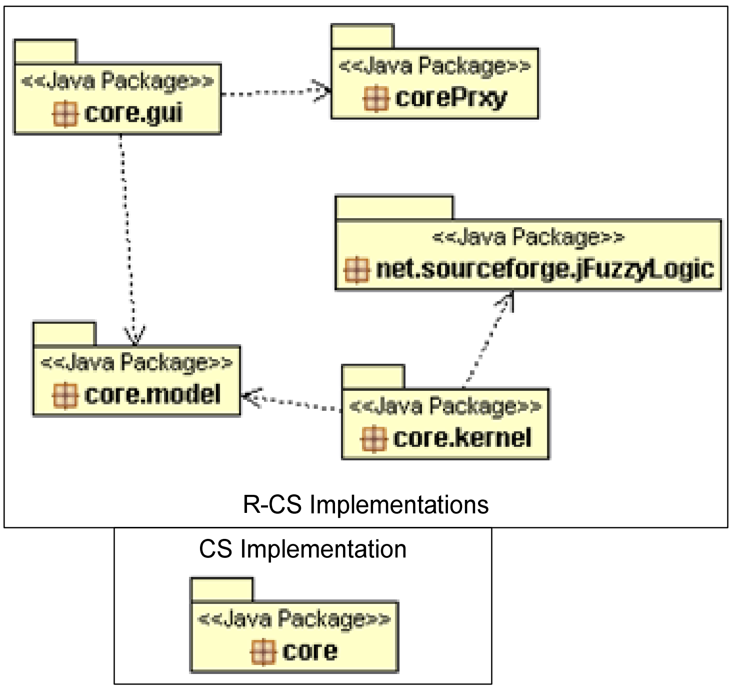


Figure 6.32: R-CS Implementation - Package Diagram

Situation Space Partition: The situation space partitions are created by implementing the following classes *partition.java* and *partitionReasoner.java.* The context attribute regions defined in the universal situation space are partitioned by *partition.java* class and the *partitionReasoner.java* is used during reasoning process i.e. taking into consideration each partition until the required confidence threshold is reached. Figure 6.33 presents the situation partition code snippet.

```
public XSpace partition(XSpace sp){
 Vector r = sp.getRegions();
 for (int i=0; i< r.size(); i++){
      XAttributeRegion xa = (XAttributeRegion) r.get(i);
      double w = xa.getWeight();
      for (int j=0;j<pregionList.size(); j++){
            Vector tempRegion = (Vector) pregionList.get(j);
            if (computeRegion(tempRegion, w)){
                  xa.setPar_region(j + 1 );           }}
      System.out.println("Weight : " + xa.getWeight());
      System.out.println("Region based on Weight : " +
xa.getPar_region());    }
 sp.setPregions(region_no);
 return sp; }
```

Figure 6.33: R-CS Implementation - Situation Partition - Code Snippet

The code in Figure 6.33 uses the partition predicate definition to partition the universal situation space. The code in Figure 6.34 reasons the temporal situation space computed dynamically from the partitioned universal situation space. The universal situation space partitions determined by the partition predicate are maintained in the universal situation space definition. At runtime, to reason a current situation, each partition in the universal situation space is iteratively explored to compute the overall confidence. New partitions in the universal situation space are only explored if the required confidence threshold is not reached using the context attribute within the current partition.

```
public Vector reason(XMatrix x, boolean ReReason){
 XMatrix y = null;
 if (ReReason)
   y = reEncodeMatrix(x);
 else
  y = x;
 //We partition the situation space based on Importance
 XMatrix matrix = partitionSituationSpace(y);

 Vector Matrix;
 Matrix = matrix.getMatrix();
 for (int j=0;j< Matrix.size();j++) {
  XStConf xst = null;
  double sum = 0;
  Vector tempMatrix = (Vector) Matrix.get(j);
  XDecoderNew decoder = new XDecoderNew(tempMatrix);
  for (int ii=1; ii<=part.getRegion_no();ii++){
       xst  = decoder.decode(st,ii, ReReason);
       sum = sum + xst.getConfidence();
  }
  sum = sum/part.getRegion_no();
  xst.setConfidence(sum);
  reasonedList.add(xst);
 }
 return (computeMaxMin());
}
```

Figure 6.34: R-CS Implementation: Partition-based Situation Reasoning

The dynamic weight re-computation of context attributes for orthogonal situations (situations that cannot occur in parallel) is performed during the reasoning process. The weight re-computation increases the reasoning outcome’s confidence by reducing the influence of context attributes whose context state value satisfies situations that are orthogonal. The situation space referred in this instance is the temporal situation space. As new context attributes from other partitions are added to the temporal situation space, the weight re-computation algorithm is repeated to reduce the influence of overlapping context attributes. The code snippet that computes similarities between situations is presented in Figure 6.35. The code compares attribute regions of situations to compute the situation similarity co-efficient. The attribute region similarity value is appended to each attribute region definition which is further used during the weight re-computation process.

```
public XSpace compareSituations(XSpace st1, XSpace st2, XState st) {
 XSpace tmpSitutaion = new XSpace(st1.getName(),st1.getThreshold());
 Vector attrs = st1.getRegions(), attrs1 = st2.getRegions();
 double contribution =0,  contribution1 = 0;
 for (int i = 0; i< attrs.size() ; i++) {
  boolean similar = false;
   XAttributeRegion region1 =   (XAttributeRegion)attrs.get(i);
  XAttributeRegion r = region1.copyValue();
  XAttribute attr = (XAttribute) st.hattributes.get(region1.getName());
  if (attr != null)
       for (int j=0; j< attrs1.size(); j ++) {
             XAttributeRegion region2 = (XAttributeRegion)attrs1.get(j);
             if (region1.getName() == region2.getName())
                   contribution1 = computeContribution(st);
             if (contribution == 1)
                   ComputeSimilarityCoefficient();
                   similar = true;   }}
  if (similar)
       tempSitutaion.addAttributeRegion(r,  r.getWeight(),
r.getOptional(),true);
  else
       tempSitutaion.addAttributeRegion(r, r.getWeight(),
r.getOptional(),false);
  }
  return tempSitutaion;
}
```

Figure 6.35: R-CS Implementation - Situation Comparison Code Snippet

Flexible region and sensor inaccuracy heuristics implementation: These two features implemented in R-CS incorporate flexible attribute region definition and sensor data quality (data freshness) heuristics. The data freshness considers the life of the collected sensor data during the reasoning process. The current context attribute (sensor) definition in Context Spaces has been re-implemented to incorporate the data freshness heuristic. To implement data freshness in R-CS, we have incorporated a freshness threshold. This value determines if the data is recent or old. Figure 6.36 presents the modified context attribute definition code snippet with the proposed data freshness threshold.

```
  public XAttribute(String attributeName, String value, double
errorProb, double freshness)
  {
 this.attributeName = attributeName;
 this.textValue = value;
 this.errorProb = errorProb;
 this.freshness = freshness;
 Date dt = new Date();
 entryTimeStamp = dt.getTime();
 this.errorRange = 1;
  }
```

Figure 6.36: R-CS Implementation - Context Attribute Definition

To implement flexible attribute regions, inner-outer regions are defined for each context attribute. Based on the context attribute region definition, the contribution value of the context attribute within the situation is computed. The contribution of a context attribute determines the influence (confidence) of the context attribute's in the reasoned situation. For implementation and testing purposes we have assigned the inner-outer region value for all context attributes ±0.5. This assumption can be modified allowing greater control over the reasoning process. Figure 6.37 presents the code snippet to compute the context attribute's contribution when the context state value falls within the inner-outer region.

```
public double getContributionOuterRegions(double value) throws
PredicateException  {
 for (int i = 0; i < subregions.size(); i++) {
  XSubRegion subregion = (XSubRegion) subregions.elementAt(i);
  double contribution = subregion.getContributionOuterRegions(value);
  if (contribution != 0)
       return contribution;    }
 return 0; }
```

Figure 6.37: R-CS Implementation - Flexible Context Attribute Regions

Hierarchical Context Attribute Region Definition: Hierarchical context attribute regions allow definition relations between situation space definitions. For example, the defining relationship between the context attribute *speed* and context attribute *age*. To implement hierarchical context attribute regions, we have modified the attribute region definition in CS to incorporate hierarchical definitions i.e. the ability to add one context attribute region as a child to another context attribute region. To compute the cumulative contribution of the hierarchical context attribute regions, the contribution value of every individual child attributes is computed iteratively. The cumulative contribution of the parent context attribute is then computed as the sum of child context attribute contributions. Each child context attribute region is assigned a weight determining its influence in the total situation outcome.

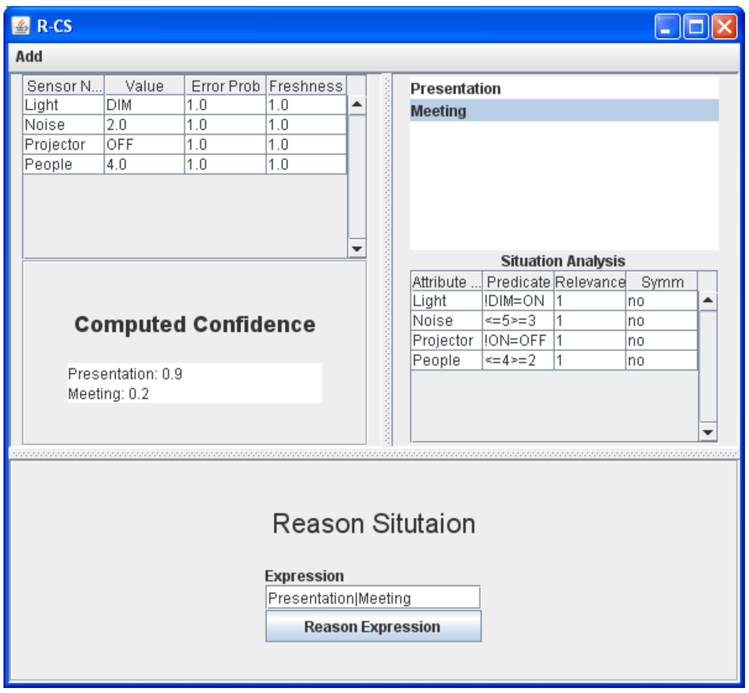
```
//Attribute Region Definitions
XAttributeRegion region1 = new XAttributeRegion("Age", true);
Predicate p1 = new Predicate(">=", "20");
Predicate p11 = new Predicate("<=", "30");
Predicate[] predicate1 = { p1, p11 };
region1.addSubRegion(predicate1, 1);

//Define Sub Region
XAttributeRegion region2 = new XAttributeRegion("Speed");
Predicate p2 = new Predicate("<=", 3);
Predicate p21 = new Predicate(">=", 5);
Predicate[] predicates2 = { p2, p21 };
//Add Subregion to parent Region
region1.addSubRegion(predicates2, 1);

//Defining Attribute SubRegions
public void addSubRegion(XAttributeRegion subDim, double weight) {
  subdim.add(subDim);
  subDim.setWeight(weight);
}
```

Figure 6.38: R-CS Implementation - Hierarchical Context Attribute Region Definition

Finally, we have implemented a graphical user interface that allows users to test and evaluate the proposed R-CS system. The interface allows the user to 1) define situation spaces with corresponding attribute regions; 2) define context attributes (sensor) with their corresponding context state values and 3) reason different situations using different expressions e.g. *Presentation or Meeting*. The user interface also has the capability to simulate a stream of synthetic context state values for corresponding context attributes (sensors). Figure 6.39 presents a screen shot of R-CS user interface. The user interface code also produces text file-based outputs (dumps) used for evaluation.


R-CS

Add

Sensor N...	Value	Error Prob	Freshness
Light	DIM	1.0	1.0
Noise	2.0	1.0	1.0
Projector	OFF	1.0	1.0
People	4.0	1.0	1.0

Presentation
Meeting

Computed Confidence

Presentation: 0.9
Meeting: 0.2

Situation Analysis

Attribute ...	Predicate	Relevance	Symm
Light	!DIM=ON	1	no
Noise	<=5>=3	1	no
Projector	!ON=OFF	1	no
People	<=4>=2	1	no

Reason Situtaion

Expression

Presentation|Meeting

Reason Expression

Figure 6.39: R-CS Implementation - User Interface

6.5 Summary

This chapter has presented implementations details of the mobile data mule-based sensor data collection approaches proposed in Chapters 3 and 4, namely, sGaRuDa system framework and 3D-KNN algorithm and the Context Spaces extension, R-CS system proposed in Chapter 5.

A prototype of the sGaRuDa sensor data collection framework has been implemented on real-world mobile devices, namely, PDA and an ER1 mobile robot. The development platform used was Microsoft.NET Compact Framework (CF) and JAVA. We used VB.NET to develop code in the Microsoft.NET CF platform. The wireless sensor node chosen for the prototype implementation was the Bluetooth Mulle. The code for the Mulle sensor node was developed in C using the proprietary IAR compiler. Finally, we implemented a crack-down version of a centralised data sink that acts as the destination for collected data. The sink was developed in VB.NET. The implementation of the sGaRuDa framework proved the real-world feasibility of the proposed data collection architecture. The sGaRuDa system facilitates sensor data collection using real-world mobile devices without the need for a dedicated data collection infrastructure.

Further, we presented the implementation details of the proposed *3D-KNN* algorithm. The *3D-KNN* algorithm was implemented and simulated in GloMoSim, a parallel discrete event simulator. Simulating *3D-KNN* in GloMoSim allowed us to evaluate the *3D-KNN* algorithm within a three-dimensional large-scale sensor network (up to 1000 sensor nodes).

Finally, we presented implementation details of the proposed R-CS system. R-CS i.e. ranked Context Spaces was developed in JAVA by wrapping the existing Context Spaces functionality already implemented using JAVA. We also implemented a simple graphical interface allowing us to evaluate the R-CS. The proposed R-CS algorithms were incorporated in R-CS by developing new functionality and modifying existing Context Spaces functionalities.

7

Evaluation of Implemented sGaRuDa, 3D-KNN and R-CS

7.1 Introduction

In this chapter, we evaluate the proposed sensor data collection framework sGaRuDa, the 3D-KNN algorithm and the R-CS modelling approach. This chapter is divided into three main sections focusing on the three main contributions of this thesis. In each section, we present evaluation criteria followed by experimental results. The three sections of this chapter are:

1) Evaluating sGaRuDa, the practical system framework for sensor data collection using mobile data mules targeted at Bluetooth-based sensor networks. Some parts of the system evaluations presented in section 7.2 are from the following published papers (Jayaraman et al., 2007, Jayaraman et al., 2008a)
2) Evaluating 3D-KNN, a *k*-Nearest Neighbour-based data collection algorithm using mobile data mules targeted at broadcast-based sensor networks. The evaluations of the 3D-KNN algorithm presented in section 7.3 are from the following published papers (Jayaraman et al., 2010a, Jayaraman et al., 2010c, Jayaraman et al., 2010b)
3) Evaluating R-CS, the dynamic situation-based smart spaces modelling approach based on Context Spaces. The evaluation of R-CS presented in section 7.4 are from the following papers (Jayaraman et al., 2009b, Jayaraman et al., 2009a)

7.2 sGaRuDa: Implementation Evaluation

The key research question that we tried to answer in this thesis is the feasibility of using mobile devices as a cost-efficient alternative for sensor data collection. In this section, we present experimental and evaluation outcomes validating the feasibility and cost-effectiveness of the sGaRuDa framework. We demonstrate the feasibility of the sGaRuDa framework in real-world scenarios. To validate the cost-efficiency of the sGaRuDa framework, we perform experiments to evaluate the proposed data collection and dynamic activation schedule algorithms. The cost-based evaluation comprises the following components:

1) Energy consumed by sensors during sensor data collection
2) Time spent in discovery i.e. percentage of time the mobile data mule is able to successfully discover and collect data from neighbouring sensors.

Our test scenario used for system evaluations is a building environment (scenario) depicted in Figure 7.1. The building environment is a typical example of a real-world pervasive environment with abundant availability of mobile devices.

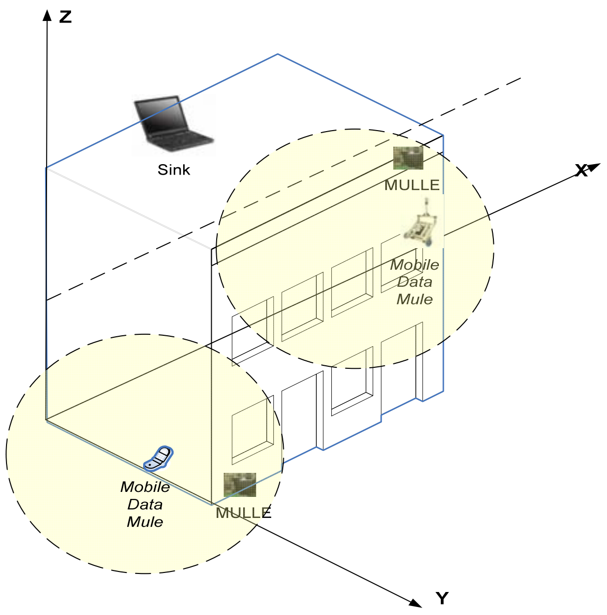


Figure 7.1: A Building Scenario used for Evaluation

Further, a building environment accurately captures our notion of three-dimensional sensor network deployment. Another reason to support the choice of environment is the recent need for environmental monitoring and control in building especially in large offices. Research shows that average cost of installing a single wired sensor in a building is $200 (Rabaey et al., 2000). Further, such deployments require a fixed data collection infrastructure to collect sensor data. This further adds to the deployment cost. The sGaRuDa system is a cost-efficient alternative as wireless sensors like Mulle can be deployed with relative ease without the need for a fixed data collection infrastructure. In such a scenario mobile devices that are within the office space (environment) can be used as mobile sensor data collectors.

7.2.1 Dynamic Activation Schedule: Implementation Evaluation

In this section, we evaluate the implementation of the dynamic activation schedule. We present a scenario that updates the activation schedule of Mulle dynamically using the mobile data mule. Figure 7.2 presents a flow diagram of the operations involved between the mobile data mule, the sink and the sensor node (Mulle).

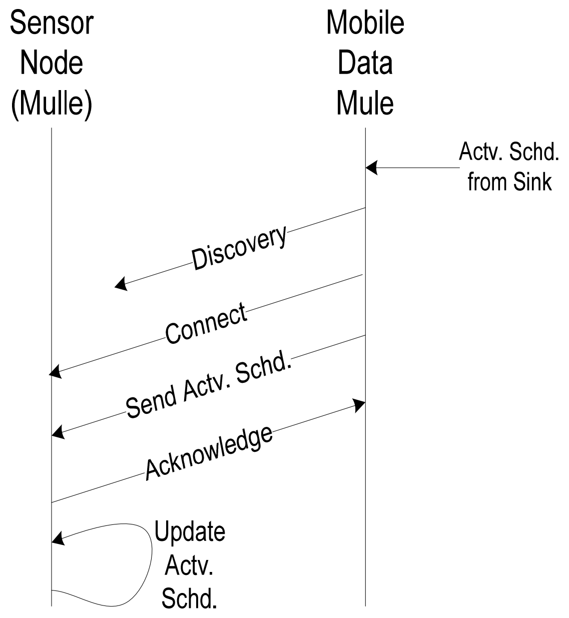
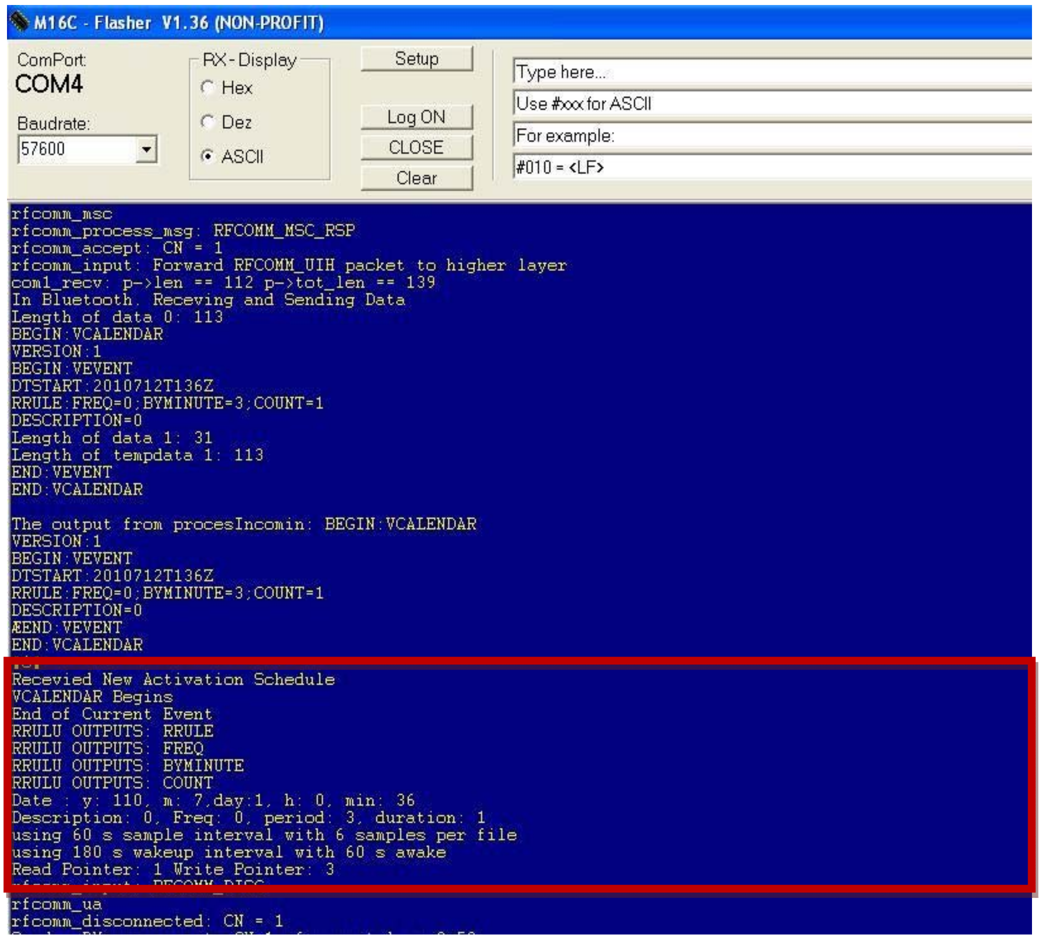


Figure 7.2: Activation Schedule Update - Flow Diagram

The mobile data mule used for the experiment was the personal digital assistant (PDA). The Mulle was connected to its expansion board to obtain real-time outputs. The outputs from the Mulle sensor node and the mobile data mule are presented in Figure 7.3 and Figure 7.4. For evaluation and illustration purpose, we have implemented menu controls on the PDA for various sensor operations. Figure 7.3 depicts output from the Mulle in real-time. For this experimentation, the activation schedule comprised the schedule presented in Table 7.1.

Bluetooth Radio Interval	Sense Interval
180s wake-up, 60s awake	Sample every 60s. 6 samples per file

Table 7.1: Activation Schedule Experiment



```
rfcomm_msc
rfcomm_process_msg: RFCOMM_MSC_RSP
rfcomm_accept: CN = 1
rfcomm_input: Forward RFCOMM_UIH packet to higher layer
com1_recv: p->len == 112 p->tot_len == 139
In Bluetooth. Receving and Sending Data
Length of data 0: 113
BEGIN:VCALENDAR
VERSION:1
BEGIN:VEVENT
DTSTART:2010712T136Z
RRULE:FREQ=0;BYMINUTE=3;COUNT=1
DESCRIPTION=0
Length of data 1: 31
Length of tempdata 1: 113
END:VEVENT
END:VCALENDAR

The output from procesIncomin: BEGIN:VCALENDAR
VERSION:1
BEGIN:VEVENT
DTSTART:2010712T136Z
RRULE:FREQ=0;BYMINUTE=3;COUNT=1
DESCRIPTION=0
ÆEND:VEVENT
END:VCALENDAR
Recevied New Activation Schedule
VCALENDAR Begins
End of Current Event
RRULU OUTPUTS: RRULE
RRULU OUTPUTS: FREQ
RRULU OUTPUTS: BYMINUTE
RRULU OUTPUTS: COUNT
Date : y: 110, m: 7,day:1, h: 0, min: 36
Description: 0, Freq: 0, period: 3, duration: 1
using 60 s sample interval with 6 samples per file
using 180 s wakeup interval with 60 s awake
Read Pointer: 1 Write Pointer: 3
rfcomm_ua
rfcomm_disconnected: CN = 1
```

Figure 7.3: Activation Schedule Dynamic Update - Mulle Evaluation Screen Shots

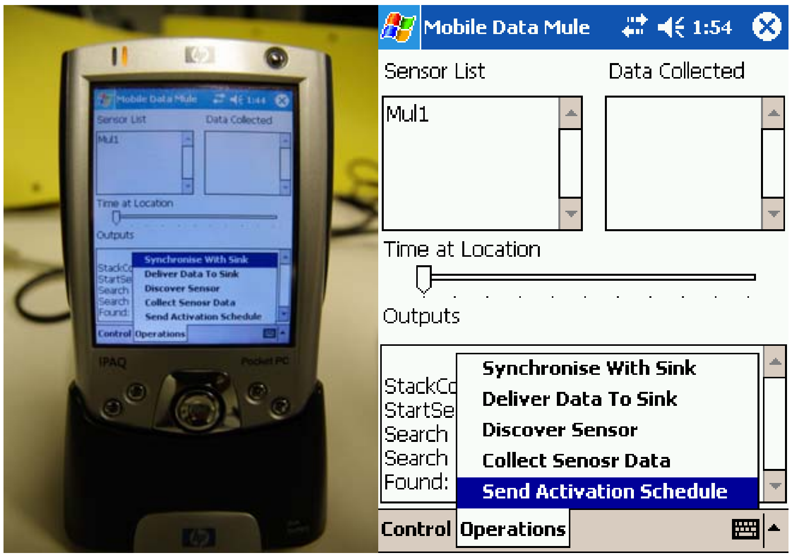


Figure 7.4: Activation Schedule Dynamic Update - PDA Screen Shot

7.2.2 Evaluating Shortened Bluetooth Discovery

The shortened Bluetooth discovery process is used for quick and efficient device discovery as against the complete 10.54 seconds discovery process given by Bluetooth specification (BluetoothSIG, 2010d). This approach was presented in detail in Chapter 3 and the implementation details using restricted search timers were presented in Chapter 6. In this section, we evaluate the success rate of the discovery process using shortened interlaced Bluetooth discovery. To determine the best possible Bluetooth discovery time rather than the standard 10.54 seconds, we performed Bluetooth discovery operation with 4 devices for varying time intervals. In each case, we compute the *discovery success ratio*. The *discovery success ratio* is defined as the ratio between devices that are in discoverable mode to devices that are successfully discovered. We varied our search interval between 2 and 4 seconds (at least 60% less than the standard interval). The respective results are presented in Figure 7.5. The evaluation outcomes presented were averaged over 10 independent experimental runs with devices at varying distances. Our evaluation was based on v1.2 of Bluetooth stack specification (BluetoothSIG, 2010d) available on the PDA. We conclude the following from the results. 1) The use of 3-4 second discovery interval produces 90% discovery success. 2) The 3-4 second discovery interval is at least 50% less than the Bluetooth specification recommendation and 3) No modification to the existing Bluetooth stack is required to achieve 90% discovery success. The experimental outcome validates the advantages of using the shortened Bluetooth discovery in the sGaRuDa framework. By reducing sensor discovery time, the overall data collection time also reduces.

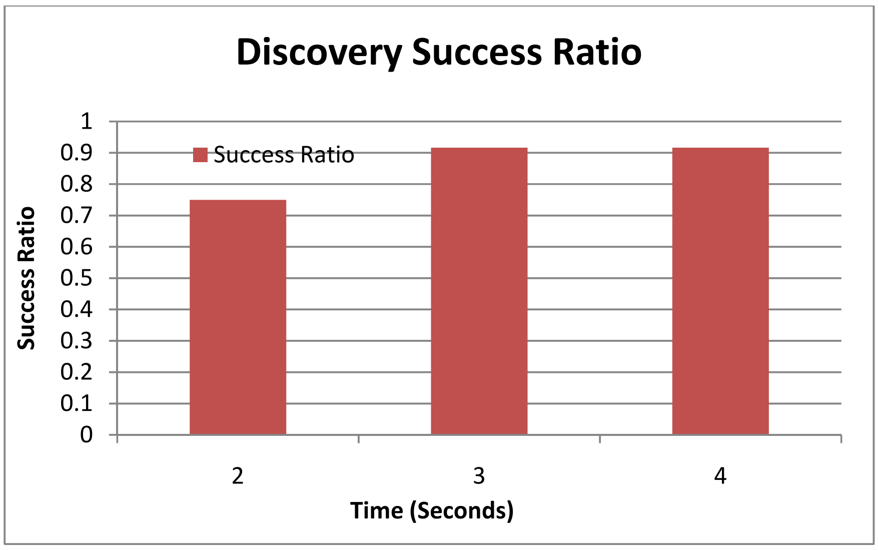
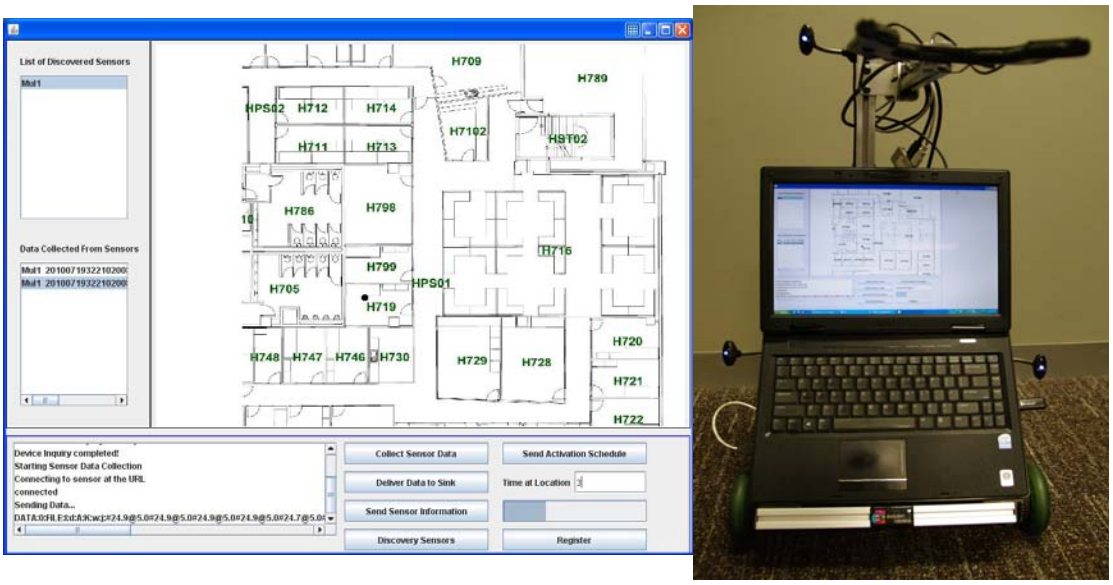


Figure 7.5: Discovery Success Ratio

7.2.3 Window-based Data Collection: Implementation Evaluation

In this section, we evaluate the implementation of the proposed window-based data collection technique. Our aim is to validate the feasibility of the proposed data collection protocol in real-world scenarios. The sensor node receives the window size from the mobile data mule. For experimentation purposes, we have used a window size of 1. Hence, the Mulle sensor node waits for an acknowledgement after every successful transmission. The Mulle waits for a data transfer request from the mobile data mule. On receiving a new request with window size *w*, the Mulle sends *w* packets before waiting for an acknowledgement. The underlying link layer is used to identify any disconnection. When a new data transfer request is received from another mobile data mule, the sensor starts resending the last unacknowledged packet. The screen dump of the mobile data mule (Robot platform) after a single round of data collection is presented in Figure 7.6. The Mulle sensor node screen dump is presented in Figure B1 in Appendix B. In this experiment, we used a mobile Robot as the mobile data mule. The mobile robot ER1 is depicted on the right side of Figure 7.6. The screenshot on the left of the Figure 7.6 is a magnified view of the sGaRuDa software framework running on the robot.



```
Mul1 201007193221020
Mul1 201007193221020
```

```
Device Inquiry completed!
Starting Sensor Data Collection
Connecting to sensor at the URL
connected
Sending Data...
DATA:0:FILE:t:d:A:K:wj:#24.9@5.0#24.9@5.0#24.9@5.0#24.9@5.0#24.7@5.0
```

Figure 7.6: Window-based Data Collection: Mobile Robot Screen Shot

The above experimentation presented a data collection run involving a mobile data mule. The key feature of the proposed data collection algorithm is to facilitate multi-part data collection in the presence of multiple mobile data mules. A black box of the experimental setup is presented in Figure 7.7. Figure 7.8 illustrates the working of the data collection algorithm in the presence of multiple mobile data mules. A flow diagram is used to represent the arrival of mobile data mules that communicate with the sensor.

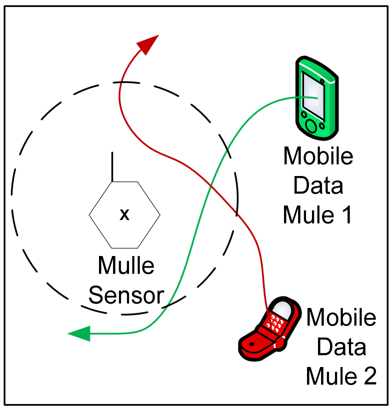


Figure 7.7: Disconnected Data Collection - System Setup

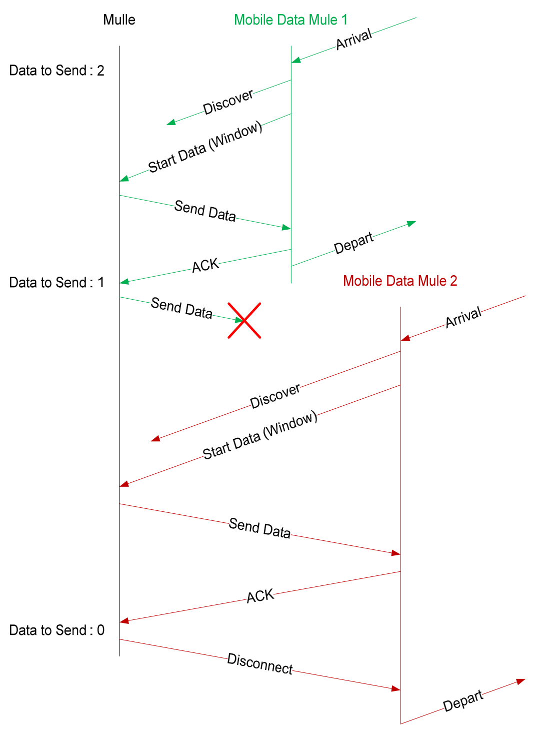
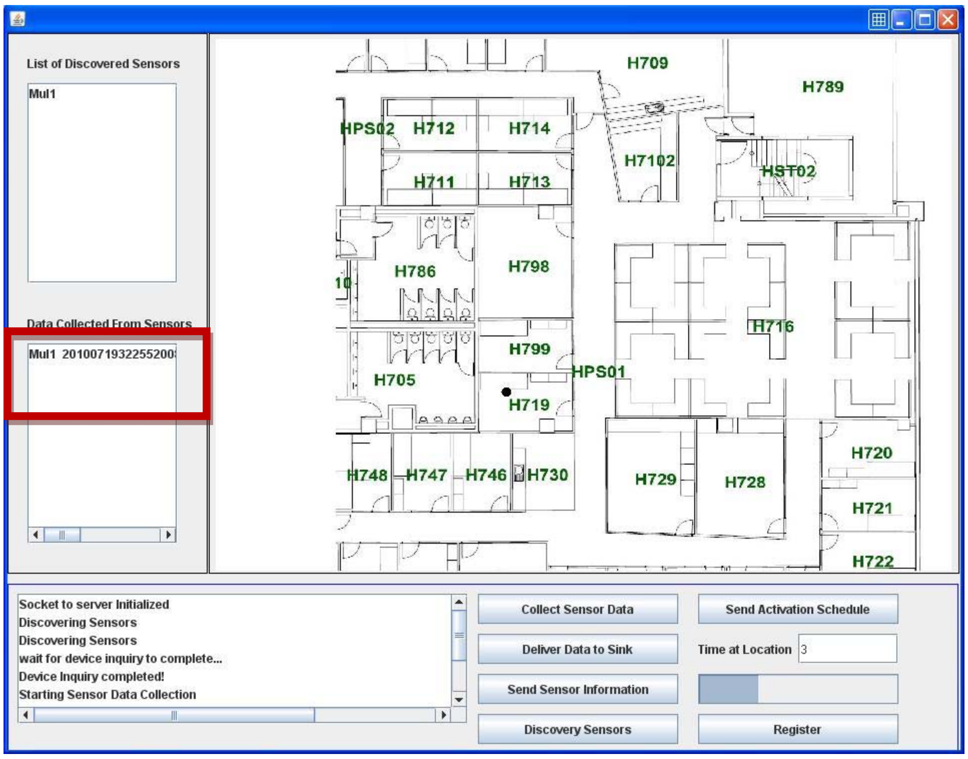


Figure 7.8: Flow diagram for disconnected Data Collection

The experimentation parameters are presented in Table 7.2. We use smaller values for our parameters for ease of screenshot presentation. For this experimentation, we used a PDA and a mobile robot based data mules. The implementation evaluations are presented as screen dumps of the PDA and the mobile robot respectively. These screen dumps are presented in Figure 7.9 and Figure 7.10. In this section we focus on the implementation evaluation of the window-based data collection protocol in the presence of multiple mobile data mules. A quantitative evaluation of the data collection algorithm is presented later.

Parameters	Value
Mobile Data Mules	2
Number of Packets	2
Sensors	1
Window Size	1

Table 7.2: Window-based Data Collection - Evaluation Parameters



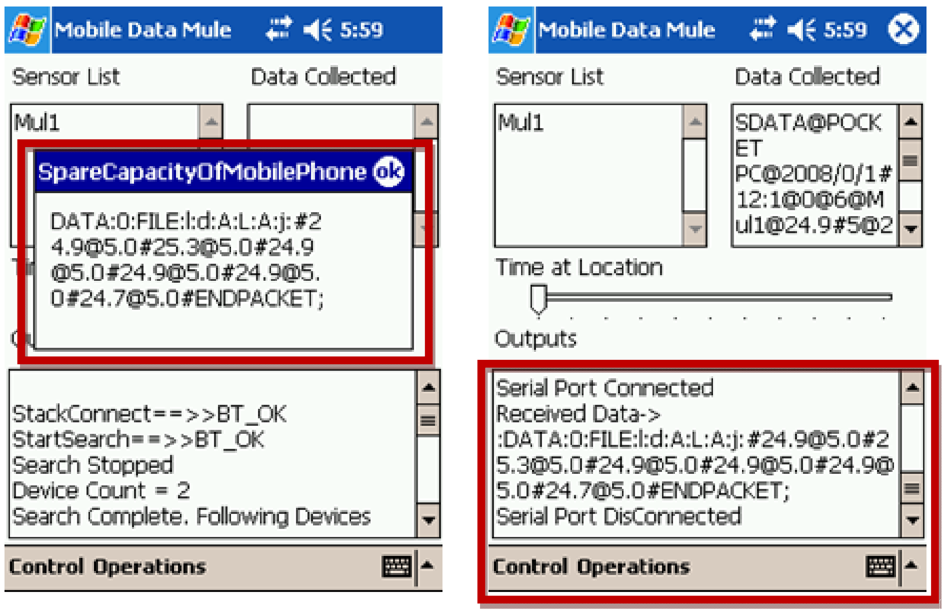
Mul1

Mul1 2010071932255200

```
Socket to server Initialized
Discovering Sensors
Discovering Sensors
wait for device inquiry to complete...
Device Inquiry completed!
Starting Sensor Data Collection
```

Time at Location: 3

Figure 7.9: Window-based disconnected data collection - Mobile Robot



Mul1

SpareCapacityOfMobilePhone

```
DATA:0:FILE:l:d:A:L:A:j:#2
4.9@5.0#25.3@5.0#24.9
@5.0#24.9@5.0#24.9@5.
0#24.7@5.0#ENDPACKET;
```

```
StackConnect==>>BT_OK
StartSearch==>>BT_OK
Search Stopped
Device Count = 2
Search Complete. Following Devices
```

Mul1

```
SDATA@POCK
ET
PC@2008/0/1#
12:1@0@6@M
ul1@24.9#5@2
```

```
Serial Port Connected
Received Data->
:DATA:0:FILE:l:d:A:L:A:j:#24.9@5.0#2
5.3@5.0#24.9@5.0#24.9@5.0#24.9@
5.0#24.7@5.0#ENDPACKET;
Serial Port DisConnected
```

Figure 7.10: Window-based disconnected data collection - PDA

The data available on the Mulle are collected by the PDA and the mobile robot during independent data collection runs. The screenshots in Figure 7.9 and Figure 7.10 show the collected sensor data. As described previously, the use of acknowledgements creates a platform for disconnected data transfer in the presence of multiple mobile data mules.

7.2.4 Mulle Sensor Node - Energy Consumption Experiments

In this section, we present Mulle control experiment results. The control experiments focus on the energy consumption of the Mulle during the following operations: 1) idle/sleeping, 2) listening 3) sensing and 4) communication. These states have been discussed in detail in Chapter 3. The control experiments on the Mulle were conducted using *Mulle v3* sensor node. The experimental setup comprises an ADC PCI card connected to the Mulle's expansion board. The ADC PCI card is used to measure the energy consumed by the Mulle sensor node. A desktop equipped with LabView (NationalInstruments, 2008) software was used to record the energy consumed.

Experiment 1: Boot-up/Low-power Mode- In this case, we compute the energy consumed by the Mulle to perform a boot-up operation. After a successful boot, the Mulle enters a low-power mode i.e. a state with the properties: 1) Bluetooth radio turned off and 2) MCU turned off. Result of our experiment is presented in Figure 7.11. In low-power mode the Mulle consumes only 1.4mW.

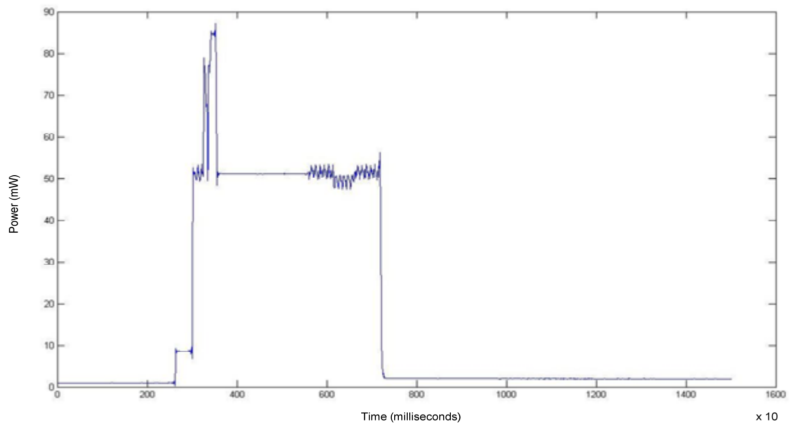


Figure 7.11: Experiment1: Bluetooth Mulle Boot-up/idle

Experiment 2: Sense- During sense, the Bluetooth radio is turned off and the MCU is turned on. The energy consumed during a sense operation varies with the type of sensor used on the Mulle. For example, an accelerometer might consume more energy than the on-board temperature sensor. The experimental result presented in Figure 7.12 used the on-board temperature sensor. The following operation is performed by the Mulle to sense a sample of data. They are: 1) change from idle state to sense state 2) read current temperature and 3) use the on-board analog-to-digital convertor (ADC) to store the measured temperature to the on-board flash. For experimentation, the Mulle was programmed to sense every 2 minutes. Each sense operation consumed 19.82mW of energy. The section marked as *a* in Figure 7.12 indicates the Mulle's boot operation. The section marked *b* indicates the sensing operation using the on-board temperature sensor. The Mulle enters low-power mode during the time period between sense intervals.

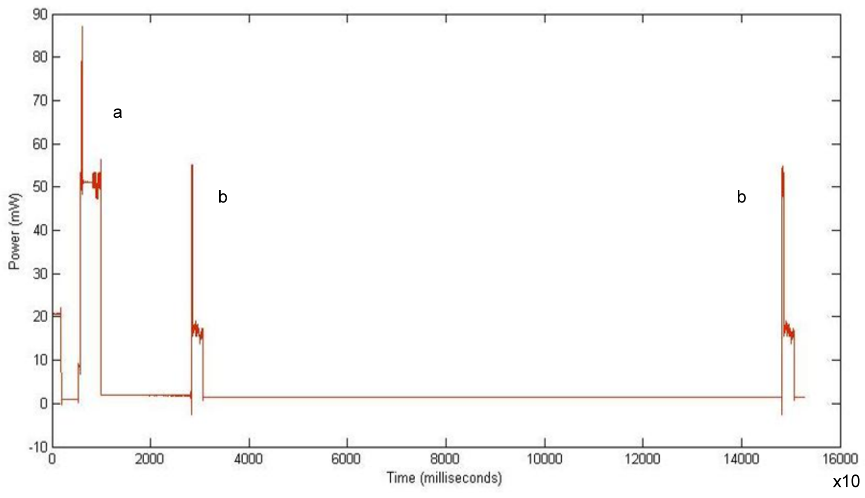


Figure 7.12: Experiment 2: Mulle Sense Operation

Experiment 3: Bluetooth Listen- During Bluetooth listen, the Bluetooth radio is powered on and is in listen mode. For the following experiment, the Bluetooth radio and the MCU were switched on. The Mulle performs frequent inquiry scan enabling it to respond to any new incoming request. The Mulle only accepts incoming connection and does not self-initiate a connection. The experimental result is presented in Figure 7.13. The peaks in the power consumption shown in Figure 7.13 are due to the back-off between subsequent inquiry scan. The energy consumed by the Mulle in the Bluetooth listen state was measured to be 27.4mW.

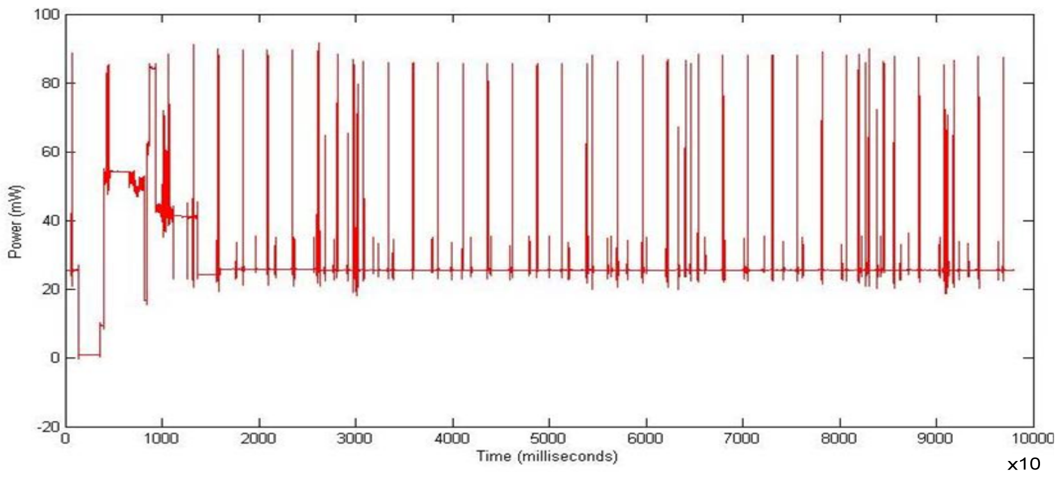


Figure 7.13: Experiment 3: Bluetooth Listen

Experiment 4: Bluetooth Connection: In this state, a Bluetooth connection is established between the Mulle and the mobile data mule. The MCU, Bluetooth radio and the real time clock are all powered up. The experimental outcome is presented in Figure 7.14. The energy consumed by the Mulle during the connection operation was measured to be 160.61mW per millisecond. We have stated through the thesis that communication is the major energy consuming operation of a sensor node. The outcome of our control experiment validates this claim.

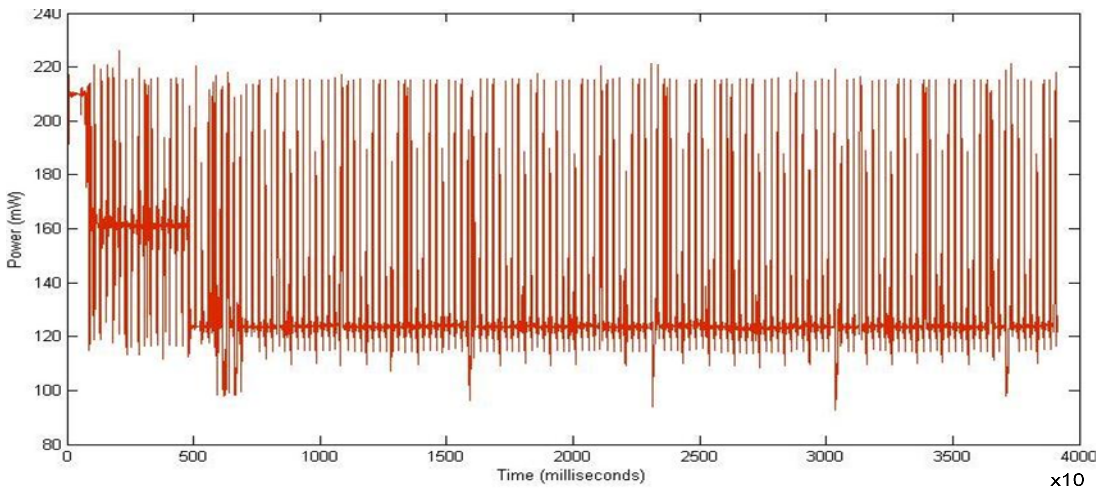


Figure 7.14: Experiment 4: Bluetooth Connection

Further, to evaluate the use of threshold-based data collection by the sGaRuDa framework, we conducted experiments to compare the energy consumed by the Mulle when communicating with a mobile data mule at different locations. The experiment was conducted for 30 seconds of continuous data exchange between the Mulle and the mobile data mule. The mobile data mule used for this experimentation was the robot which was placed in two different locations. The first location was an obstacle free environment while the other was an obstacle filled environment. The mobile data mule was moving at a constant speed of 30cm/sec. The result of our experimentation is presented in Figure 7.15. The result presented in Figure 7.15 shows higher energy consumption in an obstacle filled environment. The energy consumption presented is over every second. From this result, we prove the influence of obstacles and distance on the power consumed by the Mulle. The experimental outcome clearly validates our argument that greater distance and poor channel quality (obstacles) result in higher power consumption. The result supports our proposed threshold-based data collection approach that considers real-world channel quality metrics in the data collection process.

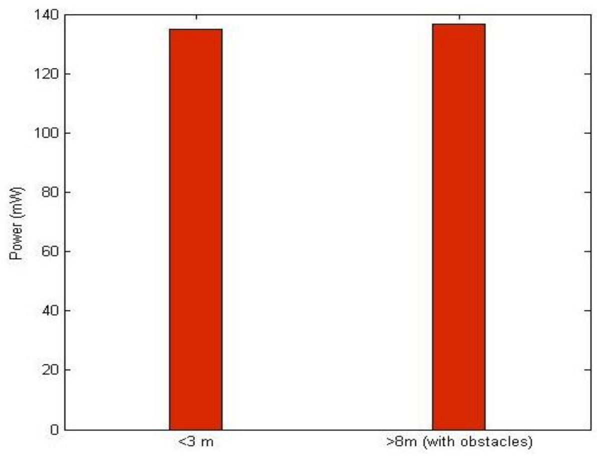


Figure 7.15: Experiment 5: Comparing Energy Consumption at varying Distances

7.2.5 Window-Based Data Collection Algorithm: Quantitative Evaluation

In this section we evaluate the proposed window-based data collection algorithm quantitatively. The discussion presented in section 7.2.3 evaluated the feasibility of the proposed window-based data collection in real-world scenarios. To evaluate the data collection algorithm quantitatively, we perform the following experiment: 1) data collection in the absence of a window-based technique; 2) data collection using the proposed window-based technique and 3) data collection using multiple mobile data mules. The experiments were conducted under varying channel qualities and mobile data mule movement speeds. The mobile data mule used for experimentation is the mobile robot whose speed was varied to simulate real-world movements. To achieve varying channel quality, the mobile robot (data mule) was moved along paths within the building separated by walls, desks and glass windows. Each experiment was repeated 5 consecutive times and the results were averaged. The parameters used for the experimentation are presented in Table 7.3. The experimental setup within the building is depicted in Figure 7.16. The mobile robot moves along the path represented by a red arrow in the Figure 7.16. At different locations along the mobile data mule's trajectory Mulle 1 and Mulle 2 enter and exit the communication range.

Parameters	Values
Total number of packets	100
Mobile Data Mule Speed	30 and 50 cm/second
Channel Quality	Clear Straight path/Obstacle (walls, doors, chairs, etc) filled path
Number of Sensors	3
Number of Mobile Data Mules	2

Table 7.3: Window-based Data Collection - Experiment Parameters

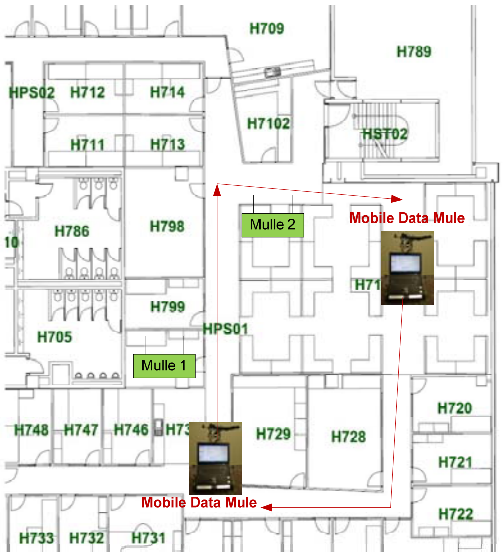


Figure 7.16: Windows -based Data Collection - Experimental Setup

Experiment 1: In this experiment, we compute the total number of packets received by the mobile data mule from Mulle after connection establishment using non-window based communication (case 1) and window-based communication (case 2). The time the mobile data mule was within the coverage of the sensor nodes (Mulle) was averaged to be 19 seconds for both cases (window-based and non-window). For this experimental run, we only considered a single data collection run. A data collection run is a single traversal of the sensor network by the mobile data mule. The time spent within the sensor node's coverage does not include the time involved in discovery. The mobile data mule, with the help of Ekahau position engine (EPE) and available trajectory information, estimates the residual time, i.e. time within the radio coverage range of the sensor node. We assume sensors are location-aware and the location information is obtained during the discovery phase.

For the experimental run, we used a clear data communication channel (clear line of sight) between the Mulle and the mobile data mule at 30cm/sec movement speed. At higher speeds, the communication path between the mobile data mule and the Mulle were filled with obstacles. The experimentation outcome is presented in Figure 7.17. The results show that even at higher speeds, at least 50% of the data is collected. Further, at 30cm/sec, the mobile data mule receives more data packets. The results validate the effect of mobility, speed and poor communication channel quality on the efficiency of data collection process. The window-based approach collects fewer packets than the non-window based approach due the use of acknowledgements. In this example, the window size used was 10. In both cases, using a single data collection run, the estimated time was not sufficient to collect all the data. To validate the use of multi-part data collection, we repeated the above experiment with an additional data collection run. The outcome of the experimentation is presented in Figure 7.18.

The experimental outcome validates the advantage of using the proposed multi-part data collection approach. As indicated by the experimental outcome, the use of subsequent data collection runs, without changing any other data collection parameters, increased the packet collection rate to 100%. With the non-window based approach, the amount of packets collected remains the same, as none of the data collection parameters were modified, i.e. the time the sensor and the mobile data mule are in range has not been modified. The major drawback of the non-window based approach is that Mulle is not aware of the number of successfully delivered packets. Hence, any disconnection results in the entire set of data packets being re-transmitted. This is not the case with the window-based data collection technique. The sensor re-transmits only the lost data packets.

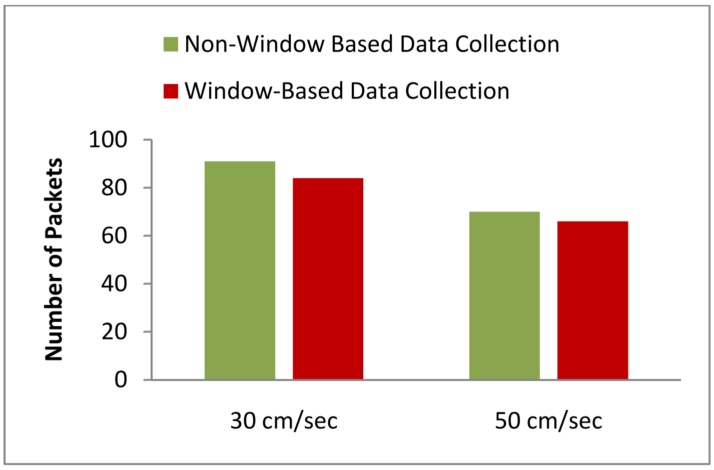


Figure 7.17: Experiment1: Non-Window based and Window-based Data Collection - 1 Data Collection Run

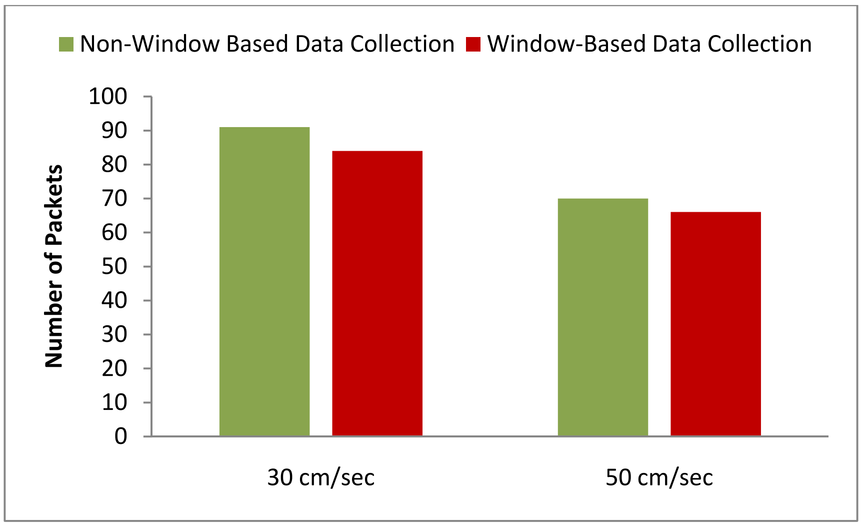


Figure 7.18: Experiment1: Non-Window based and Window-based Data Collection - 2 Data Collection Runs

Experiment 2: Experiment 2 aims to evaluate the energy efficiency of the proposed window-based data collection technique by evaluating a scenario involving two mobile data mules. The experimental setup is similar to the one presented in Figure 7.16.

In this experiment, we consider a sequential mobile data mule arrival. The first mobile data mule stays within the communication range for 10 seconds while the second one stay until it collects all the available data packets. In case 1, the experiment was conducted without window-based acknowledgements while in case 2, the experiment was conducted with a window size of 10. The outcome of the experimentation is presented in Figure 7.19.

The experiment was performed under two situations, a clear line of sight environment and an obstacle filled environment. In each case, the mobile data mule's speed was randomly changed to determine the effect of mobility on the window-based data collection approach. The result presented in Figure 7.19 is the total time taken to collect the entire sensor data. The results show the savings in time using the window-based technique. The results validate the observations made previously in *Experiment 1*. Further, the non-window based approach requires at least *30sec* to complete the data collection while the adaptive window-based approach will run to completion during subsequent data collection cycles/runs. To compute the total energy spent during the communication we use the experimental result presented in Figure 7.14. We conclude that, the use of the proposed window-based data collection technique results in at least 25% energy saving (based on Bluetooth energy consumption presented earlier).

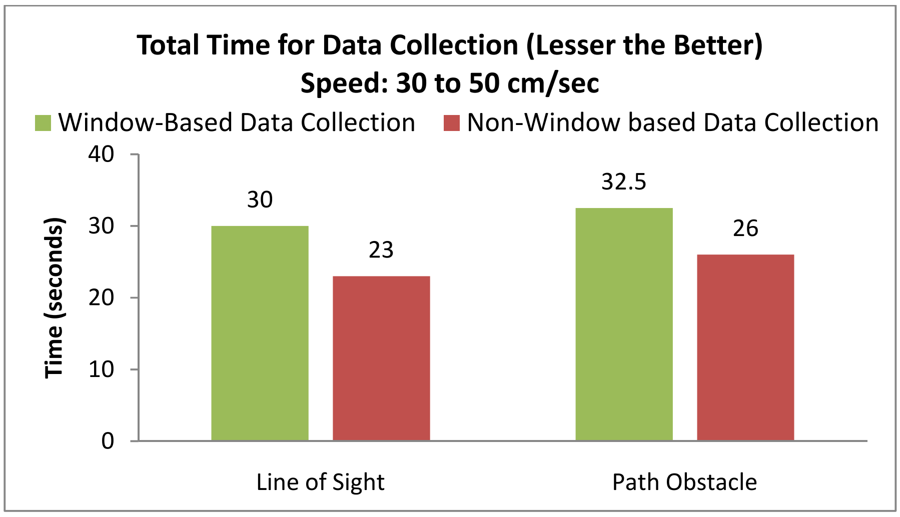


Figure 7.19: Experiment2: Non-Window based and Window-based Data Collection

7.2.6 Dynamic Activation Schedule: Quantitative Evaluation

In this section, we evaluate the proposed dynamic activation schedule algorithm by determining the sensor discovery ratio. The discovery ratio is defined in (1). The sensor discovery ratio helps to compute an energy consumption graph using the experimental outcomes. To verify and validate the gain in energy using the proposed approach, we compare the evaluation outcome with a non-adaptive approach i.e. classic sensor operation without dynamic duty cycle adaptation based on mobile data mule arrival.

$$\boldsymbol{Discovery\ Ratio} = \frac{\boldsymbol{Total\ Number\ of\ Successful\ Connection}}{\boldsymbol{Total\ Number\ of\ Listen\ Intervals}} \qquad (7\text{-}1)$$

The ability to change the activation schedule of the Mulle using the mobile data mule allows the sensor to adapt to changing mobile data mule arrival rate. This is not feasible in the classic sleep/wake (listen) approach. Moreover, computing the mobile data mule's arrival time on the sensor is an expensive process and requires enough information about data mule arrivals. We use the terms wake and listen inter-changeably to describe the process of the sensor waking up from the sleep mode and entering the listen mode. For our experimentation, we have used 3 Mulles and one mobile data mule. For the mobile data mule arrival we assumed a Poisson arrival process. The experimental setup is presented in Figure 7.20. The trajectory of the mobile data mule is a straight path and sensor nodes are placed in a row. This setup in no way restricts our approach for other sensor deployments. We merely use this setup for ease of experimentation. We consider two cases, Case 1) No sensor duty cycle adaptation and Case 2) Dynamic sensor duty cycle (activation schedule) adaptation using the mobile data mule.

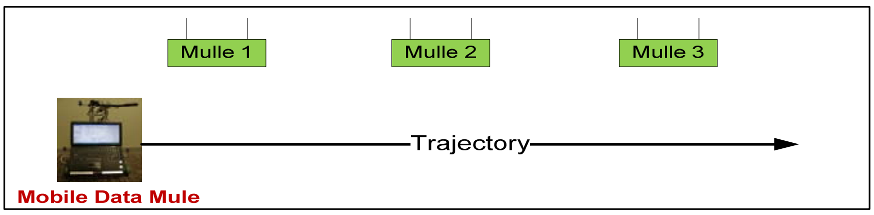


Figure 7.20: Experimental Setup - Dynamic Activation Schedule

We evaluate the proposed dynamic activation schedule-based discovery by computing the discovery ratio for the setup presented in Figure 7.20. To compute the Poisson arrival, we used MATLAB to generate arrival times over unit of time. To provide an illustrative understanding of the dynamic activation schedule based sensor discovery, the Mulle sensor's wake interval, adapted (activation schedule) and non-adapted are computed and the results are presented as a time series in Figure 7.21 using MATLAB. The blue dots represent the mobile data mule's arrival at time *T*. The red dots show the adapted sleep/wake schedule of the Mulle. The green dots represent the non-adapted sleep/wake interval of the Mulle. As illustrated in the figure, the non-adapted sensors follow a periodic sleep/wake interval. On the contrary, the adapted sensor nodes have a changing sleep/wake interval based on the mobile data mule arrival. Moreover, as we can see in the figure, the dynamic activation schedule of the Mulle represented in red has less wait time before discovering a mobile data mule compared to the non-adapted approach. The experiment consisted of 4 data collection rounds and was repeated 5 times.

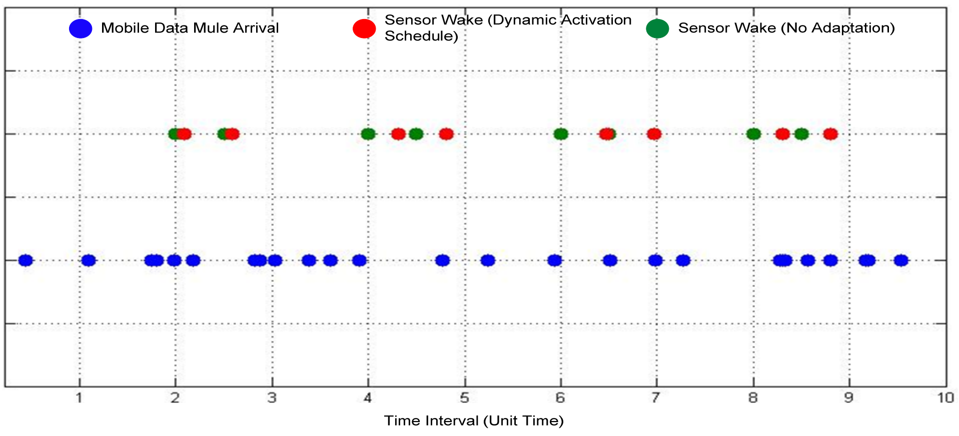


Figure 7.21: Experiment 1: Dynamic Activation Schedule

Based on the result presented in Figure 7.21, we present the result of the discovery ratio computation in Figure 7.22. As we can see, the adapted approach (dynamic activation schedule) produces 100% success rate in most cases compared to the non-adapted approach. The overall discovery percentage using adapted approach is 94% while the overall discovery percentage using the non-adapted approach is 70%. Discovering a sensor node allows the mobile data mule to collect the sensor data immediately, hence reducing the sensor node's listen duty cycle. The result presented in

Figure 7.21 is used to compute the overall energy consumed during the data collection process. The total energy spent by the Mulle is given by (7.2)

$$\boldsymbol{Total\ Energy\ Consumed} = \boldsymbol{d_{idle}} + \boldsymbol{d_{sense}} + \boldsymbol{d_{listen}} + \boldsymbol{f}(\boldsymbol{d_{connection}}, \boldsymbol{distance}) * \mathbf{N} \qquad (7\text{-}2)$$

Where *d* denotes the duty cycle for the corresponding state i.e. *idle, sense, listen* and *f (d*$_{connection}$*, distance)* denotes the energy spent during connection which is impacted by distance. *N* is the total number of connections established between the Mulle and the mobile data mule.

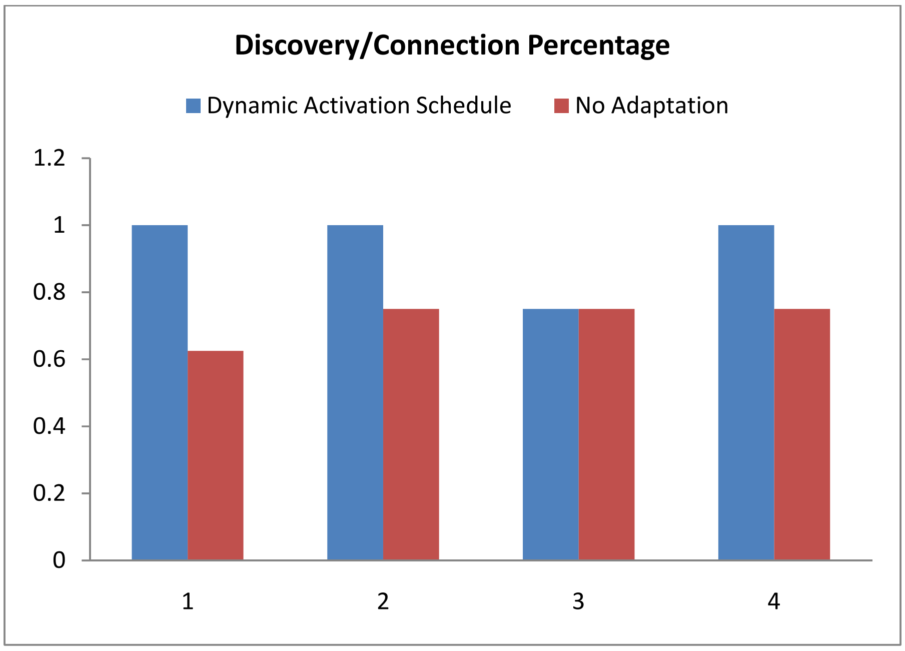


Figure 7.22: Experiment 2: Discovery Ratio

To evaluate the energy consumed by the Mulle using the dynamic activation schedule, we only consider the energy spent during the waiting state. We ignore the energy spent during connection establishment and data exchange. The outcome of our experiment is presented in Figure 7.23. The result in Figure 7.23 presents the total energy spent by the Mulle in waiting for the data mule arrival. The result presents the cumulative energy consumed at the end of each data collection round.

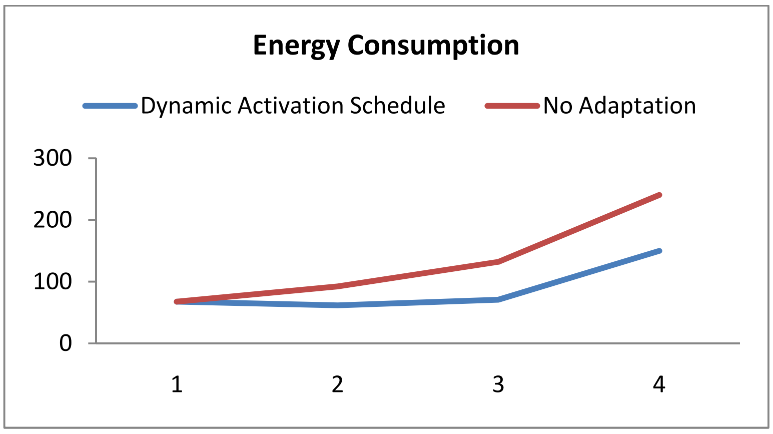


Figure 7.23: Experiment 3: Dynamic Activation Schedule

As the result illustrates, the energy consumed by the non-adaptive approach increases significantly. By contrast, the proposed activation schedule approach adapts the energy consumption based on the data mule arrival. This is evident from the linear increase in energy consumption. The experimental outcome verifies and validates the proposed dynamic activation schedule algorithm's energy efficiency over the classic non-adaptive approach. We conclude that using the proposed dynamic activation scheduling, the lifetime of the sensor node can be extended considerably hence improving the overall sensor network lifetime. In this experimentation, we did not assume any sensor-sensor communication. By introducing sensor communication, load balancing can be achieved whereby only one Mulle is awake during a data collection round. This approach can further reduce the energy consumed by individual sensors, therefore increasing the overall sensor network lifetime.

7.3 3D-KNN: *kNN*-based Data Collection Using Mobile Data Mules - Evaluations

In this section, we evaluate the cost-efficiency of the 3D-KNN algorithm. We defined cost-efficiency within the scope of the 3D-KNN algorithm as a function of the following: 1) energy involved in communication, 2) query processing latency, (performance) and 3) impact on overall network lifetime (total energy consumed). The energy spent on communication involves the energy consumed during each phase of the

3D-KNN algorithm. The energy spent during the final data collection phase varies depending on the amount of data that needs to be collected from the sensor network.

We use the following key evaluation metrics to validate the cost-efficiency of the proposed 3D-KNN algorithm. The evaluation metrics used are:

Boundary Estimation: Size of the *kNN* boundary estimated has a direct impact on the overall energy consumption i.e. bigger the boundary, more communication is involved and lesser the boundary, enough sensor nodes may not be covered.

Query Latency: Time taken to issue a *kNN* query and obtain a result for varying *kNN* boundary sizes determined by *k.*

Energy Consumption: Overall energy consumed by the entire network to process a *kNN* query request.

Energy Consumption per Node: Energy consumed by each sensor over a single round of data collection in the presence of a mobile data mule.

Further, we validate the proposed 3D-KNN prediction algorithm by comparing the results of the predicted 3D-KNN based data collection algorithm against the non-predictive 3D-KNN algorithm. Our experimental evaluations were performed in GloMoSim (2010) details of which were presented in Chapter 6. The parameters presented in Table 7.4 were used for our experiments. A value of 2.5 was used for *c* to compute the *KNN-METRIC*

Parameter	Value
Number of Nodes (N)	20 to 200
Area Size (A_T)	1000 x 1000 x 1000
Node Distribution	Random
Radio T_x Power	10 dBm to 15 dBm
c	2.5
Mobile Data Mule	1
Mobile Data Mule Mobility	Trajectory(Using mobility file)

Table 7.4: GloMoSim Simulation Parameters

The sensor network environment simulated was the building scenario presented in Figure 7.1. Each sensor node is assumed to be location-aware. The sensors during initial run do not have any knowledge about the network or their neighbouring sensors. Neighbour discovery is done at runtime when sensors process the *kNN* query. This rule makes the proposed 3D-KNN algorithm more flexible to network infrastructure changes. The wireless sensor nodes are assumed to be static and listening to broadcast messages using a periodic sleep/wake schedule. Using a periodic wake/sleep schedule might result in some nodes being unavailable during data collection. This problem can be overcome using the dynamic activation schedule algorithm proposed and evaluated in previous sections. The mobile data mule has the ability to move within the sensor network along a pre-defined path. This path cannot be changed as we assume that mobile data mule cannot be controlled by the simulation environment. For our evaluations, we only consider *one-shot kNN* queries. *One-Shot kNN* queries are issued only once and the results are computed based on the query response. Subsequent *kNN* query is not issued from the same location. To validate the proposed 3D-KNN algorithm, we have performed the following experiments:

1) Evaluate the boundary estimation algorithm
2) Evaluate the *kNN* query processing efficiency
3) Evaluate the energy consumed during *kNN* query processing

7.3.1 Boundary Estimation Algorithm Evaluation

The *kNN* query is propagated by the mobile data mule into the sensor network. The boundary area is estimated by the mobile data mule based on the sample size *k*. In this section, we present experimentation by considering a range of values for the sample size *k*. Our aim is to validate the computing efficiency of the boundary estimation algorithm for different values of *k*. The result of our experiment is presented in Figure 7.24. The result of the boundary estimation algorithm is a set *S* which comprises at least *k* nearest neighbours.

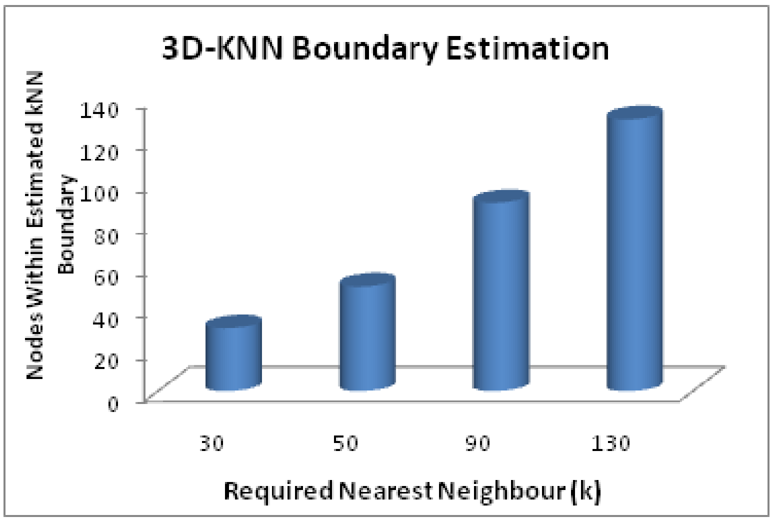


Figure 7.24: 3D-KNN Experiment 1: Boundary Estimation

The key evaluation criterion for the boundary estimation algorithm is the size of set *S*. The boundary area comprising *s* sensors needs to cover at least *k* sensor nodes such that $k \leq s$. The size of the set *S* impacts the overall energy consumption. As illustrated by the evaluation outcome, for a nearest neighbour size *k*, the estimated boundary set *S* covers at least *k* sensor nodes. The size of the set *S* is sufficiently large encompassing the required number of sensors. This is verified by the experimental outcome in case 4. In case 4, the experiment was conducted with a sensor network capacity of 200 nodes and for a requested *k* of size 130, the computed *kNN* boundary consisted of 140 sensor nodes. The experimental outcome validates the efficiency of the boundary estimation algorithm. The efficiency is determined by the size of set *S* i.e. *S* is neither too small nor too large.

7.3.2 Query Processing Latency

The query latency is the time taken to process a *kNN* query for varying sizes of *k*. To verify the query performance of the proposed 3D-KNN algorithm, we have performed experimentation for various sensor network sizes under changing simulator parameters. To validate the performance of the 3D-KNN algorithm, we have compared the experimental results of the 3D-KNN algorithm with a static sensor network-based *kNN* query processing algorithm, namely, KBT (Winter et al., 2005). The KBT approach is most relevant to the proposed 3D-KNN as it uses a infrastructure-free approach. Other techniques employing *kNN* queries in sensor networks use indices or require special sensor hardware (Wu et al., 2007) to execute the *kNN* queries. The KBT approach employs a fixed TreeHeight i.e. the maximum hop distance the query propagates. This value is assumed to be static and hence influences the *kNN* boundary. The problem with this approach is that *kNN* boundary cannot adapt dynamically for varying values of *k*. We implemented the KBT algorithm in GloMoSim using a fixed hop-count technique i.e. sensors stop broadcasting after a particular number of hops have been reached. We varied the value of the TreeHeight manually during every simulation run based on the nearest neighbour size (*k*). This operation is done dynamically during 3D-KNN execution. The result of the experimental outcomes is presented in Figure 7.25. The experiment was conducted by varying the number of sensors from 100 to 200. The value of *k* was varied between 30 and 100. The time required to process the *kNN* query involved the operations query preparation, query propagation and query response. With KBT, the centralised sink was fixed at a particular location while in the case of 3D-KNN the mobile data mule's location was changed based on the pre-defined trajectory. For increasing size of the sensor network, the radio range was modified to suit large-scale deployment areas. We present a short analysis of the results in the following paragraph. To help explaining the experimentation outcome, we have plotted a trend line over the results.

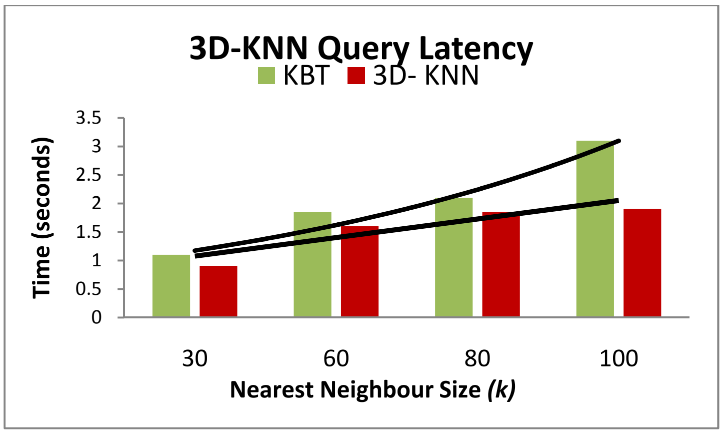


Figure 7.25: 3D-KNN Experiment 2: Query Latency

The trend line projected over KBT indicates a slightly non-linear increase in time as the number of nearest neighbours increase. On the contrary, a linear increase in time is noticed for the 3D-KNN algorithm as the number of nearest neighbours increase. The primary reason for the increase in time using the KBT algorithm is attributed to the use of fixed TreeHeight while the linear increase in time using the 3D-KNN algorithm is attributed to the use of dynamic *kNN* boundary computation. The result clearly proves the performance advantage of the 3D-KNN algorithm over KBT under the given simulation parameters. Further, the increase in time impacts the amount of energy spent by the sensor network to process the *kNN* query. This influence is investigated in the next section.

7.3.3 Energy Consumption

In this section, we perform experimentation to evaluate the energy consumption of the 3D-KNN algorithm. The experimental results presented in this section include the energy consumed during query preparation, query propagation, query execution and nearest neighbour computation. To validate the energy-efficiency of the 3D-KNN algorithm we compare our experimental outcomes with KBT (Winter et al., 2005).

The result of our experimentation includes the energy spent by the mobile data mule to propagate and process the *kNN* query. The energy involved in moving the mobile data mule from one location to another is not taken into consideration. For each experiment, we changed the value of *k* (nearest neighbours) to verify the feasibility of the proposed 3D-KNN algorithm over large-scale sensor networks. Moreover, it also proves the energy-efficiency of our proposed approach in large-scale sensor networks. The result of the experimental outcomes is presented in Figure 7.26. We present a short analysis of the experimental results in the paragraph following Figure 7.26.

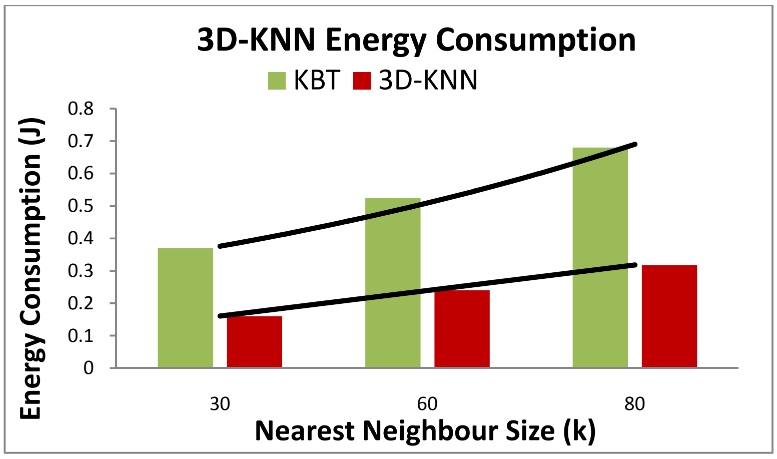


Figure 7.26: 3D-KNN Experiment 3: Energy Consumption

To help the analysis, we have projected trend lines on the experimental results. It can be noted that increase in energy consumption using KBT is non-linear, while the increase in energy consumption is linear using the 3D-KNN algorithm. The energy efficiency of the proposed 3D-KNN algorithm is attributed to the following: 1) the boundary estimation based on network density that covers at least *k* sensors, hence, reducing the total number of message broadcasts in the network; 2) plane rotation and nearest neighbour selection based on the *KNN-METRIC* choosing sensor nodes that are both closer and energy-efficient, (better communication channel). We use KBT's experimental outcomes as a benchmark for *kNN* query processing in sensor networks. The experimental outcomes clearly validate the energy efficiency of the 3D-KNN algorithm. Further, under given test environment and our simulation parameters, 3D-KNN proves to be more energy-efficient than KBT.

To further evaluate the energy consumption of the proposed 3D-KNN algorithm, we evaluate the neighbour selection algorithm that employs *KNN-METRIC* for nearest neighbour selection. To provide an understanding of the performance gain using the proposed neighbour selection algorithm, we compare 3D-KNN against a basic *kNN* (KNN) implementation. The basic *kNN* uses the same principles as KBT, employing a fixed TreeHeight (hop count). The only difference though is that basic *kNN* uses a modified *KNN-METRIC* to compute the list of nearest neighbours. The modified KNN-NETRIC takes only distance into consideration, excluding signal-to-noise ratio (SNR) parameter. For the evaluation, we compare the selected neighbour list based on the corresponding metrics. We use two cases for the simulations, Case 1) Basic *kNN* (KNN) and Case 2) 3D-KNN.

The simulation results are presented in Figure 7.27. The *KNN-METRIC* computed in case 1 is higher than the *KNN-METRIC* computed in case 2. This is quite obvious since only distance is taken into account in case 1. In both cases, higher value represents better performance i.e. higher *KNN-METRIC* indicates lesser distance in case 1. Similarly, higher *KNN-METRIC* in case 2 indicates lesser distance and better SNR. Further, we discuss two specific cases highlighted in the graph by black circles involving sensor node 5, 6, 8 and 9.

As indicated by the results the sensors have varying *KNN-METRIC* for corresponding cases. *KNN-METRIC* for node 5 is higher than node 6 for case 1. By contrast, the *KNN-METRIC* for node 5 is lower than node 6 in case 2. This indicates a different sensor node selection using the two metrics. The reason behind the difference is the introduction of SNR parameter in the *KNN-METRIC* which determines the node selection process. To elaborate further, assume sensor node 5 is at a lesser distance than sensor node 6. With the basic KNN approach, sensor node 5 would be selected due to lesser distance. Using 3D-KNN approach which introduces SNR parameter, the result can be interpreted as node 5 which is at a lesser distance than node 6 has poor communication channel quality. Hence, node 6 is a better choice from an energy perspective even though it is at a farther distance. We come to this conclusion from our previous experimentations where we have proven that distance and signal quality produces an impact on the overall energy consumed during communication. The same observation can be made for the sensor nodes 8 and 9. As stated in chapter 4, the *KNN-METRIC* can be further extended by introducing additional parameters that increase the energy-efficiency of the data collection process.

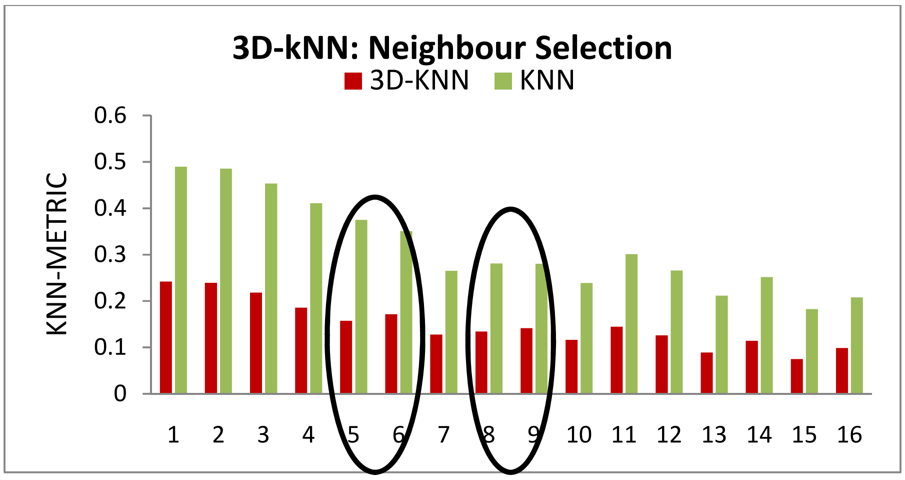


Figure 7.27: 3D-KNN Experiment 4: Neighbour Selection

7.3.4 Energy Consumption of Individual Sensor Nodes

In this section, we perform experimentation to compute the energy spent by individual sensors. The experimental result is the outcome of a data collection run. The experimental setup involves 80 sensors within a 1000 x 1000 x 1000 space. The mobile data mule travels within the sensor network in a known trajectory collecting sensor data. At each instance, it issues a *kNN* query, collects query responses, computes nearest neighbours and collects sensor data. Unlike approaches discussed in the literature (Shah et al., 2003) the sensors were not distributed in a grid fashion. A random sensor network deployment was used. The result of our simulation is presented in Figure 7.28.

The noticeable outcome of our experimentation outcome presented in Figure 7.28 is the uniform depletion of energy across the entire sensor network. The illustration was computed using MATLAB's 3D bar chart. Though the sensors in the graph are represented in grids, the experimentation was not performed by placing sensors in grids. Hence, the position of a sensor in the graph does not correspond to its actual location within the environment. The nodes with nil energy usage were the sensor nodes that did not fall within the path of the mobile data mule. These sensors were distributed outside the *kNN* boundary. The simulation results presented were averaged from 10 simulation runs. The energy consumption of individual sensors further validates the energy-efficiency of the proposed 3D-KNN algorithm.

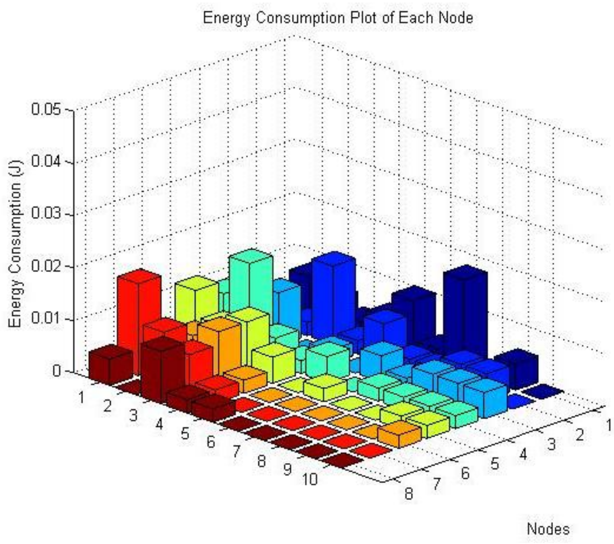


Figure 7.28: 3D-KNN Experiment 5: Individual Energy Consumption

7.3.5 Energy Consumption with Neighbour Prediction

In this section, we evaluate the proposed neighbour prediction algorithm used by the mobile data mule to improve the energy efficiency of the data collection process. To validate the proposed algorithm, we compare the outcome of the neighbour prediction experimentation with a non-predictive 3D-KNN approach. The non-predictive approach does not compute future nearest neighbours. Hence, a new *kNN* query with no reference to previously discovered sensors is broadcast into the network. Our experimentation setup is presented in Figure 7.29. The red triangle and the lines indicate the position and the trajectory of the mobile data mule.

The prediction algorithm is employed by the mobile data mule over the result-set obtained from the *kNN* query. The experimental setup consisted of 20 sensors within a 1000 x 1000 x 1000 space as shown in Figure 7.29. The mobile data mule propagates a *kNN* query at each location marked by the red triangle. Based on the query response, the mobile data mule computes future neighbours along its path. This information is then propagated with the next *kNN* query at the subsequent location. Sensor nodes receiving the new broadcast with pre-selected node list send the data directly to the mobile data mule. The outcome of the experimentation is presented in Figure 7.30. The energy consumption results are an average of 5 independent data collection runs using both predictive and non-predictive approaches.

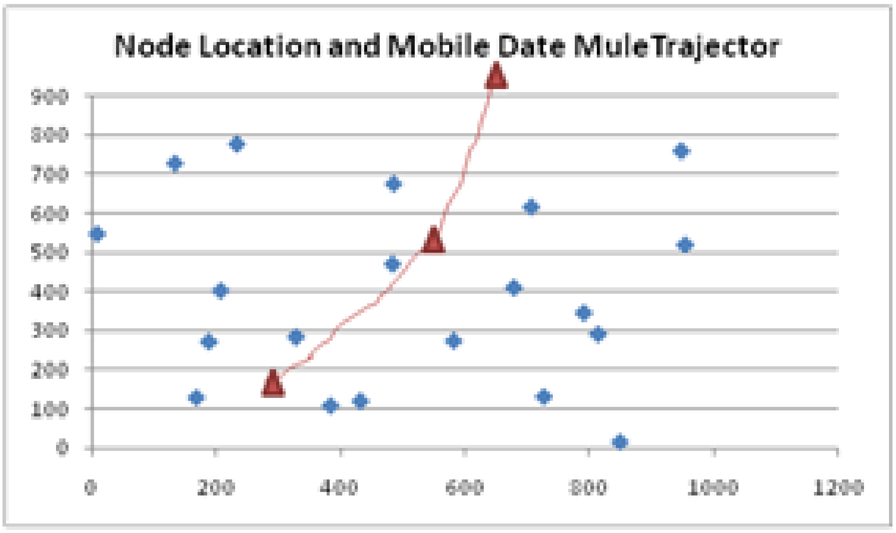


Figure 7.29: 3D-KNN Experiment 6a: Neighbour Prediction Simulation Setup

The experimental results show considerable energy savings using the proposed prediction algorithm. The prediction-based approach saves up to 35% more energy than the non-predictive approach. This validates the advantage of using prediction in the 3D-

KNN algorithm. Further, savings in energy depicted in this case for a sensor network of size *20* with a requested *k* of *15*. The sample size 15 almost covers 75% of the sensor network deployment area. Hence, extending our outcome to large-scale sensor networks would result in higher energy savings.

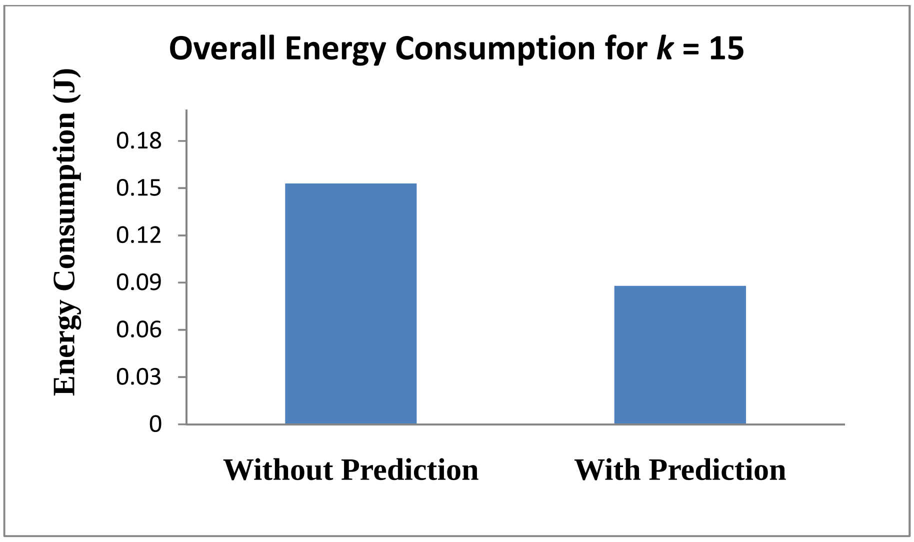


Figure 7.30: 3D-KNN Experiment 6b: Neighbour Prediction

7.4 R-CS (Context Spaces Extensions) - Evaluation

In this section, we evaluate the proposed Context Spaces (Padovitz et al., 2004) extensions incorporated in R-CS allowing R-CS to dynamically model smart spaces using collected sensor data. The smart spaces in Context Spaces are represented as situations. Situation inference is made possible using contextual information from sensor sources. We evaluate the following features of R-CS: 1) Hierarchical context attribute regions, 2) Sensor quality metric incorporation into the Context Spaces error computation algorithm and 3) Dynamic situation modelling using partitioned situation spaces.

Our experimentation results are based on synthetic sensor values using our R-CS simulator presented in Chapter 6. The simulator facilitates defining sensor sources, situation spaces and context state values. Further, the simulator can be used to reason and infer situations based on continuous streams of synthetic sensor data. In each experiment, we define situations with corresponding context attributes. The confidence computed is to determine the occurrence of a situation based on available evidence i.e. context state values from sensor. For example, consider two situations *playing* and *resting* defined with the context attribute *heart rate*. Depending on the value of *heart rate* at instance *t* it is possible to determine the occurrence of the situation *playing* or *resting*. The confidence computed for each situation provides a measure of certainty with which a particular situation's occurrence can be inferred.

7.4.1 Hierarchical Context Attribute Regions- Evaluation

R-CS incorporates multilevel context attributes regions by defining relationships between parent attributes and dependent attributes. For example, reasoning the situation *running/walking* using a set of context attributes age, speed, heart rate, etc, the Context Spaces (CS) model allows us to define the corresponding attribute regions without the ability to define relationships among them. We argue that defining relationships between related attributes would increase the reasoning ability of the context-aware system i.e. speed, heart rate, etc, of a person at an age of 20 to 30 is different from a person at the age of 40 to 50. By defining specific attribute regions based on parent attribute criteria, we reduce the uncertainty in the reasoning process.

In our simulation experiment, we model the two situations *running* and *walking* based on the context attribute definition presented in Table 7.5 and Table 7.6. The simulation experiment takes in consideration an average human being and does deal with specific categories like athletes, etc., though extending the situation definition using hierarchical attributes for different categories of human beings is straightforward.

Situation Attribute	Running Regions			Relevance 0 to 5
Location	"=GYM"	"=PARK"	"!=OFFICE"	2
Age	Sub Regions	"20 - 30"	"40 - 50"	5
	Speed	>= 6 & <= 8	>= 2 & <= 4	5
	Heart Rate	>=93 & <=146	>=83 & <=131	5
	Systolic BP	>=108 & <=122	>=112 & <=130	5

Table 7.5: R-CS Experiment 1: Situation Running Definition

Situation Attribute	Walking Regions			Relevance 0 to 5
Location	"=GYM"	"=PARK"	"=OFFICE"	2
Age	Sub Regions	"20 - 30"	"40 - 50"	5
	Speed	>= 3 & <= 5	>= 1 & < 2	5
	Heart Rate	>=93 & <=143	>=83 & <=127	5
	Systolic BP	>=108 & <=122	>=112 & <=130	5

Table 7.6: R-CS Experiment 1: Situation Walking Definition

The definition of *speed, heart rate and systolic blood pressure (BP)* are defined as sub regions (hierarchical) for the *age* attribute region. With changing context state value for *age* the corresponding attribute region is used to infer the situation. To validate the use of hierarchical attribute regions we compute the overall confidence for each situation being inferred by varying the age parameters to satisfy the parent attribute region predicate. The outcome of the experimentation is presented in Figure 7.31.

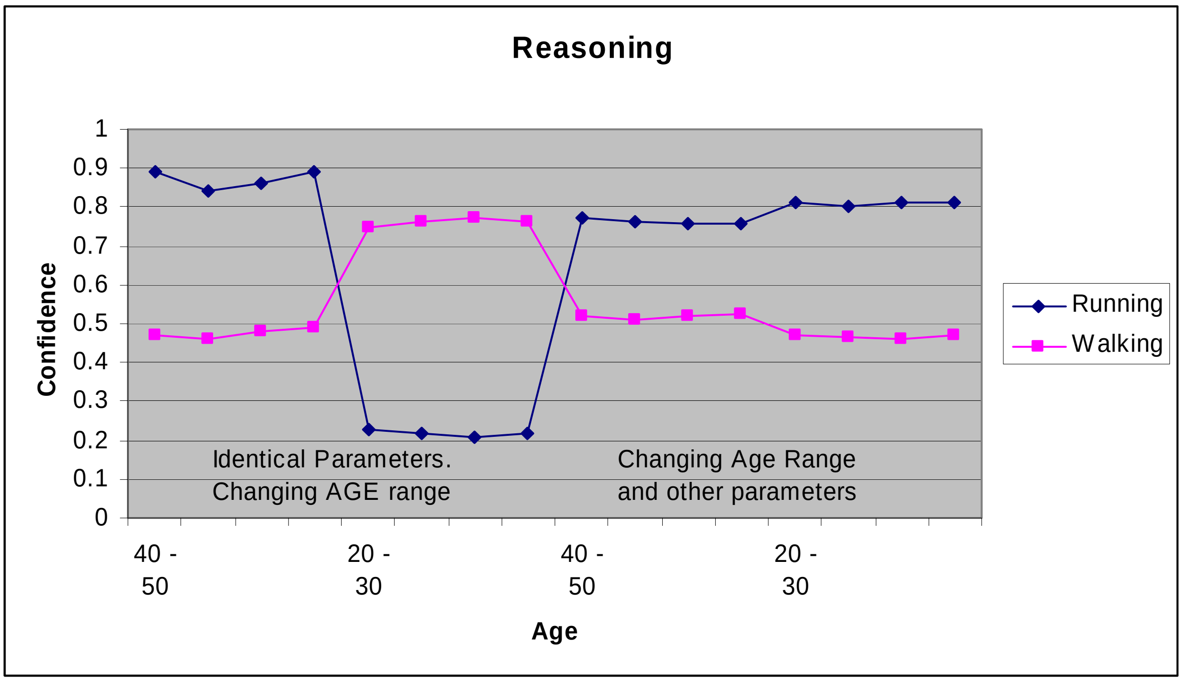


Figure 7.31: R-CS Experiment 1: Hierarchical Attribute Region Evaluation

During the experimentation, for the first two cases, we used overlapping values for the dependent context attributes representing the situations walking/running and changed the age parameter. The outcome of this experiment is reflected in the first 8 outcomes in Figure 7.31. For the rest of the experimentation all the parameters were changed but were kept within the attribute region definitions. The key observation from the experimentation is the ability for R-CS to quickly infer situations based on changing attribute region definitions that are dependent on a parent attribute region. To simulate a similar scenario in CS, we would need to define individual situations for each case, namely: 1) "Running – 40 to 50"; 2) "Walking – 40 – 50"; 3) "Running – 20 – 30"; 4) "Walking – 40 – 50". Further, a direct comparison with CS is not feasible as CS does not have the capability to define attribute region relationships.

7.4.2 Sensor Data Quality and Flexible Attribute Region - Evaluation

In this subsection, we evaluate the proposed sensor data freshness-based error computation algorithm and the flexible attribute region used to determine the contribution of a context attribute. We compare the result of our experimentations with Context Spaces (CS) to validate the importance of introducing sensor data freshness and flexible attribute regions in the reasoning process. The sensor data freshness and the flexible attribute regions have been incorporated into the contribution computation function of R-CS. The simulation experimentation to validate the proposed approach was performed for the situation *presentation*. The definition for the situation *presentation* is presented in Table 7.7. Outcome of the contribution computation for a sample synthetic data is presented in Table 7.8.

Situation: Attributes	Presentation Regions		Weights
LIGHT	">-=10"	"<=20"	0.3
NOISE	" > 20 "	" < 30"	0.25
PROJECTOR	"= ON"	"= OFF"	0.35
PEOPLE	"> 5 "	"< 10 "	0.1

Table 7.7: R-CS Experiment 2: Presentation Situation Definition

Freshness Threshold ≥ 2					CS	R-CS
Context Attribute	Attribute Region	Context State	Error	Freshness	Contribution	
Light	10 to 20	12	90%	1	1	1
Noise	20 to 30	19.95	90%	2	0	1
People	5 to 10	10	100%	3	1	0

Table 7.8: R-CS Experiment 2: Comparing CS and R-CS contribution computation

We discuss two specific cases related to flexible attribute region and sensor data freshness. The R-CS system computes a contribution of 1 for the context attribute *noise,* taking into consideration the outer attribute region definition. For the same context attribute, Context Spaces compute a contribution of 0. Similarly, for the context attribute *people*, CS computes a contribution of 1 while R-CS computes a contribution of 0. This is due to the introduction of freshness threshold used by R-CS to compute the attribute's contribution i.e. sensor data used for reasoning the current situation is old. The outcome of the experimentation is presented in Figure 7.32. The x-axis indicates the number of simulation runs. For each simulation run, a different set of sensor value was used. The y-axis indicates the overall confidence of the reasoned situation computed using CS and R-CS approaches.

For the simulation, we used a freshness threshold value of 2. The reasoning process was performed over 6 synthetic data sets. We note some interesting outcomes from the experimentation which is discussed in detail. The data samples 1, 2, 5 and 6 produce a higher confidence using R-CS approach due to the incorporation of flexible attribute region. In each of those cases, one or more of the context attributes context state values, (current value of the sensor), satisfied the outer range of the context attribute region definition. Since CS does not account for any moderate deviation from the defined attribute regions, a clear difference in the confidence computed is observed between CS and R-CS reasoning. Further, with respect to data samples 3 and 4, we note the confidence computed using R-CS is equal to CS in the first case, while it is lesser than CS in the second case. In the first case, they are equal because all context state values and data freshness values are assumed to be within the defined ranges. Hence, CS and R-CS results are identical. In the second case, the data freshness value of one of the context attributes was generated to be outside the data freshness threshold. In this case R-CS detects the loss in data quality, (old sensor data), influencing the overall computed confidence. The effect of old data does not affect CS reasoning. Even though the computed confidence using R-CS is less than CS, we claim that R-CS outcome is more reliable with lesser error when compared to CS. Hence, the introduction of sensor data quality increases reasoning quality and reliability. This claim is supported and validated by the experimental outcomes. We also deduce from the experimental result that incorporating data freshness heuristic greatly increases the reasoning accuracy.

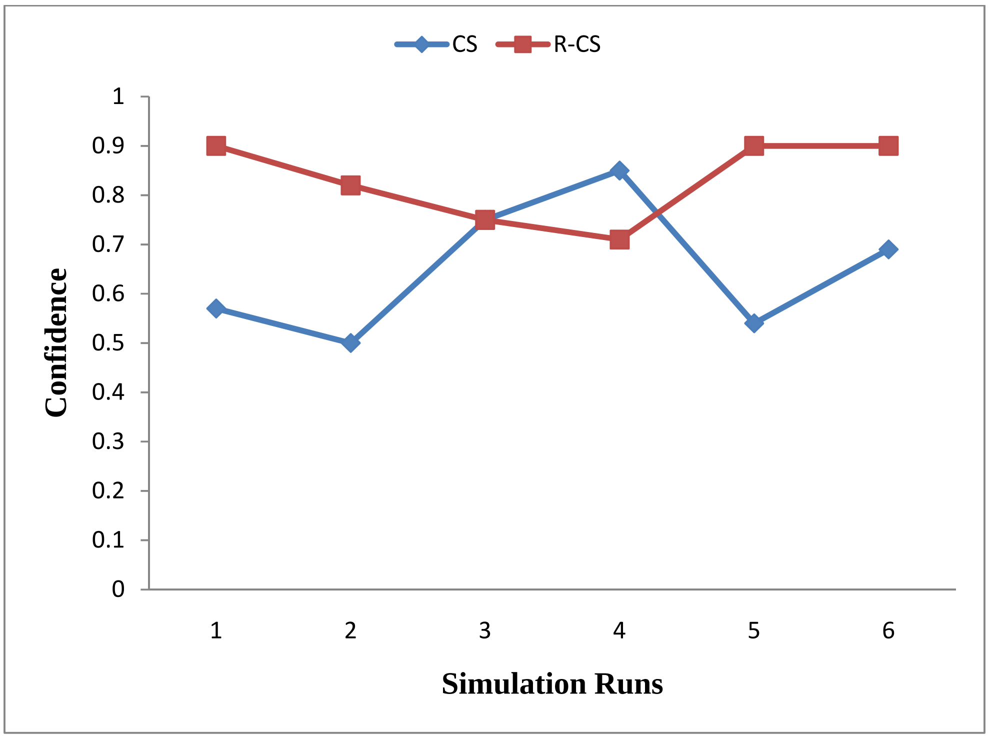


Figure 7.32: R-CS Experiment 2: Evaluating Sensor Data Quality and Flexible Attribute Region

7.4.3 Dynamic Situation Modelling using Partitioned Situation Spaces - Evaluation

In this section, we evaluate the proposed dynamic situation modelling technique that computes temporal situation space definitions on-the-fly using available sensor data. The dynamic situation composition is achieved using the situation partitioning approach.

Reasoning using dynamic situations is further assisted by the use of dynamic context attribute weight re-computation algorithm.

To evaluate and validate the dynamic situation modelling approach we perform experimentation using JAVA-based R-CS implementation. For our experimentation, we have used six context attributes to define the universal situation space. We used two universal situation spaces, namely, *presentation* and *meeting* for the experimentation. The context attribute definitions, situation definitions and importance of context attributes (relevance) are presented in Table 7.9 and Table 7.10. With R-CS implementation weights can be defined within a specified minimum and maximum range which is then normalized during the reasoning process.

The partitioned situation space is computed based on the predicate definition given by the partition conditions (δ). The partition parameter predicate definition used for experimentation is $0 < \delta < 2$, $2<=\delta<3$ and $3<\delta <=5$. The partitioned universal situation space definitions based on the partition parameter are presented in Table 7.11 and Table 7.12. Based on the partitioned universal situation space the temporal situation space is computed by R-CS on-the-fly. The temporal situation space computed during the simulation selects context attributes from corresponding partition based on available sensor data.

Situation:	Presentation			Relevance
Attributes	Regions			(0 to 5)
A1	"= DIM"	"! =ON"	"! =OFF"	3
A2	" > 1 "	" <=3"		3
A3	"= ON"	"!= OFF"		2
A4	">= 2 "	"< 8 "		2
A5	"= Presentation Mode"	"= Standby"	"!= OFF"	0.8
A6	"= Inside"	"!= Outside"		0.9

Table 7.9: R-CS Experiment 3: Situation definitions for Presentation

Situation: Attributes	Meeting Regions			Relevance (0 to 5)
A1	"!= DIM"	"= ON"	" !=OFF"	3
A2	" > =2.5 "	" <=5"		3
A3	"!=ON"	"= OFF"		2
A4	">4 "	"< 10"		2
A5	"!= Presentation Mode"	"= Standby"	"= OFF"	0.8
A6	"= Inside"	"!= Outside"		0.9

Table 7.10: R-CS Experiment 3: Situation definitions for Meeting

Partition	$0 < \delta < 2$	$2<=\delta<3$	$3< \delta <=5$		
Situation Attributes	Presentation Regions			Relevance (0 to 5)	Partition
A1	"= DIM"	"!= ON"	"!= OFF"	3	P1
A2	" > 1 "	" <= 3"		3	P1
A3	"= ON"	"!= OFF"		2	P2
A4	">= 2 "	"< 8 "		2	P2
A5	"= Presentation Mode"	"= Standby"	"!= OFF"	0.8	P3
A6	"= Inside"	"!= Outside"		0.9	P3

Table 7.11: R-CS Experiment 3: Partitioned Situation Space definitions for Presentation

Situation: Attributes	Meeting Regions			Relevance (0 to 5)	Partition
A1	"!= DIM"	"= ON"	"!= OFF"	3	P1
A2	"> 2.5 "	"< =5"		3	P1
A3	"!=ON"	"= OFF"		2	P2
A4	" > 4 "	" <10"		2	P2
A5	"!= Presentation Mode"	"= Standby "	"= OFF"	0.8	P3
A6	"= Inside"	"!= Outside"		0.9	P3

Table 7.12: R-CS Experiment 3: Partitioned Situation Space definitions for Meeting

The CS approach would use the attributes in partition 1 to define the situation space while R-CS uses universal situation space which has the list of all possible context attributes that can describe the situation. At each step, the reasoning is performed by expanding the search into subsequent partitions, re-computing weights until the required confidence threshold is reached. Before we discuss the results of our experimentation we present the computation involved in computing the confidence using CS and R-CS approaches for a set of synthetic sensor data. In each case, we compute the overall confidence of a situation to determine the situation that is currently occurring based on available context attribute. In this case, using the CS and R-CS model, we try to determine the occurrence of the situation *meeting* and *presentation*.

We compute the confidence for each reasoned situation using sensor values ("OFF", 3,) for CS and ("OFF", 3, "ON", "5","Standby","Inside") for R-CS. As CS considers only the primary (first) partition, the confidence computed using CS approach is obtained from context state values for context attributes in partition 1. Further, CS does allow flexible attribute regions i.e. outer and inner range for attribute regions. The reasoning process is presented below.

$$S\ (\text{Presentation}) = \{A_1, A_2\} \text{ and } S\ (\text{Meeting}) = \{A_1, A_2\}$$

$$CS_{\text{presentation}} = A_1 * w_1 + A_2 * w_2 = 0.5 * 0 + 0.5 * 1 = 0.5$$

$$CS_{\text{meeting}} = A_1 * w_1 + A_2 * w_2 = 0.5 * 0 + 0.5 * 1 = 0.5$$

Based on the two context attributes A1 and A2, the confidence computed by CS is 0.5 for each situation. The computed confidence is not sufficient enough to infer the occurrence of a situation. We now compute the confidence of the two situations using the R-CS approach. By expanding the search into other partitions, we define a temporal situation space S_T taking partition 1 and partition 2 into consideration. For this computation, we incorporate flexible context attribute regions. The reasoning outcome is presented below.

$$S_T\ (\text{Presentation}) = \{A_1, A_2, A_3, A_4\} \text{ and } ST\ (\text{Meeting}) = \{A_1, A_2, A_3, A_4\}$$

$$R\text{-}CS_{\text{Presentation}} = 0.3 * 0 + 0.3 * 0.8 + 0.2 * 1 + 0.2 * 1 = 0.64$$

$$R\text{-}CS_{\text{meeting}} = 0.3 * 0 + 0.3 * 1 + 0.2 * 0 + 0.2 * 1 = 0.5$$

The results clearly show improvement in the computed confidence for the situation *presentation*. This is attributed to the use of temporal situation spaces. The temporal situation space dynamically represents the virtual situation using available context information (defined in other partitions). The result of the experiment validates R-CS's advantage over CS. A key observation from our experimentation is the difference between the confidences of the two situations being reasoned. With CS approach the difference is 0, while using R-CS the value is 0.14. We argue this may not be sufficient to infer the situation *presentation*. Our proposed solution to this problem is the dynamic weight re-computation approach. The computations related to the weight re-computation technique for the context attribute in the partitions 1 and 2 are presented in Table 7.13 and Table 7.14.

Temporal Situation:		Presentation					
Attributes	Regions			Relevance (0 to 5)	Initial Weights	Recomputed Weights	Normalized
A1	"= DIM"	"!= ON"	"!= OFF"	3	0.3	0.5	0.36
A2	" > 2 "	" <= 4"		3	0.3	0.3	0.21
A3	"= ON"	"!= OFF"		2	0.2	0.4	0.29
A4	">= 2 "	"< 8 "		2	0.2	0.2	0.14

Table 7.13: R-CS Experiment 3: Weight Re-Computation for Overlapping Regions (Presentation)

Temporal Situation:		Meeting					
Attributes	Regions			Relevance (0 to 5)	Initial Weights	Recomputed Weights	Normalized
A1	"!= DIM"	"= ON"	"= OFF"	3	0.3	0.5	0.36
A2	"> 2.5 "	"<=5"		3	0.3	0.3	0.21
A3	"!=ON"	"= OFF"		2	0.2	0.4	0.285714
A4	" > 4 "	" < =10"		2	0.2	0.2	0.142857

Table 7.14: R-CS Experiment 3: Weight Re-Computation for Overlapping Regions (Meeting)

We re-compute the confidence for the situations *meeting* and *presentation* using the newly computed weights. The confidence calculation is presented below. The weight re-computation is round 2 of the R-CS reasoning technique.

$$C_{Presentation} = 0.36 * 0 + 0.21 * 0.8 + 0.29 * 1 + 0.14 * 1 = 0.6$$

$$C_{meeting} = 0.36 * 0 + 0.21 * 1 + 0.29 * 0 + 0.14 * 1 = 0.35$$

The following are the inferences from the above results obtained using R-CS. The computed confidence for the situation *presentation* reduces by a very small percentage over round 1. But the difference in the confidence between the two situations has increased significantly. The weight re-computation algorithm aims to increase the certainty and hence the accuracy of the reasoning process. The higher the confidence of the situation being reasoned, the higher is the certainty of its occurrence. Uncertainty in this case is introduced by how accurately the current situation can be inferred from a list of other possible situations.

We now present the result of our simulation experimentation for a set of synthetic sensor data for which confidence is computed using R-CS and CS approaches. For evaluation, we compute the difference in confidence for the reasoned situation. This evaluation is used to verify the improvement in the reasoning ability of R-CS over CS. The simulation outcomes are presented in Figure 7.33, Figure 7.34 and Figure 7.35. The results are presented in the following format:

1) Figure 7.33 presents the reasoning outcomes using CS for 10 sets of synthetic sensor values.
2) Figure 7.34 presents the reasoning outcomes using partitioned situation space-based R-CS approach for the same set of 10 synthetic sensor values.
3) Finally Figure 7.35 presents a comparison of difference in confidence computed for each situation using CS and R-CS approaches.

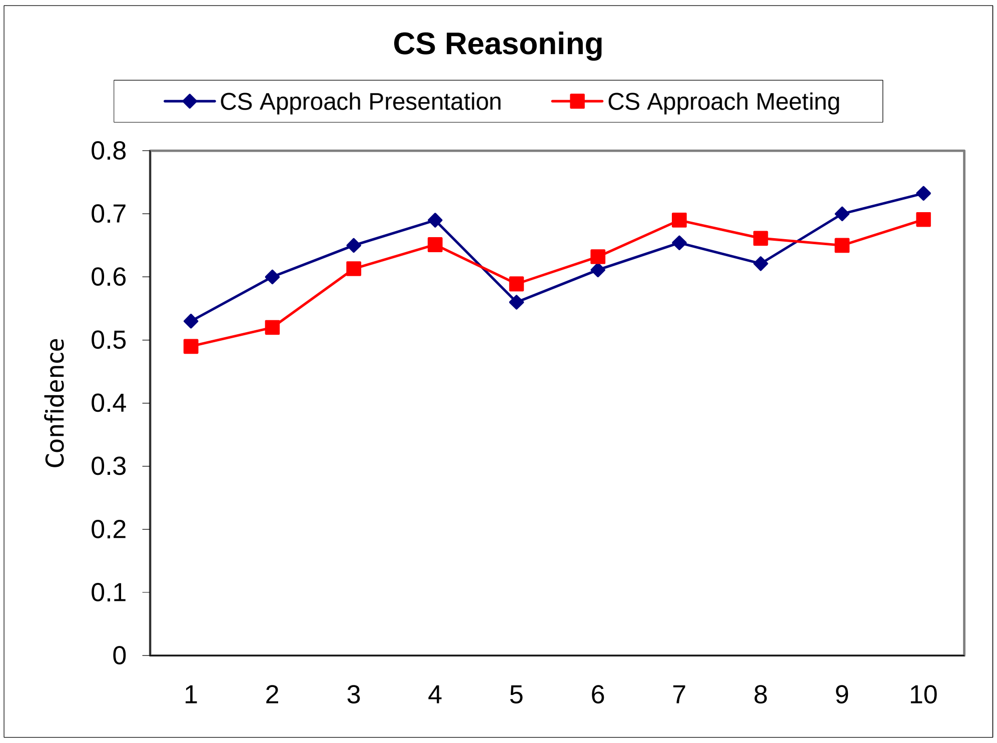


Figure 7.33: R-CS Experiment 3 - CS Reasoning Results

The key observations of the experiments are discussed further. From the result presented in Figure 7.33, we note that difference in the confidence between the situations being reasoned is less for CS. This reduces Context Spaces ability to infer the occurrence of the situation with high certainty. By certainty, we refer to the ability of the reasoning process to infer the situation (currently occurring) correctly. The result presented in Figure 7.34 shows the outcome of confidence computation using R-CS based reasoning. By incorporating partitioned situations and weight re-computations, we note that computed confidence for the situation *presentation* is higher compared to CS. For example, in case 3, CS computes a confidence of 0.6 for the situation *presentation* while R-CS computes a confidence of 0.7. The result clearly indicates the importance of dynamic situation modelling based on available context information. The result also validates the reasoning effectiveness of R-CS over CS approach.

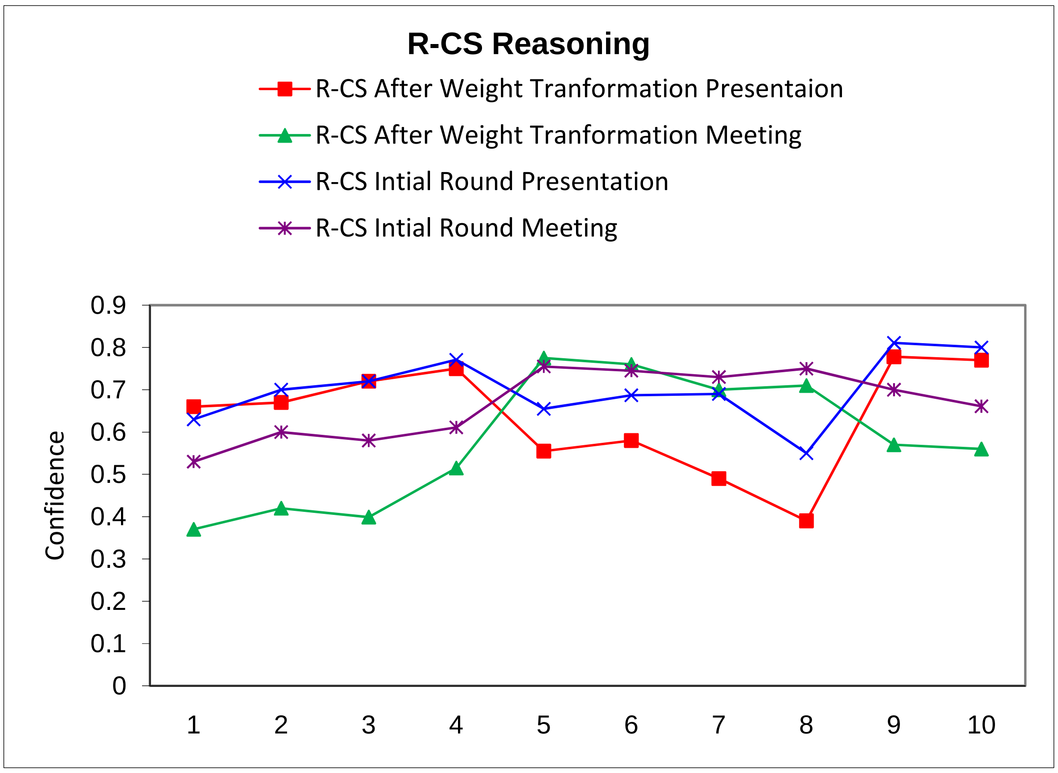


Figure 7.34: R-CS Experiment 3 - R-CS Reasoning Results

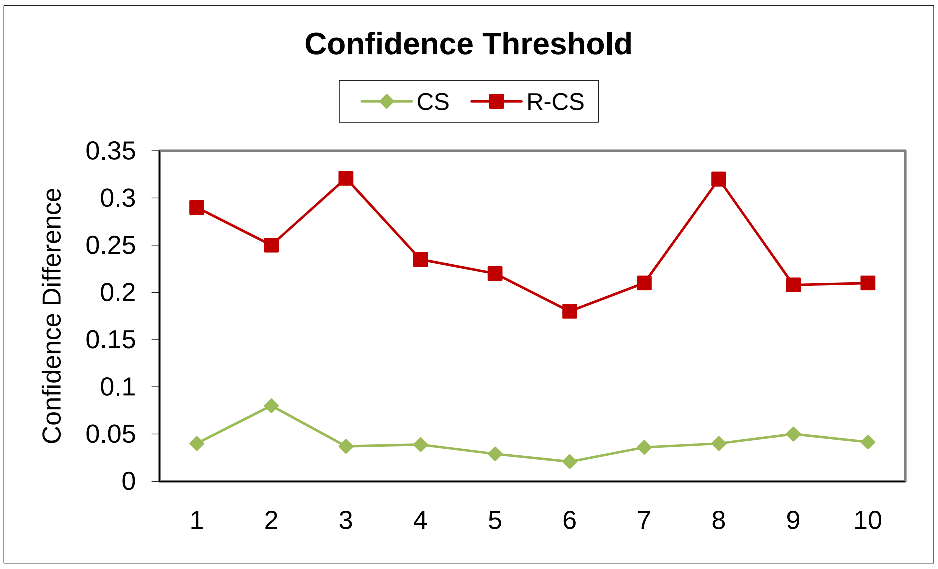


Figure 7.35: R-CS Experiment 3: Difference in Confidence of Reasoned Situations using CS and R-CS

To further exemplify the improvement in the reasoning process using R-CS, we have computed the difference in the computed confidences for the situations being reasoned. The graph presented in Figure 7.35 is the outcome of this computation. As the result indicates, the difference in confidences computed by CS is less than 0.05 in most cases. This value may not be sufficient to infer the current situation, as the results indicate the likelihood of occurrence of both situations. Since, the situations are orthogonal (cannot occur parallel), this is not a possibility. For example, in case 3, the confidences computed for *presentation* and *meeting* using CS is *0.57* and *0.6* and R-CS is *0.7* and *0.4* respectively. With CS reasoning, it may be difficult for the context-aware system (system that uses the CS context model) to determine the occurring situation as the confidences of both situations are almost identical. By contrast, with R-CS approach, the considerable difference in computed confidences increases R-CS reasoning capability to infer the situation *presentation* with higher confidence. This is indicated from R-CS results presented in Figure 7.35. The R-CS results show an average difference of 0.3. The experimental results verify and validate the enhancement in the reasoning process using the proposed R-CS model.

7.5 Summary

This chapter has presented outcomes of experiments performed to verify and validate our algorithms and methodologies proposed in chapters 3, 4 and 5.

Our experimental evaluations of the proposed sensor data collection framework using day-today mobile devices (mobile data mule) have significant potential for energy saving. Our approach can be considered as a new paradigm for data collection in future pervasive environments. Further, our experimental evaluations for data collection using *kNN* queries perform better than current sensor *kNN* query processing algorithms within our test simulation environment, exhibiting significant savings in sensor energy and hence increasing sensor network lifetime. Finally the Context Spaces extensions incorporated in R-CS have been validated with sufficient experimental evaluations. The results prove the enhanced reasoning ability of R-CS over CS reasoning.

8

Conclusions and Future Work

Data collection is a key area that needs to be addressed for widespread adoption of sensor networks. Sensor networks play a vital role in pervasive computing enabling sensing data from physical environments. They provide access to environmental data which in the past may not be feasible. Sensor networks have helped realise the pervasive computing vision.

This thesis has addressed the challenges of sensor data collection and dynamic modelling of smart situation spaces using collected sensor data. We conclude the thesis by highlighting the contributions and the discussing the lessons learnt in addressing the three research questions presented in section 1.3.

8.1 Contributions of the Thesis Work

A system framework for sensor data collection (sGaRuDa): Highlighting the first research question, we investigated the feasibility of using *day-to-day* mobile devices as an energy-efficient alternative to collect and deliver data from sensor networks using short range communication technologies (e.g. Bluetooth-based sensor networks). Mobile devices, more specifically mobile phones usually have enough spare capacity, (battery and processing), resulting from the few hours of usage. We exploit the spare capacity available on mobile phones to build a mobile access network for wireless sensor networks. The proposed system framework, sGaRuDa is a data collection architecture that facilitates *day-to-day* mobile devices (e.g. mobile phones) to act as sensor data collectors. The proposed system facilitates cost-efficient sensor data collection compared to fixed data collection infrastructure. Moreover, in many applications, the proposed system can function in the absence of a fixed data collection infrastructure. The use of mobility results in considerable energy savings. Specifically, the use of mobile data mules can result in an energy saving in the range of 10% to 40%. This saving in energy can be directly related to extension in overall network lifetime of at least 10%. Extensive evaluation and experimentation presented in this thesis verify and validate these results. Further, the proposed system does not require a dedicated mobile sensor data collector to be present within the sensor network environment. The system relies on user mobility and mobile devices within smart spaces to accomplish sensor data collection. Adopting the proposed sGaRuDa sensor data collection architecture may result in reduced sensor network setup time, involving less planning and lesser infrastructure costs. A proof-of-concept implementation was presented to validate the feasibility of the proposed system in real-world scenarios. The use of mobile devices as an access network creates much more application possibilities few of which were presented in the thesis. Based on our experimentations and prototype implementations, we conclude that it is feasible to employ mobile devices as a cost-efficient alternative to collect sensor data. The proposed sGaRuDa architecture has the following advantages:

- sGaRuDa, the system framework, is software based and takes advantage of ubiquitous availability of Bluetooth. Hence, no special hardware is required to communicate with the sensor node.
- Mobility is accomplished by taking advantage of user mobility. Hence, our approach does not require introduction of special hardware to realise device mobility.
- The sGaRuDa system is interoperable and platform independent. This has been verified by the prototype implementation.

At the outset, the proposed sGaRuDa framework has certain limitations that need to be addressed for widespread adaptation. These limitations include security, privacy and accounting. We treat the limitations as future research challenges.

***A Data Collection algorithm for Broadcast-based Sensor Networks* (3D-KNN*)*:**

The 3D-KNN algorithm was proposed to address the second research question investigating the extension of the sensor data collection architecture to suit a range of sensor network platforms with broadcasting capabilities. The proposed 3D-KNN algorithm facilitates cost-efficient multi-hop collection of sensor data using mobile data mules. Multi-hop data collection is facilitated using a *k*-Nearest Neighbour query processing technique. The proposed algorithm has the capability to account for real-world communication channel characteristics (e.g. obstacles, channel error, etc.) in 3D spaces. This feature of *3D-kNN* differentiates it from existing approaches that focus only on distance with the assumption of an error-free communication channel. We note that the proposed 3D-KNN algorithm is a pioneering work in the area of *kNN*-based sensor data collection using mobile data mules in an infrastructure-less three-dimensional sensor network. We proposed energy-efficient algorithms for discovery, sensing and data collection keeping in mind the resource constrained nature of wireless sensors. We performed experimental evaluation of the 3D-KNN algorithm within large-scale sensor networks in a simulator environment. The results validate the extent of energy saving using the proposed 3D-KNN algorithm. More specifically, the 3D-KNN algorithm had a linear expenditure of energy for increasing sensor network sizes compared to *kNN*-based data collection algorithm in the literature which exhibited a non-linear increase in energy consumption. Further, the proposed predictive 3D-KNN approach results in 40% savings in energy over the non-predictive approach. This result is significant and may results in extended sensor network lifetimes. The 3D-KNN algorithm also proved to be performance oriented, reducing the overall time required to process the *kNN* queries. The evaluation outcomes prove the cost-efficiency (time and energy efficiency) of the 3D-KNN algorithm. Simulation experiments also verify the suitability of the 3D-KNN algorithm for large-scale sensor networks. In conclusion, the proposed 3D-KNN employed by the mobile data mule proves to be more energy-efficient and performance oriented approach to collect data from broadcast-based sensors. The 3D-KNN algorithm has the following advantages

- The 3D-KNN algorithm is completely dynamic i.e. no prior network infrastructure information is required to compute nearest neighbours. Further, no specific assumption of the sensor network deployment/topology is made.
- The 3D-KNN algorithms accounts for three dimensional distribution of sensors in smart spaces.
- The 3D-KNN algorithm addresses real-world radio communication characteristics like obstacles and interference by taking into account SNR and device mobility rather than just distance.

***A Dynamic Smart Situation Spaces Modelling Approach (R-CS)*:** Finally we addressed the third research question by developing a dynamic situation-based context model that has the capability to adapt to changing sensor data. We proposed R-CS, a ranked-Context Spaces model based on Context Spaces theory that has the ability to model situations dynamically. The proposed extensions to Context Spaces theory have been incorporated into R-CS. Dynamic situation modelling in R-CS is achieved using partitioned and temporal situation spaces. By introducing dynamic situation space modelling in Context Spaces, we facilitate reasoning under uncertainty when the real-world situation definition changes dynamically. We performed experiments to evaluate, verify and validate the Context Spaces extensions incorporated into R-CS. The Context Spaces extensions incorporated into R-CS have a considerable impact on the reasoning outcome. More specifically, R-CS is able to infer situations with better accuracy. Moreover, R-CS evaluation results validate the advantage of using dynamic situation modelling to reduce ambiguity, when a fixed situation space definition as used by Context Spaces may not be sufficient to infer situations. The R-CS system has the following advantages:

- R-CS algorithms perform reasoning over adapted situation definitions, computed using available sensor data (collected and delivered by mobile data mules).

- R-CS model works towards best representing the current situation rather than relying on fixed situation definitions
- R-CS has the capability to account for sensor data quality and flexible attribute regions making it more susceptible to changing sensor data.

8.2 Future Work

The research questions addressed in this thesis have created new opportunities for further research. We highlight some of them in this section.

Pay for Resources: The use of mobile devices as sensor data collectors opens the door for numerous applications. The proposed approach creates the platform to bridge low-powered sensors with the external world in an energy-efficient way. The use of mobile devices creates a new area of work which needs to look into mobile usage i.e. accounting for resources being used. This might be creating a commercial model that can be used by Telcos to reward its customers for the use of mobile devices for sensor data collection. Further, this model can be extended to enterprises that would need real-time information from pervasive environments at a low cost.

Security and Privacy: The use of *day-to-day* mobile devices introduces privacy and security risks. These concerns need to be addressed for adoption of the sGaRuDa framework. Security and privacy is an important topic of debate, especially within environments comprising mobile devices. A number of existing techniques may be adapted to suit the sGaRuDa system requirements. We suggest this as an interesting area of future work.

Sensor Errors: Sensor errors play a vital role in the data collection process. For example, the reported location of a sensor and its actual location might influence the energy consumption during communication. Current approaches focusing on sensor localisation fail to address the issue during computation (making decisions based on sensor data). We suggest this as a future extension to the proposed data collection algorithms.

Ontology in R-CS: The R-CS reasoning ability can be improved further by introducing ontology. The use of ontology can reduce uncertainty in cases where the newly discovered contextual information does not have any relation to the current situations. In such cases, using ontology, an extensive search can be performed to determine the relation between newly discovered context and current situation.

References

Abbasi A.A. & Younis M., 2007. *A survey on clustering algorithms for wireless sensor networks*. Comput. Commun., vol. 30, no. 14-15, pages: 2826-2841.

Abowd G.D., Atkeson C.G., Hong J., Long S., Kooper R. & Pinkerton M., 1997. *Cyberguide: a mobile context-aware tour guide*. Wireless Networking, vol. 3, no. 5, pages: 421-433.

Abowd G.D., Dey A.K., Brown P.J., Davies N., Smith M. & Steggles P., 1999. *Towards a Better Understanding of Context and Context-Awareness*, In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, pages: 304-307

Acharya D., Kumar V., Garvin N., Greca A. & Gaddis G.M., Year. A sun SPOT based automatic vehicular accident notification system. *In:* Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on, 2008. 296-299.

Akman V. & Surav M., 1997. *The use of situation theory in context modelling, Computational Intelligence*. Computational Intelligence: An International Journal, vol. 13(3), no., pages: 427–438.

Akyildiz I.F. & Kasimoglu I.H., 2004. *Wireless Sensor and Actor Networks: Research Challenges*. Ad Hoc Networks Journal (Elsevier), vol. 2, no. 4, pages: 351-367.

Akyildiz I.F., Weilian S., Sankarasubramaniam Y. & Cayirci E., 2002. *A survey on sensor networks*. Communications Magazine, IEEE, vol. 40, no. 8, pages: 102-114.

Al-Karaki J.N. & Kamal A.E., 2004. *Routing techniques in wireless sensor networks: a survey*. Wireless Communications, IEEE, vol. 11, no. 6, pages: 6-28.

Alippi C. & Galperti C., 2008. *An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes*. Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 55, no. 6, pages: 1742-1750.

Amirtharajah R. & Chandrakasan A.P., 2004. *A Micropower Programmable DSP Using Approximate Signal Processing Based on Distributed Arithmetic*. IEEE Journal of Solid-State Circuits, vol. 39, no. 2, pages: 337-347.

Anastasi G., Conti M. & Francesco M., 2009a. *An Analytical Study of Reliable and Energy-Efficient Data Collection in Sparse Sensor Networks with Mobile Relays*, In Proceedings of the 6th European Conference on Wireless Sensor Networks, Cork, Ireland, pages: 199-215

Anastasi G., Conti M. & Francesco M.D., 2009b. *Reliable and energy-efficient data collection in sparse sensor networks with mobile elements*. Perform. Eval., vol. 66, no. 12, pages: 791-810.

Anastasi G., Conti M., Francesco M.D. & Passarella A., 2009c. *Energy conservation in wireless sensor networks: A survey*. Ad Hoc Networks, vol. 7, no. 3, pages: 537-568.

Anastasi G., Conti M., Passarella A. & Pelusi L., 2008. *Mobile-relay Forwarding in Opportunistic Networks*. *In:* IBNKAHLA, M. (ed.) *Chapter in Adaptive Techniques in Wireless Networks.* New York (USA): CRC Press. pages:

Arampatzis T., Lygeros J. & Manesis S., 2005. *A Survey of Applications of Wireless Sensors and Wireless Sensor Networks*, In Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation, Limassol, Cyprus, pages: 719 - 724

Asada G., Dong M., Lin T., Newberg F., Pottie G., Marcy H. & Kaiser W., 1998. *Wireless integrated network sensors: Low-power systems on a chip*, In Proceedings of the 24th IEEE European Solid-State Circuits Conference, Den Hague, The Netherlands, pages: 9-12

Atmel, 2009a. *AT91RM9200* [Online]. Available: http://www.atmel.com/dyn/products/Product_card.asp?part_id=2983 [Accessed 29/03/2010].

Atmel, 2009b. *ATmega128 - 8-bit Microcontroller with 128K Bytes In-System Programmable Flash* [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf [Accessed 29/03/2010].

Aurenhammer F., 1991. *Voronoi diagrams- a survey of a fundamental geometric data structure*. ACM Comput. Surv., vol. 23, no. 3, pages: 345-405.

Bagci F., Petold J., Trumler W. & Ungerer T., 2004. *Ubiquitous Mobile Agent System in a P2P-Network, System Support for Ubiquitous Computing*, In Workshop at the Sixth Annual Conference on Ubiquitous Computing (UbiComp'04), Nottingham, England, pages: 12-15

Baker C.R., Armijo K., Belka S., Benhabib M., Bhargava V., Burkhart N., Minassians A.D., Dervisoglu G., Gutnik L., Haick M.B., Ho C., Koplow M., Mangold J., Robinson S., Rosa M., Schwartz M., Sims C., Stoffregen H., Waterbury A., Leland E.S., Pering T. & Wright P.K., 2007. *Wireless Sensor Networks for Home Health Care*, In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, pages: 832-837

Bandyopadhyay S. & Coyle E.J., 2003. *An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks*, In IEEE INFOCOM, pages: 1183-1193

Basagni S., Carosi A., Melachrinoudis E., Petrioli C. & Wang Z.M., 2007. *Controlled sink mobility for prolonging wireless sensor networks lifetime*. ACM/Elsevier Journal on Wireless Networks, vol. 54, no. 6, pages: 1895-1822.

Bayes M. & Price M., 1763. *An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S.* Philosophical Transactions, vol. 53, no., pages: 370-418.

Becker C. & Dürr F., 2005. *On location models for ubiquitous computing*. Personal Ubiquitous Comput., vol. 9, no. 1, pages: 20-31.

Bellotti V. & Sellen A., 1993. *Design for Privacy in Ubiquitous Computing Environments*, In Proc. of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), pages: 77-92

Bergbreiter S. & Pister K.S.J., 2003. *CostBots: An Off-theShelf Platform for Distribuited Robotics*, In Proc. of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pages: 1632 - 1637

Bluecove, 2007. Available: http://sourceforge.net/projects/bluecove [Accessed 10/01/2007].

Bluetoothsig, 2010a. *Bluetooth Homepage* [Online]. Available: http://www.bluetooth.com/English/Pages/default.aspx [Accessed 05/05/2010].

Bluetoothsig, 2010b. *Bluetooth Low Energy Technology* [Online]. Available: http://www.bluetooth.com/English/Products/Pages/Low_Energy.aspx [Accessed 01/03/2010].

Bluetoothsig, 2010c. *Bluetooth Special Interest Group* [Online]. Available: http://www.bluetooth.org [Accessed 26/03/2010].

Bluetoothsig, 2010d. *Bluetooth Specification Documents* [Online]. Available: http://www.bluetooth.com/GERMAN/TECHNOLOGY/BUILDING/Pages/Specif cation.aspx [Accessed 10/05/2010].

Brown P.J., Bovey J.D. & Chen X., 1997. *Context-Aware Applications: From the Laboratory to the Marketplace*. IEEE Personal Communications, vol. 4, no. 5**,** pages: 58-64.

Btaccess, 2008. *Technical Whitepapers - Developing Bluetooth Enabled Applications using BTAccess* [Online]. Available: http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c 108973a801?ciid=f76d2a73c503b110VgnVCM100000a360ea10RCRD [Accessed 01/06/2010].

Buchholz T., Krause M., Linnhoff-Popien C. & Schiffers M., 2004. *CoCo: dynamic composition of context information*, In The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. , pages: 335-343

Buchholz T., Kupper A. & Schiffers M., 2003. *Quality of context: What it is and why we need it*, In Proceedings of the 10th HP–OVUA Workshop, Geneva, Switzerland, pages:

Bult K., Burstein A., Chang D., Dong M., Fielding M., Kruglick E., Ho J., Lin F., Lin T., Kaiser W., Marcy H., Mukai R., Nelson P., Newburg F., Pister K.S.J., Pottie G., Sanchez H., Stafsudd O., Tan K., Xue S. & Yao J., 1996. *Low power systems for wireless microsensors*, In Proceedings of the 1996 international symposium on Low power electronics and design, Monterey, California, United States, pages: 17-21

Burke J., Friedman J., Mendelowitz E., Park H. & Srivastava M.B., 2006. *Embedding expression: Pervasive computing architecture for art and entertainment*. Pervasive and Mobile Computing, vol. 2, no. 1**,** pages: 1-36.

Butler Z. & Rus D., 2003. *Event-Based Motion Control for Mobile-Sensor Networks*. IEEE Pervasive Computing, vol. 2, no. 4**,** pages: 34-42.

Byun H.E. & Cheverst K., 2003. *Supporting Proactive 'Intelligent' Behaviour: the Problem of Uncertainty*, In Proceedings of the UM03 Workshop on User Modeling for Ubiquitous Computing, Johnstown, PA, pages: 17-25

Campbell A.T., Eisenman S.B., Lane N.D., Miluzzo E. & Peterson R., 2006. *People-Centric Urban Sensing*, In Proc. of Second ACM/IEEE Annual International Wireless Internet Conference (WICON 2006), Boston, Massachusetts, USA, pages:

Capra L., Emmerich W. & Mascolo C., 2001. *Reflective Middleware Solutions for Context-Aware Applications*, In Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, pages: 126-133

Castro P. & Munz R., 2000. *Managing context data for smart spaces*. IEEE Personal Communications vol. 7, no. 5, pages: 44-46.

Chakrabarti A., Sabharwal A. & Aazhang B., 2003. *Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks*, In Proc. of the 2nd International Workshop on Information Processing in Sensor Networks, Palo Alto, CA, USA, pages: 129-145

Chalmers D., 2002. *Contextual Mediation to support Ubiquitous computing.* PhD Thesis, Imperial College.London

Chandrakasan A., Min R., Bhardwaj M., Cho S. & Wang A., 2002. *Power aware wireless microsensor systems*, In Proceedings of the 28th European Solid-State Circuits Conference, ESSCIRC 2002. , pages: 47-54

Chatzigiannakis I., Kinalis A. & Nikoletseas S., 2006. *Sink mobility protocols for data collection in wireless sensor networks*, In Proceedings of the 4th ACM international workshop on Mobility management and wireless access, Terromolinos, Spain, pages: 52-59

Chen H., Finin T. & Joshi A., 2003. *Using OWL in a Pervasive Computing Broker*, In Workshop on Ontologies in Agent Systems, AAMAS-2003, Melbourne, Australia, pages:

Chong S.K., Krishnaswamy S., Loke S.W. & Gaben M.M., 2008. *Using association rules for energy conservation in wireless sensor networks*, In Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, pages: 971-975

Chu D., Deshpande A., Hellerstein J.M. & Hong W., 2006. *Approximate Data Collection in Sensor Networks using Probabilistic Models*, In Proceedings of the 22nd International Conference on Data Engineering, pages: 42-48

Cogill R., 2009. *Poisson Processes* [Online]. Available: http://people.virginia.edu/~rlc9s/sys6005/SYS_6005_Poisson_Proc.pdf [Accessed 25/04/2010].

Coleri S., Cheung S.Y. & Varaiya P., 2004. *Sensor Networks for Monitoring Traffic*, In Forty-Second Annual Allerton Conference on Commuinication, Control, and Computing, pages:

Cordeiro C.D.M. & Agrawal D.P., 2006. *Ad Hoc And Sensor Networks-Theory and Applications*, World Scientific Publishing Co. Pte. Ltd.

Crossbowtechnology, 2010a. *Crossbow Product Reference Guide* [Online]. Available: http://www.xbow.com/Support/Support_pdf_files/Product_Feature_Reference_Chart.pdf [Accessed 29/03/2010].

Crossbowtechnology, 2010b. *TelosB - TelosB Mote Platform* [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf [Accessed 29/03/2010].

Culler D., Estrin D. & Srivastava M.B., 2004. *Overview of Sensor Networks*. Computer, vol. 37, no. 8, pages: 41-49.

Curino C., Giani M., Giorgetta M., Giusti A., Murphy A.L. & Picco G.P., 2005. *Mobile Data Collection in Sensor Networks: The TinyLIME Middleware*. Pervasive and Mobile Computing, vol. 1, no. 4, pages: 446-469.

Dantu K., Rahimi M., Shah H., Babel S., Dhariwal A. & Sukhatme G.S., 2005. *Robomote: enabling mobility in sensor networks*, In Proceedings of the 4th international symposium on Information processing in sensor networks, Los Angeles, California, pages: 404-409

Dawson F. & Stenerson D., 1998. *Internet Calendaring and Scheduling Core Object Specification (iCalendar)* [Online]. Available: http://www.ietf.org/rfc/rfc2445.txt [Accessed 11/05/2010].

Demirbas M. & Ferhatosmanoglu H., 2003. *Peer-to-Peer Spatial Queries in Sensor Networks*, In Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pages: 32

Deventer J.V., Gustafsson J., Delsing J. & Eliasson J., 2009. *Wireless Infrastructure in a District Heating Substation*, In 3rd Annual IEEE International Systems Conference, Vancouver, Canada, pages: 139 - 143

Dey A.K., 2001. *Understanding and Using Context*. Personal Ubiquitous Comput., vol. 5, no. 1, pages: 4-7.

Dey A.K., Abowd G.D. & Salber D., 2001. *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*. Computer Interaction (HCI) Journal, vol. 16, no. 2, pages: 97-166.

Dey A.K. & Mankoff J., 2005. *Designing mediation for context-aware applications*. ACM Trans. Comput.-Hum. Interact., vol. 12, no. 1, pages: 53-80.

Di Francesco M., Shah K., Kumar M. & Anastasi G., 2010. *An Adaptive Strategy for Energy-Efficient Data Collection in Sparse Wireless Sensor Networks*, In in Proceedings of 7th European Conference Wireless Sensor Networks, EWSN 2010., Coimbra, Portugal, pages: 322-337

Dini G., Pelagatti M. & Savino I.M., 2008. *An Algorithm for Reconnecting Wireless Sensor Network Partitions*, In Proc. of the 5th European conference on Wireless Sensor Networks (EWSN 2008), Bologna (Italy), pages: 253-267

Doan A., Madhavan J., Domingos P. & Halevy A., 2002. *Learning to map between ontologies on the semantic web*, In 11th international World Wide Web Conference, Honolulu, Hawaii, USA, pages: 662 - 673

Dunkels A., 2010. *The Operating System for Embedded Smart Objects - the Internet of Things* [Online]. Available: http://www.sics.se/contiki/ [Accessed 29/03/2010].

Eclipse.Org, 2010. *Open Source IDE* [Online]. Available: http://www.eclipse.org/ [Accessed 01/06/2010].

Eislab, 2010. *The Mulle - A node for Bluetooth Sensor Networks / Development information* [Online]. Available: http://www.ltu.se/csee/research/eislab/areas/mixedmode/projects/mulle [Accessed 28/03/2010].

Eistec, 2009. *Mulle Hardware Guide* [Online]. Available: http://www.eistec.se/docs/Mulle_HW_Guide.pdf [Accessed 10/03/2010].

Ekahau, 2010. *WiFi-based tracking* [Online]. Available: http://www.ekahau.com/ [Accessed 11/02/2010].

Eliasson J., Lindgren P. & Delsing J., 2008. *A Bluetooth-based Sensor Node for Low-Power Ad Hoc Networks*. Journal of Computers, vol. 3, no. 5, pages: 1-10.

Eliasson J., Lindgren P., Delsing J., Thompson S.J. & Cheng Y.-B., 2007. *A power management architecture for sensor nodes*, In IEEE Wireless Communications and Networking Conference, pages: 3008 - 3013

Eliasson J., Lundberg M. & Lindgren P., 2006. *Time synchronous Bluetooth sensor networks*, In 3rd IEEE Consumer Communications and Networking Conference, 2006. CCNC 2006. , pages: 336-340

Elson J. & Romer K., 2003. *Wireless sensor networks: a new regime for time synchronization*. SIGCOMM Comput. Commun. Rev., vol. 33, no. 1, pages: 149-154.

Erricson, 2010. *Erricson Mobile Platforms* [Online]. Available: http://www1.ericsson.com/solutions/mobileplatforms [Accessed 07/05/2010].

Eth-Zurich, 2007. *BTnode rev3 Hardware Reference* [Online]. Available: http://www.btnode.ethz.ch/Documentation/BTnodeRev3HardwareReference [Accessed 29/03/2010].

Eth-Zurich, 2010. *The Sensor Network Museum* [Online]. Available: http://www.snm.ethz.ch/Main/HomePage [Accessed 29/03/2010].

Evolution-Robotics, 2010. Available: http://www.evolution.com/ [Accessed 01/02/2010].

Fortune S., 1986. *A sweepline algorithm for Voronoi diagrams*, In Proceedings of the second annual symposium on Computational geometry, Yorktown Heights, New York, United States, pages: 313-322

Fox D., Hightower J., Liao L., Schulz D. & Borriello G., 2003. *Bayesian filtering for location estimation*. IEEE Pervasive Computing, vol. 2, no. 3, pages: 24-23.

Frodigh M., Johansson P. & Larsson P., 2000. *Wireless Ad Hoc Networking- The Art of Networking Without A Network* [Online]. Ericsson Review. Available: http://www.ericsson.com/ericsson/corpinfo/publications/review/2000_04/124.shtml [Accessed 01/03/2010].

Gandham S.R., Dawande M., Prakash R. & Venkatesan S., 2003. *Energy efficient schemes for wireless sensor networks with multiple mobile base stations*, In Proc. of IEEE Globecom 2003, San Francisco, CA, pages: 377-381

Ganesan D., Ratnasamy S., Wang H. & Estrin D., 2004. *Coping with irregular spatio-temporal sampling in sensor networks*. SIGCOMM Comput. Commun. Rev., vol. 34, no. 1, pages: 125-130.

Gharavi H. & Kumar S.P., 2003. *Special issue on sensor networks and applications*. Proceedings of the IEEE, vol. 91, no. 8, pages: 1151-1153.

Ghiasi S., Srivastava A., Yang X. & Sarrafzadeh M., 2002. *Optimal Energy Aware Clustering in Sensor Networks*. MDPI Sensors, vol. 2, no. 7, pages: 258-269.

Glassey R., Stevenson G., Richmond M., Nixon P., Terzis S., Wang F. & Ferguson I., 2003. *Towards a Middleware for Generalised Context Management*, In First International Workshop for Middleware for Pervasive and Ad Hoc Computing, Rio De Janeiro, Brazil, pages: 45-52

Glomosim, 2010. *Global Mobile Information Systems Simulator Library* [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/ [Accessed 01/07/2010].

Google, 2010. *Android* [Online]. Available: http://www.android.com/ [Accessed 01/06/2010].

Goslar K. & Schill A., 2004. *Modeling Contextual Information Using Active Data Structures*, In Current Trends in Database Technology - EDBT 2004 Workshops, pages: 325-334

Gostev A., 2006. *Bluetooth: London 2006* [Online]. London. Available: http://www.securelist.com/en/analysis?pubid=188833782 [Accessed 06/10/2010].

Grapenetworks, 2010. Available: http://www.grapenetworks.com/ [Accessed 01/06/2010].

Gu T., Pung H.K. & Zhang D.Q., 2004. *Toward an OSGi-based infrastructure for context-aware applications*. Pervasive Computing, IEEE, vol. 3, no. 4, pages: 66-74.

Gu Y., Bozdag D., Brewer R.W. & Ekici E., 2006. *Data harvesting with mobile elements in wireless sensor networks*. Comput. Netw., vol. 50, no. 17, pages: 3449-3465.

Gutnik V. & Chandrakasan A., 1997. *Embedded Power Supply for Low-Power DSP*. IEEE Trans. on VLSI Systems, vol. 5, no. 4, pages: 425-435.

Guttman A., 1984. *R-trees: a dynamic index structure for spatial searching*, In Proceedings of the 1984 ACM SIGMOD international conference on Management of data, Boston, Massachusetts, pages: 47-57

Haas Z., Halpern J. & Li L., 2002. *Gossip based ad hoc routing*, In Proc. of IEEE INFOCOM, New York, pages: 1707-1716

Haghighi P.D., Zaslavsky A., Krishnaswamy S., Gaber M.M. & Loke S., 2009. *Context-aware adaptive data stream mining*. Intell. Data Anal., vol. 13, no. 3, pages: 423-434.

Hähner J., Becker C. & Marrón P.J., 2004. *Consistent Context Management in Mobile Ad Hoc Networks*, In Proceedings Workshop "Get Connected to the Mobile World - Data Management in Mobile Environments", Informatik 2004, Ulm, Germany, pages: 308-313

Handy M., 2004. *DCP: A New Data Collection Protocol for Bluetooth-Based Sensor Networks*, In Euromicro Symposium on Digital Systems Design, pages: 566-573

He T., Krishnamurthy S., Stankovic J.A., Abdelzaher T., Luo L., Stoleru R., Yan T., Gu L., Hui J. & Krogh B., 2004. *Energy-efficient surveillance system using wireless sensor networks*, In Proceedings of the 2nd international conference on Mobile systems, applications, and services, Boston, MA, USA, pages: 270-283

Hedetniemi S.M., Hedetniemi S.T. & Liestman A.L., 1988. *A survey of gossiping and broadcasting in communication networks*. Networks, vol. 18, no. 4, pages: 319-349.

Heinzelman W.B., 2000. *Application-specific protocol architectures for wireless networks.* Ph.D. dissertstion, Mass. Inst. Technology.Cambridge

Heinzelman W.B., Chandrakasan A.P. & Balakrishnan H., 2002. *An Application-Specific Protocol Architecture for Wireless Microsensor Networks*. IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, vol. 1, no. 4, pages: 660 - 670.

Heinzelman W.R., Chandrakasan A.P. & Balakrishnan H., 2000. *Energy-Efficient Communication Protocol for Wireless Sensor Networks*, In Proceedings of the 33th Hawaii International Conference on System Sciences, pages: 10

Heinzelman W.R., Kulik J. & Balakrishnan H., 1999. *Adaptive protocols for information dissemination in wireless sensor networks*, In Proceedings of the 5th annual

ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, United States, pages: 174-185

Henkel D., Dixon C., Elston J. & Brown T.X., 2006. *A reliable sensor data collection network using unmanned aircraft*, In Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality, Florence, Italy, pages: 125-127

Howard A., Matari´C M.J. & Sukhatme G.S., 2002. *An Incremental Self-Deployment Algorithm for Mobile Sensor Networks*. Autonomous Robots, Special Issue on Intelligent Embedded Systems*,* vol. 13, no.**,** pages: 113--126.

Howitt I. & Gutierrez J.A., 2003. *IEEE802.15.4 low rate-wireless personal area network coexistence issues*, In IEEE Wireless Communications and Networking Conference (WCNC), pages: 1481–1486

Huang G.T., 2003. *Casting the Wireless Sensor Net*, MIT's Magazine of Innovation, pages: 51–56

Huang J.H., Amjad S. & Mishra S., 2005. *CenWits: A sensor-based loosely coupled search and rescue system using witnesses*, In Proceedings of the Third International Conference on Embedded Networked Sensor Systems (Sensys), San Diego, CA, pages: 180 - 191

Hull B., Bychkovsky V., Zhang Y., Chen K., Goraczko M., Miu A., Shih E., Balakrishnan H. & Madden S., 2006. *CarTel: a distributed mobile sensor computing system*, In Proceedings of the 4th international conference on Embedded networked sensor systems, Boulder, Colorado, USA, pages: 125-138

Iarsystems, 2008. *IAR Embedded Workbench* [Online]. Available: http://www.iar.com/website1/1.0.1.0/50/1/ [Accessed 01/08/2008].

Ieee 2003. Standard 802.15.3, Wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wireless person area networks (WPANs).

Intanagonwiwat C., Govindan R. & Estrin D., 2000. *Directed diffusion: a scalable and robust communication paradigm for sensor networks*, In Proceedings of ACM MobiCom, Boston, MA, pages: 56-67

Iroad, 2010. *iRoad* [Online]. Available: http://www.iroad.se/ [Accessed 27/03/2010].

Itu, 2008. *Corporate Annual Report* [Online]. Available: http://www.itu.int/dms_pub/itu-s/opb/conf/S-CONF-AREP-2008-E06-PDF-E.pdf [Accessed 25/10/2009].

Jain S., Shah R., Brunette W., Borriello G. & Roy S., 2006. *Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks*. ACM/Springer Mobile Networks and Applications*,* vol. 11, no. 3**,** pages: 327-339.

Jayaraman P.P., Zaslavsky A. & Delsing J., 2007. *Sensor Data Collection Using Heterogeneous Mobile Devices*, In in Proceedings of the IEEE International Conference on Pervasive Services, Istanbul, Turkey, pages: 161-164

Jayaraman P.P., Zaslavsky A. & Delsing J., 2008a. *Cost Efficient Data Collection of Sensory Originated Data using Context-Aware Mobile Devices*, In Ninth International Conference on Mobile Data Management Workshops. MDMW 2008. , pages: 190-200

Jayaraman P.P., Zaslavsky A. & Delsing J., 2008b. *Coverage Area Computation on the Run for Efficient Sensor Data Collection*, In New Technologies, Mobility and Security, 2008. NTMS '08., Tangier, Morocco, pages: 1-4

Jayaraman P.P., Zaslavsky A. & Delsing J., 2008c. *Smart Sensing and Sensor Data Collection on the Move for Modelling Intelligent Environments*, In Proceedings of the 8th international conference, NEW2AN and 1st Russian Conference on Smart Spaces, ruSMART on Next Generation Teletraffic and Wired/Wireless Advanced Networking, St. Petersburg, Russia, pages: 306-317

Jayaraman P.P., Zaslavsky A. & Delsing J., 2009a. *Dynamic situation modeling and reasoning under uncertainty*, In Proceedings of the 2009 international conference on Pervasive services, London, United Kingdom, pages: 113-122

Jayaraman P.P., Zaslavsky A. & Delsing J., 2009b. *On-the-Fly Situation Composition within Smart Spaces*, In Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces, St. Petersburg, Russia, pages: 52-65

Jayaraman P.P., Zaslavsky A. & Delsing J., 2010a. *Cost-Efficient Data Collection Approach Using K-Nearest Neighbors in a 3D Sensor Network*, In IEEE International Conference on Mobile Data Management, Kansas City, Missouri, pages: 183-188

Jayaraman P.P., Zaslavsky A. & Delsing J., 2010b. *Intelligent Mobile Data Mules for Cost-Efficient Sensor Data Collection*. International Journal of Next-Generation Computing, vol. 1, no. 1**,** pages: 73-90.

Jayaraman P.P., Zaslavsky A. & Delsing J., 2010c. *Intelligent Processing of K-Nearest Neighbours Queries using Mobile Data Collectors in a Location Aware 3D Wireless Sensor Network*, In The Twenty Third International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2010), Cordoba, Spain, pages: 260-270

Jea D.D., Somasundara A.A. & Srivastava M.B., 2005. *Multiple controlled mobile elements (data mules) for data collection in sensor networks*, In International Conference on Distributed Computing in Sensor Systems (DCOSS), pages: 244--257

Jenkins A., Henkel D. & Brown T., 2007. *Sensor Data Collection through Unmanned Aircraft Gateways*, In Proc. of the AIAA Infotech@Aerospace 2007 Conference and Exhibit, California, pages:

Jian Z., Yinong L., Yang J. & Ping Z., 2007. *A Context-Aware Infrastructure with Reasoning Mechanism and Aggregating Mechanism for Pervasive Computing Application*, In IEEE 65th Vehicular Technology Conference, 2007. VTC2007-Spring., pages: 257-261

Juang P., Oki H., Wang Y., Martonosi M., Peh L.S. & Rubenstein D., 2002. *Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet*. SIGARCH Comput. Archit. News*,* vol. 30, no. 5**,** pages: 96-107.

Kahn J.M., Katz R.H. & Pister K.S.J., 1999. *Mobile Networking for Smart Dust*, In ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, pages:

Kansal A., Hsu J., Zahedi S. & Srivastava M.B., 2007. *Power management in energy harvesting sensor networks*. ACM Trans actions on Embed. Comput. Syst.*,* vol. 6, no. 4**,** pages: 32.

Kansal A., Somasundara A.A., Jea D.D., Srivastava M.B. & Estrin D., 2004. *Intelligent Fluid Infrastructure for Embedded Networks*, In ACM MobiSYS'04, Boston, Massachusetts, USA, pages: 111-124

Kansal A. & Srivastava M.B., 2003. *An environmental energy harvesting framework for sensor networks*, In Proceedings of the International Symposiumon LowPower Electronics andDesign, pages: 481–486

Khedr M. & Karmouch A., 2005. *ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications*. Journal of Network & Computer Applications*,* vol. 28, no. 1**,** pages: 19-44.

Ko K.E. & Sim K.B., 2008. *Development of context aware system based on Bayesian network driven context reasoning method and ontology context modeling*, In International Conference on Control, Automation and Systems, 2008. ICCAS 2008., pages: 2309-2313

Krishnamachari B., Estrin D. & Wicker S.B., 2002. *The Impact of Data Aggregation in Wireless Sensor Networks*, In Proceedings of the 22nd International Conference on Distributed Computing Systems, pages: 575-578

Kulik J., Heinzelman W.R. & Balakrishnan H., 2002. *Negotiation-based protocols for disseminating information in wireless sensor networks*. Wireless Networks*,* vol. 8, no. 2/3**,** pages: 169-185.

Kulik L., Tanin E. & Umer M., 2008. *Efficient Data Collection and Selective Queries in Sensor Networks*. *GeoSensor Networks: Second International Conference, GSN 2006, Boston, MA, USA, October 1-3, 2006.* Springer-Verlag. pages: 25-44

Kumar V., 2003. *Sensor: the atomic computing particle*. SIGMOD Rec.*,* vol. 32, no. 4**,** pages: 16-21.

Kumar V., 2005. *Data in Your Space*. *In:* GHOSH, R. & MOHANTY, H. (eds.) *Distributed Computing and Internet Technology.* Springer Berlin / Heidelberg. pages: 39-66

Lamarca A., Koizumi D., Lease M., Sigurdsson S., Borriello G., Brunette W., Sikorski K. & Fox D., 2002a. *Making Sensor Networks Practical with Robots*, In First International Conference on Pervasive Computing, pages: 152-166

Lamarca A., Sigurdsson S., Brunette W., Koizumi D., Lease M., Sigurdsson S.B., Sikorski K., Fox D. & Borriello G., 2002b. *PlantCare: An Investigation in Practical Ubiquitous Systems*, In Proceedings of the 4th international conference on Ubiquitous Computing, Goteborg, Sweden, pages: 316 - 332

Leopold M., Dydensborg M.B. & Bonnet P., 2003. *Bluetooth and sensor networks: a reality check*, In Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, pages: 103-113

Lewis F.L., 2004. *Wireless Sensor Networks*. *In:* COOK, D. J. & DAS, S. K. (eds.) *Smart Environments.* pages: 11-46

Li D., Wong K.D., Hu Y.H. & Sayeed A.M., 2002. *Detection, classification, and tracking of targets*. IEEE Signal Processing Mag*,* vol. 19, no.**,** pages: 17-29.

Lin C., He Y.X. & Xiong N., 2006. *An Energy-Efficient Dynamic Power Management in Wireless Sensor Networks*, In The Fifth International Symposium on Parallel and Distributed Computing, 2006. ISPDC '06. , pages: 148-154

Lindsey S. & Raghavendra C.S., 2005. *PEGASIS: Power-Efficient Gathering in Sensor Information Networks* [Online]. Available: http://ceng.usc.edu/~raghu/pegasisrev.pdf [Accessed 10/03/2007].

Loke S., 2006. *Context-Aware Pervasive Systems*, Auerbach Publications.

Lundberg M., Eliasson J., Allan J., Johansson J. & Lindgren P., 2005. *Power characterization of a bluetooth-equipped sensor node*, In Proceedings of the First REALWSN 2005 Workshop on Real-World Wireless Sensor Networks, Stockholm pages:

Luo J. & Hubaux J.P., 2005. *Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks*, In the 24th IEEE INFOCOM, Miami, USA, pages: 1735-1746

M16cflasher, 2008. *M16C-Flasher NON-PROFIT-Version* [Online]. Available: http://m16c.cco-ev.de/M16C-Flasher.4.0.html [Accessed 01/06/2008].

Madden S., Franklin M.J., Hellerstein J.M. & Hong W., 2002. *TAG: a Tiny AGgregation service for ad-hoc sensor networks*. SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pages: 131-146.

Madden S., Franklin M.J., Hellerstein J.M. & Hong W., 2003. *The design of an acquisitional query processor for sensor networks*, In Proceedings of SIGMOD, pages: 491–502

Madden S.R., Franklin M.J., Hellerstein J.M. & Hong W., 2005. *TinyDB: an acquisitional query processing system for sensor networks*. ACM Transaction on Database Syst., vol. 30, no. 1, pages: 122-173.

Mainwaring A., Polastre J., Szewczyk R., Culler D. & Anderson J., 2002. *Wireless sensor networks for habitat monitoring*, In ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, GA, pages: 88-97

Mantyjarvi J. & Seppanen T., 2002. *Adapting Applications in Mobile Terminals Using Fuzzy Context Information*, In Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, pages: 95 - 107

Margi C.B. & Obraczka K., 2006. *GloMoSim Energy Consumption Instrumentation* [Online]. Available: http://users.soe.ucsc.edu/~cintia/energy-glomo.html [Accessed 01/06/2010].

Mcmickell M.B., Goodwine B. & Montestruque L.A., 2003. *MICAbot: A Robotic Platform for Large-Scale Distributed Robotics*, In Proc. of the 2003 IEEE, Intl. Conference on Robotics and Automation, pages: 1600-1605

Mendel J.M., 1995. *Fuzzy logic systems for engineering: a tutorial*. New York, NY, ETATS-UNIS: Institute of Electrical and Electronics Engineers. pages: 345-377

Mergen G., Zhao Q. & Tong L., 2006. *Sensor Networks With Mobile Access: Energy and Capacity Considerations*. IEEE TRANSACTIONS ON COMMUNICATIONS, vol. 54, no. 11, pages: 1896 - 1896

Meyer S. & Rakotonirainy A., 2003. *A Survey of Research on Context-Aware Homes*, In Proceedings of the Australasian information security workshop conference on ACSW, Adelaide, Australia, pages: 159-168

Microsoft.Net, 2010. *Framework Developer Center* [Online]. Available: http://msdn.microsoft.com/en-us/netframework/default.aspx [Accessed 01/06/2010].

Mitchell M., 1998. *An introduction to genetic algorithms*, MIT Press Paperback Edition.

Moteiv, 2006. *Tmote Sky - Ultra low power IEEE 802.15.4 compliant wireless sensor module* [Online]. Available: http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf [Accessed 29/03/2010].

Mouratidis K., Papadias D. & Hadjieleftheriou M., 2005a. *Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring*, In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, Baltimore, Maryland, pages: 634-645

Mouratidis K., Papadias D. & Tao Y., 2005b. *A Threshold-Based Algorithm for Continuous Monitoring of k Nearest Neighbors*. IEEE Trans. on Knowl. and Data Eng., vol. 17, no. 11, pages: 1451-1464.

Moussaoui O. & Naïmi M., 2005. *A distributed energy aware routing protocol for wireless sensor networks*, In Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pages: 34-40

Mulmuley K., 1993. *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall.

Nachman L., Kling R., Adler R., Huang J. & Hummel V., 2005. *The Intel Mote platform: a Bluetooth-based sensor network for industrial monitoring*, In Proceedings of the 4th international symposium on Information processing in sensor networks, Los Angeles, California, pages: 437-442

Nath S., Gibbons P.B., Seshan S. & Anderson Z.R., 2004. *Synopsis diffusion for robust aggregation in sensor networks*, In Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, pages: 250-262

Nationalinstruments, 2008. *Product Information: What is NI LabVIEW?* [Online]. Available: http://www.ni.com/labview/ [Accessed 01/07/2008].

Niculescu D. & Nath B., 2003. *Trajectory based forwarding and its applications*, In Proceedings of the 9th annual international conference on Mobile computing and networking, San Diego, CA, USA, pages: 260-272

Nuevo J., 2004. *A Comprehensible GloMoSim Tutorial* [Online]. Available: www.cs.virginia.edu/~jx9n/courses/cs656/glomoman.pdf [Accessed 01/06/2010].

Ohult C., 2006. *lwbt - a lightweight bluetooth stack* [Online]. Available: http://www.csee.ltu.se/˜conny/lwBT [Accessed 15/07/2006].

Orecchia L., Panconesi A., Petrioli C. & Vitaletti A., 2004. *Localized Techniques for Broadcasting in Wireless Sensor Networks*, In Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications, Philadelphia, USA, pages: 41-51

Padovitz A., 2006. *Context Management and Reasoning about Situations in Pervasive Computing.* PhD, Monash University, Australia

Padovitz A., Loke S.W. & Zaslavsky A., 2004. *Towards a Theory of Context Spaces*, In IEEE International Conference on Pervasive Computing and Communications Workshops, pages: 38-42

Padovitz A., Loke S.W., Zaslavsky A., Burg B. & Bartolini C., 2005. *An Approach to Data Fusion for Context Awareness. In:* DEY, A., KOKINOV, B., LEAKE, D. & TURNER, R. (eds.) *Modeling and Using Context.* Springer Berlin / Heidelberg. pages: 353-367

Paredis C., Khosla P.K., Grabowski R. & Navarro-Serment L.E., 2002. *Millibots: The Development of a Framework and Algorithms for a Distributed Heterogeneous Robot Team*. IEEE Robotics & Automation Magazine, vol. 9, no. 4**,** pages: 31-40.

Park S. & Srivastava M.B., 2002. *Dynamic battery state aware approaches for improving battery utilization*, In Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems, Grenoble, France, pages: 225-231

Pascoe J., Ryan N.S. & Morse D.R., 1999. *Issues in developing context-aware computing*. In H-W.Gellersen (eds), Handheld and Ubiquitous Computing, vol. 1707 in LNCS, no.**,** pages: 208-221.

Passos R.M., Coelho C.J.N., Loureiro A.A.F. & Mini R.A.F., 2005. *Dynamic Power Management in Wireless Sensor Networks: An Application-Driven Approach*, In Second Annual Conference on Wireless On-demand Network Systems and Services. WONS 2005. , pages: 109-118

Pinto J., 2002. *The 3 technology laws* [Online]. Automation.com. Available: http://www.jimpinto.com/writings/techlaws.html [Accessed 21/03/2010].

Polastre J., Szewczyk R. & Culler D., 2005. *Telos: enabling ultra-low power wireless research*, In Fourth International Symposium on Information Processing in Sensor Networks. IPSN 2005., pages: 364-369

Pottie G.J. & Kaiser W.J., 2000. *Wireless integrated network sensors*. Commun. ACM, vol. 43, no. 5**,** pages: 51-58.

Rabaey J.M., Ammer M.J., Da Silva J.L., Jr., Patel D. & Roundy S., 2000. *PicoRadio supports ad hoc ultra-low power wireless networking*. Computer, vol. 33, no. 7**,** pages: 42-48.

Raghunathan V., Schurgers C., Park S. & Srivastava M.B., 2002. *Energy-Aware Wireless Microsensor Networks*. IEEE Signal Processing Magazine, vol. no.**,** pages: 40-50.

Rahimi M., Shah H., Sukhatme G., Heidemann J. & Estrin D., 2003. *Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network*, In Proceedings of the IEEE International Conference on Robotics and Automation, pages: 19-24

Ramanathan N., Balzano L., Estrin D., Hansen M., Harmon T., Jay J., Kaiser W. & Sukhatme G., 2005. *Designing Wireless Sensor Networks as a Shared Resource for Sustainable Development*, In First International Conference on Information and Communication Technologies and Development, pages: 256-265

Ranganathan A., Al-Muhtadi J. & Campbell R.H., 2004. *Reasoning about uncertain contexts in pervasive computing environments*. Pervasive Computing, IEEE, vol. 3, no. 2**,** pages: 62-70.

Ren B., Ma J. & Chen C., 2006. *The Hybrid Mobile Wireless Sensor Networks for Data Gathering*, In Proceedings of the international conference on Wireless communications and mobile computing, Vancouver, British Columbia, Canada, pages: 1085 - 1090

Renesas, 2008. *Microcontroller m16c/62m* [Online]. Available: http://www.renesas.eu/products/mpumcu/m16c/m16c60/m16c62a/m16c62a_root.jsp [Accessed 08/03/2008].

Roundy S., Wright P.K. & Rabaey J.M., 2004. *Energy Scavenging for Wireless Sensor Networks: With Special Focus on Vibrations*, Kluwer Academic Publishers.

Roussopoulos N., Kelley S. & Vincent F., 1995. *Nearest neighbor queries*, In Proceedings of the 1995 ACM SIGMOD international conference on Management of data, San Jose, California, United States, pages: 71-79

Rytter A., 2003. *Vibration based inspection of civil engineering structures.* ph.d. thesis, Aalborg univ.Denmark

Satyanarayanan M., 2000. *Caching Trust Rather Than Content*. Operating System Review, vol. 34, no. 4, pages: 245 - 246.

Satyanarayanan M., 2002. *Pervasive computing: vision and challenges*. Personal Communications, vol. 8, no. 4, pages: 10-17.

Schindelhauer C., 2006. *Mobility in Wireless Networks*. Invited Talk for SOFSEM, vol. no., pages.

Sen S. & Kumar A., 2010. *Notes in Computational Geometry Voronoi Diagrams* [Online]. Available: http://www.cse.iitd.ernet.in/~ssen/cs852/scribe/Voronoi-ScribeNotes/voronoi.pdf [Accessed 05/05/2010].

Shah R.C., Roy S., Jain S. & Brunette W., 2003. *Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks*, In Proc. IEEE Int'l Workshop on Sensor Network Protocols and Applications (SNPA 2003), pages: 30-41

Shen C.C., Srisathapornphat C. & Jaikaeo C., 2001. *Sensor information networking architecture and applications*. Personal Communications, IEEE, vol. 8, no. 4, pages: 52-59.

Shnayder V., Hempstead M., Chen B.-R., Allen G.W. & Welsh M., 2004. *Simulating the power consumption of large-scale sensor network applications*, In Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, pages: 188-200

Sinha A. & Chandrakasan A., 2001. *Dynamic power management in wireless sensor networks*. Design & Test of Computers, IEEE, vol. 18, no. 2, pages: 62-74.

Small T. & Haas Z.J., 2003. *The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)*, In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking \&amp; computing, Annapolis, Maryland, USA, pages: 233-244

Soheili A., Kalogeraki V. & Gunopulos D., 2005. *Spatial queries in sensor networks*, In Proceedings of the 13th annual ACM international workshop on Geographic information systems, Bremen, Germany, pages: 61-70

Solis I. & Obraczka K., 2006. *In network aggregation tradeoffs for data collection in wireless sensor networks*. Int. J. Sen. Netw., vol. 1, no. 3/4, pages: 200-212.

Somasundara A.A., Kansal A., Jea D.D., Estrin D. & Srivastava M.B., 2006. *Controllably Mobile Infrastructure for Low Energy Embedded Networks*. IEEE Transactions on Mobile Computing, vol. 5, no. 8, pages: 958-973.

Somasundara A.A., Ramamoorthy A. & Srivastava M.B., 2004. *Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines*, In Proceedings of the 25th IEEE International Real-Time Systems Symposium, pages: 296-305

Song L. & Hatzinakos D., 2005. *Dense wireless sensor networks with mobile sinks*, In IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). pages: iii/677-iii/680 Vol. 3

Srivastava M.B., 2002. *Sensor Node Platforms & Energy Issues* [Online]. Available: http://nesl.ee.ucla.edu/tutorials/mobicom02/slides/Mobicom-Tutorial-2-MS.pdf [Accessed 12/10/2007].

Srivastava M.B., Muntz R. & Potkonjak M., 2001. *Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments*, In Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy, pages: 132-138

Steere D.C., Baptista A., Mcnamee D., Pu C. & Walpole J., 2000. *Research challenges in environmental observation and forecasting systems*, In Proceedings of the sixth annual international conference on Mobile computing and networking, Boston, Massachusetts, United States, pages: 292-299

Sunmicrosystems, 2007. *Sun™ Small Programmable Object Technology (Sun SPOT) Theory of Operation* [Online]. Available: http://www.sunspotworld.com/docs/Purple/SunSPOT-TheoryOfOperation.pdf [Accessed 29/03/2010].

Symbianfoundation, 2010. Available: http://www.symbian.org/ [Accessed 01/06/2010].

Tanenbaum A.S., 2002. *Computer networks,* Upper Saddle River, New Jersey, Prentice Hall.

Texasinstruments, 2007. *CC2420 - Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee™ Ready RF Transceiver,* [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc2420.html [Accessed 28/03/2010].

Texasinstruments, 2010. *CC1000-Single Chip Very Low Power RF Transceiver* [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc1000.pdf [Accessed 21/03/2010].

Topicmap, 2010. *Whats happening in TopicMap world?* [Online]. Available: www.topicmap.com [Accessed 01/05/2010].

Tseng Y.C., Ni S.Y., Chen Y.S. & Sheu J.P., 2002. *The Broadcast Storm Problem in a Mobile Ad Hoc Network*. Wireless Networks, vol. 8, no. 2-3, pages: 153-167.

Ucla, 2009. *Parallel Computing Laboratory, PARSEC* [Online]. Available: http://pcl.cs.ucla.edu/projects/parsec [Accessed 01/08/2009].

Utms, 2010. *Universal Traffic Management Society of Japan* [Online]. Available: http://www.utms.or.jp/english/index.html [Accessed 01/08/2010].

Vasilescu I., Kotay K., Rus D., Dunbabin M. & Corke P., 2005. *Data collection, storage, retrieval with an underwater sensor network*, In Proceedings of the Third International Conference on Embedded Networked Sensor Systems (Sensys), San Diego, CA, pages: 154-165

Venkitasubramaniam P., Adireddy S. & Tong L., 2004. *Sensor Networks with Mobile Access: Optimal Random Access and Coding*. IEEE Journal on Selected Areas in Communications, vol. 22, no. 6, pages: 1058- 1068.

Vieira M., Coelho C.N., Jr., Da Silva D.C., Jr. & Da Mata J.M., 2003. *Survey on wireless sensor network devices*, In IEEE Conference on Emerging Technologies and Factory Automation, Belo Horizonte, Brazil, pages: 537 - 544

Virtamo J., 2010. *Poisson process* [Online]. Available: http://www.netlab.tkk.fi/opetus/s38143/luennot/E_poisson.pdf [Accessed 25/04/2010].

Wang A. & Chandrakasan A., 2001. *Energy efficient system partitioning for distributed wireless sensor networks*, In Proceedings. (ICASSP '01). 2001 IEEE International

Conference on Acoustics, Speech, and Signal Processing, 2001. , pages: 905-908 vol.2

Wang Z.M., Basagni S., Melachrinoudis E. & Petrioli C., 2005. *Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime*, In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pages: 287.1

Want R., Hopper A., Falcão V. & Gibbons J., 1992. *The active badge location system*. ACM Trans. Inf. Syst., vol. 10, no. 1, pages: 91-102.

Warneke B.A. & Pister K.S.J., 2002. *MEMS for distributed wireless sensor networks*, In 9th International Conference on Electronics, Circuits and Systems, 2002. , pages: 291-294 vol.1

Weiser M., 1999. *The computer for the 21st century*. SIGMOBILE Mob. Comput. Commun. Rev., vol. 3, no. 3, pages: 3-11.

Werner-Allen G., Lorincz K., Welsh M., Marcillo O., Johnson J., Ruiz M. & Lees J., 2006. *Deploying a wireless sensor network on an active volcano*, In IEEE Internet Computing 10, pages: 18–25

Wikipedia, 2010. *Mark Weiser* [Online]. Available: http://en.wikipedia.org/wiki/Mark_Weiser [Accessed 01/06/2010].

Williams B. & Camp T., 2002. *Comparison of broadcasting techniques for mobile ad hoc networks*, In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking \&amp; computing, Lausanne, Switzerland, pages: 194-205

Winter J. & Lee W.C., 2004. *KPT: a dynamic KNN query processing algorithm for location-aware sensor networks*, In Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004, Toronto, Canada, pages: 119-124

Winter J., Xu Y. & Lee W.C., 2005. *Energy Efficient Processing of K Nearest Neighbor Queries in Location-aware Sensor Networks*, In Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pages: 281-292

Winterfeld D. & Edwards W., 1986. *Decision Analysis and Behavioural Research,* Cambridge.

Wodajo F., 2010. *Bluetooth 4.0 is admitted to the hospital – Potential to revolutionize health care devices* [Online]. Available: http://www.imedicalapps.com/2010/04/bluetooth-admitted-to-hospital-healthcare-4/ [Accessed 07/05/2010].

Wu H., Siegel M. & Ablay S., 2003. *Sensor fusion using Dempster-Shafer theory II: static weighting and Kalman filter-like dynamic weighting*, In Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference, 2003. IMTC '03., pages: 907-912 vol.2

Wu H., Siegel M., Stiefelhagen R. & Jieyang, 2002. *Sensor Fusion Using Dempster-Shafer Theory*, In Proceedings of IEEE Instrumentation and Measurement Technology Conference, Anchorage, USA, pages: 7-12

Wu S.H., Chuang K.T., Chen C.M. & Chen M.S., 2007. *DIKNN: An Itinerary-based KNN Query Processing Algorithm for Mobile Sensor Networks*, In IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007, pages: 456-465

Xu N., 2009. *A Survey of Sensor Network Applications* [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.9647&rep=rep1&type=pdf [Accessed 10/11/2009].

Xu N., Rangwala S., Chintalapudi K.K., Ganesan D., Broad A., Govindan R. & Estrin D., 2004. *A wireless sensor network For structural monitoring*, In Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, pages: 13-24

Xu Y., Lee W.C., Xu J. & Mitchell G., 2006. *ProcessingWindow Queries in Wireless Sensor Networks*, In Proceedings of the 22nd International Conference on Data Engineering, pages: 70

Yang G.-Z., 2010. *Body Sensor Networks (BSN)* [Online]. Available: http://vip.doc.ic.ac.uk/bsn/m621.html [Accessed 20/02/2010].

Yao Y., Tang X. & Lim E.P., 2006. *In-network processing of nearest neighbor queries for wireless sensor networks*, In International Conference on Database Systems for Advanced Applications, Singapore, pages: 35-49

Yao Y., Tang X. & Lim E.P., 2009. *Localized monitoring of kNN queries in wireless sensor networks*. The VLDB Journal, vol. 18, no. 1, pages: 99-117.

Yap K.K., Srinivasan V. & Motani M., 2005. *MAX: Human-centric search of the physical world*, In Proceedings of the Third International Conference on Embedded Networked Sensor Systems (Sensys), San Diego, CA, pages: 166-179

Yick J., Mukherjee B. & Ghosal D., 2008. *Wireless sensor network survey*. Computer Networks, vol. 52, no. 12, pages: 2292-2330.

Yoshimi B., 2000. *On Sensor Frameworks for Pervasive Systems*, In Workshop on Software Engineering for Wearable and Pervasive Computing (SEWPC) at 22nd International Conference on Software Engineering (ICSE'00), Limerick, Ireland, pages:

Younis M., Youssef M. & Arisha K., 2002. *Energy-Aware Routing in Cluster-Based Sensor Networks*, In Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems, Quebec. Canada, pages: 129-136

Younis O., 2004. *HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks*. IEEE Transactions on Mobile Computing, vol. 3, no., pages: 366-379.

Zhang P., Sadler C.M., Lyon S.A. & Martonosi M., 2004. *Hardware design experiences in ZebraNet*, In Proceedings of the SenSys'04, Baltimore, MD, pages: 227-238

Zigbee, 2007. *ZigBee connects to mobile phones* [Online]. San Ramon, Calif. Available: http://www.automation.com/content/zigbee-connects-to-mobile-phones [Accessed 01/08/2009].

Zigbee, 2009a. *ZigBee Wireless Sensor Applications for Health, Wellness and Fitness* [Online]. Available: http://www.zigbee.org/imwp/download.asp?ContentID=15585 [Accessed 13/07/2009].

Zigbee, 2009b. *ZigBee: wireless control that simply works* [Online]. Available: http://www.zigbee.org [Accessed 10/08/2009].

Appendix A

The detailed class diagram of the implemented classes on PDA has been divided into two pages for easy representation and clarity.

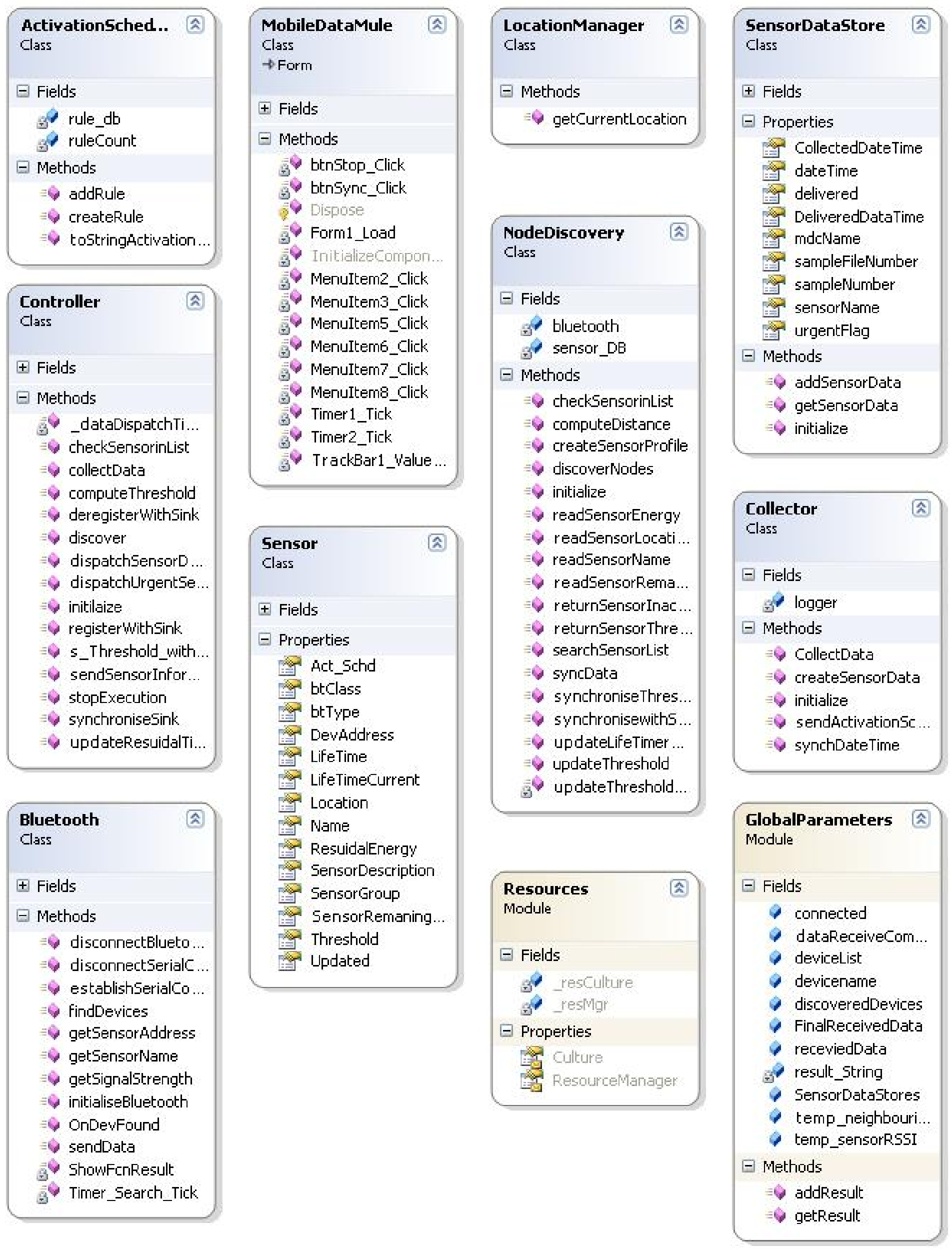


Figure A1: Detailed Class Diagram - Implementation on PDA (Part 1)

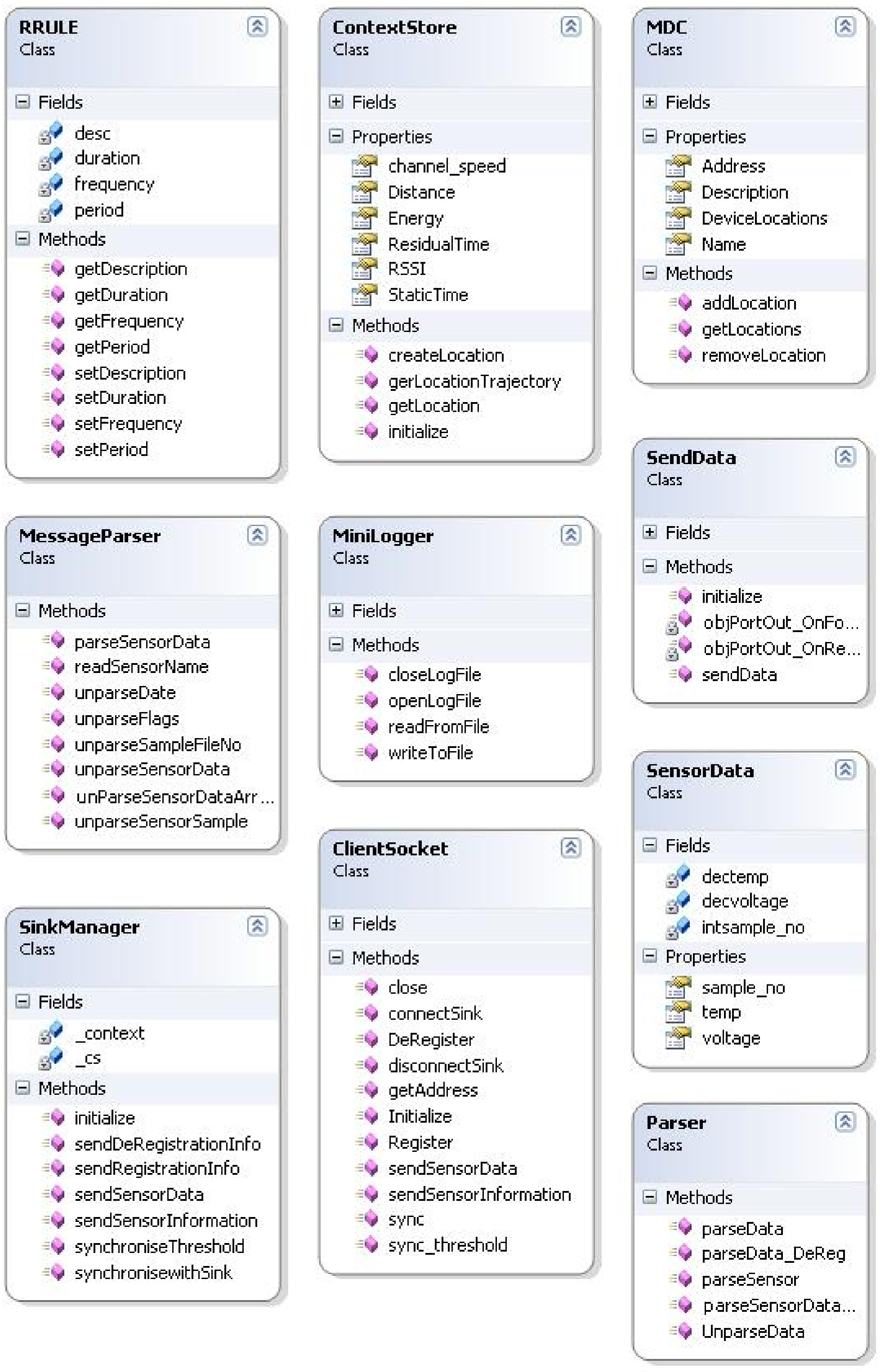
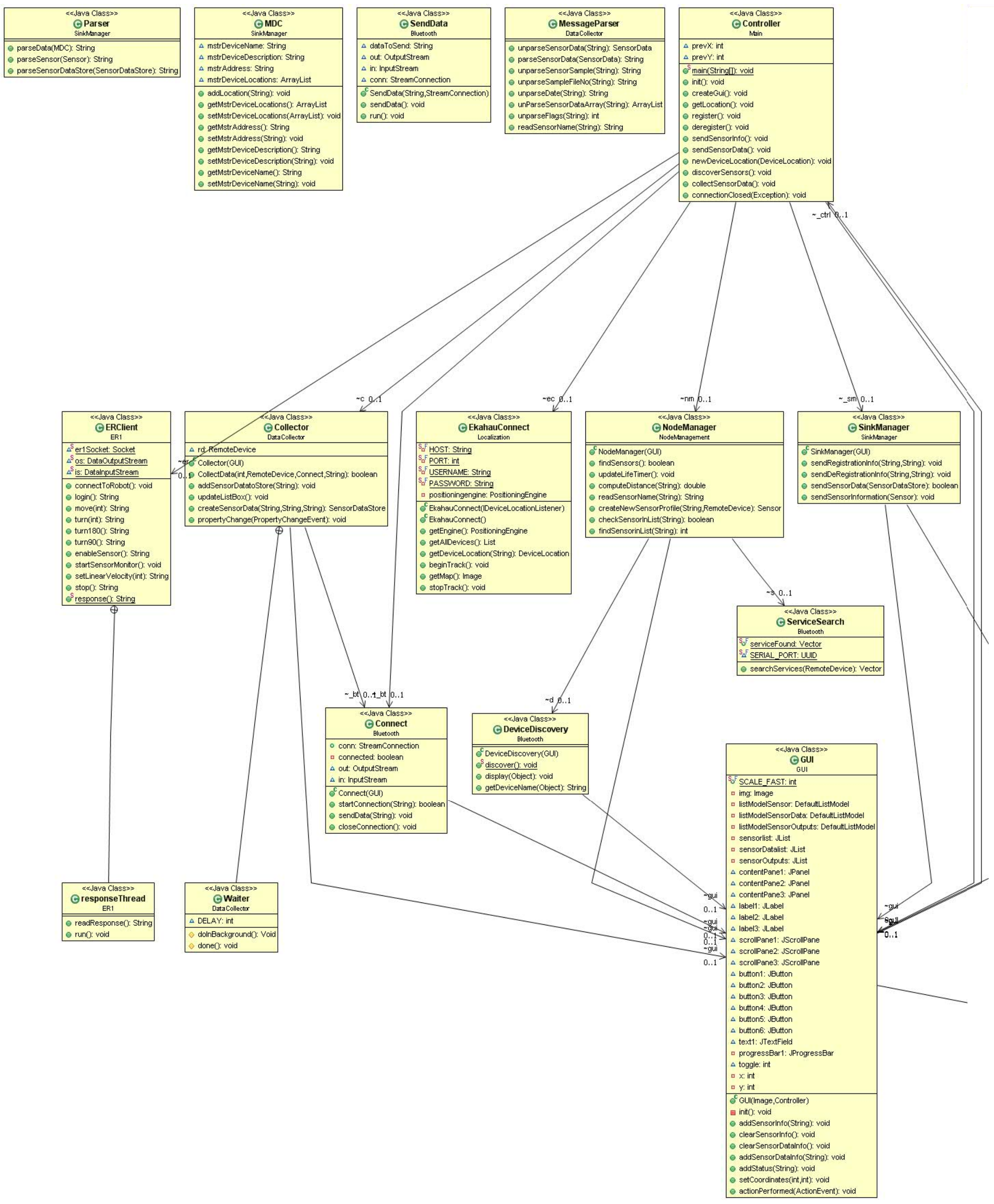
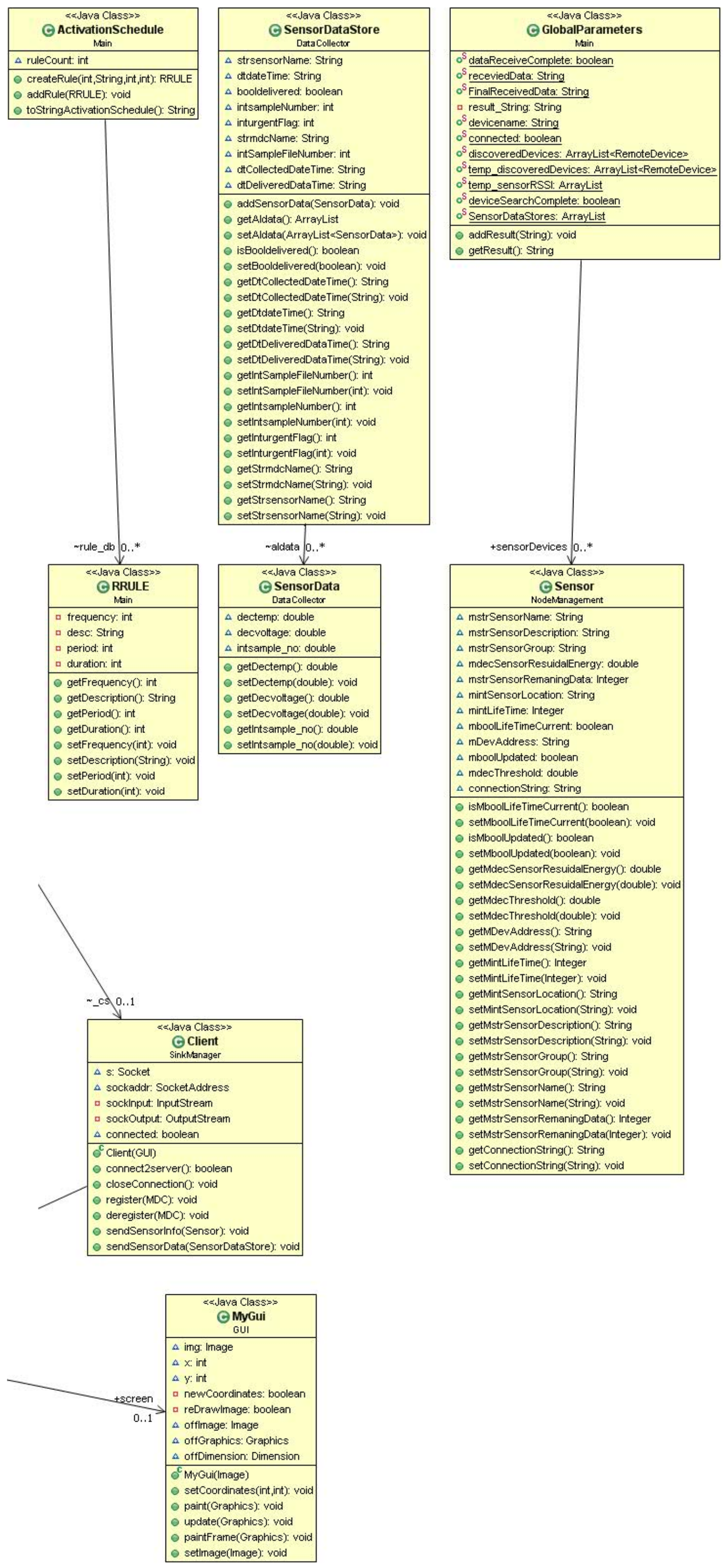


Figure A1: Detailed Class Diagram - Implementation on PDA (Part 2)



<<Java Class>> Parser SinkManager
parseData(MDC): String
parseSensor(Sensor): String
parseSensorDataStore(SensorDataStore): String
<<Java Class>> MDC SinkManager
mstrDeviceName: String
mstrDeviceDescription: String
mstrAddress: String
mstrDeviceLocations: ArrayList
addLocation(String): void
getMstrDeviceLocations(): ArrayList
setMstrDeviceLocations(ArrayList): void
getMstrAddress(): String
setMstrAddress(String): void
getMstrDeviceDescription(): String
setMstrDeviceDescription(String): void
getMstrDeviceName(): String
setMstrDeviceName(String): void
<<Java Class>> SendData Bluetooth
dataToSend: String
out: OutputStream
in: InputStream
conn: StreamConnection
SendData(String,StreamConnection)
sendData(): void
run(): void
<<Java Class>> MessageParser DataCollector
unparseSensorData(String): SensorData
parseSensorData(SensorData): String
unparseSensorSample(String): String
unparseSampleFileNo(String): String
unparseDate(String): String
unParseSensorDataArray(String): ArrayList
unparseFlags(String): int
readSensorName(String): String
<<Java Class>> Controller Main
prevX: int
prevY: int
main(String[]): void
init(): void
createGui(): void
getLocation(): void
register(): void
deregister(): void
sendSensorInfo(): void
sendSensorData(): void
newDeviceLocation(DeviceLocation): void
discoverSensors(): void
collectSensorData(): void
connectionClosed(Exception): void
~_ctrl 0..1
~c 0..1
~ec 0..1
~nm 0..1
~_sm 0..1
<<Java Class>> ERClient ER1
er1Socket: Socket
os: DataOutputStream
is: DataInputStream
connectToRobot(): void
login(): String
move(int): String
turn(int): String
turn180(): String
turn90(): String
enableSensor(): String
startSensorMonitor(): void
setLinearVelocity(int): String
stop(): String
response(): String
~er 0..1
<<Java Class>> Collector DataCollector
rd: RemoteDevice
Collector(GUI)
CollectData(int,RemoteDevice,Connect,String): boolean
addSensorDatatoStore(String): void
updateListBox(): void
createSensorData(String,String,String): SensorDataStore
propertyChange(PropertyChangeEvent): void
<<Java Class>> EkahauConnect Localization
HOST: String
PORT: int
USERNAME: String
PASSWORD: String
positioningengine: PositioningEngine
EkahauConnect(IDeviceLocationListener)
EkahauConnect()
getEngine(): PositioningEngine
getAllDevices(): List
getDeviceLocation(String): DeviceLocation
beginTrack(): void
getMap(): Image
stopTrack(): void
<<Java Class>> NodeManager NodeManagement
NodeManager(GUI)
findSensors(): boolean
updateLifeTimer(): void
computeDistance(String): double
readSensorName(String): String
createNewSensorProfile(String,RemoteDevice): Sensor
checkSensorInList(String): boolean
findSensorinList(String): int
<<Java Class>> SinkManager SinkManager
SinkManager(GUI)
sendRegistrationInfo(String,String): void
sendDeRegistrationInfo(String,String): void
sendSensorData(SensorDataStore): boolean
sendSensorInformation(Sensor): void
~s 0..1
<<Java Class>> ServiceSearch Bluetooth
serviceFound: Vector
SERIAL_PORT: UUID
searchServices(RemoteDevice): Vector
~_bt 0..1
~_bt 0..1
<<Java Class>> Connect Bluetooth
conn: StreamConnection
connected: boolean
out: OutputStream
in: InputStream
Connect(GUI)
startConnection(String): boolean
sendData(String): void
closeConnection(): void
~d 0..1
<<Java Class>> DeviceDiscovery Bluetooth
DeviceDiscovery(GUI)
discover(): void
display(Object): void
getDeviceName(Object): String
<<Java Class>> GUI GUI
SCALE_FAST: int
img: Image
listModelSensor: DefaultListModel
listModelSensorData: DefaultListModel
listModelSensorOutputs: DefaultListModel
sensorlist: JList
sensorDatalist: JList
sensorOutputs: JList
contentPane1: JPanel
contentPane2: JPanel
contentPane3: JPanel
label1: JLabel
label2: JLabel
label3: JLabel
scrollPane1: JScrollPane
scrollPane2: JScrollPane
scrollPane3: JScrollPane
button1: JButton
button2: JButton
button3: JButton
button4: JButton
button5: JButton
button6: JButton
text1: JTextField
progressBar1: JProgressBar
toggle: int
x: int
y: int
GUI(Image,Controller)
init(): void
addSensorInfo(String): void
clearSensorInfo(): void
clearSensorDataInfo(): void
addSensorDataInfo(String): void
addStatus(String): void
setCoordinates(int,int): void
actionPerformed(ActionEvent): void
~gui 0..1
~gui 0..1
~gui 0..1
~gui 0..1
~gui 0..1
~gui 0..1
<<Java Class>> responseThread ER1
readResponse(): String
run(): void
<<Java Class>> Waiter DataCollector
DELAY: int
doInBackground(): Void
done(): void

Figure A2: Detailed Class Diagram - Implementation on Mobile Robot (Part 1)



<<Java Class>>
ActivationSchedule
Main
ruleCount: int
createRule(int,String,int,int): RRULE
addRule(RRULE): void
toStringActivationSchedule(): String
<<Java Class>>
SensorDataStore
DataCollector
strsensorName: String
dtdateTime: String
booldelivered: boolean
intsampleNumber: int
inturgentFlag: int
strmdcName: String
intSampleFileNumber: int
dtCollectedDateTime: String
dtDeliveredDataTime: String
addSensorData(SensorData): void
getAldata(): ArrayList
setAldata(ArrayList<SensorData>): void
isBooldelivered(): boolean
setBooldelivered(boolean): void
getDtCollectedDateTime(): String
setDtCollectedDateTime(String): void
getDtdateTime(): String
setDtdateTime(String): void
getDtDeliveredDataTime(): String
setDtDeliveredDataTime(String): void
getIntSampleFileNumber(): int
setIntSampleFileNumber(int): void
getIntsampleNumber(): int
setIntsampleNumber(int): void
getInturgentFlag(): int
setInturgentFlag(int): void
getStrmdcName(): String
setStrmdcName(String): void
getStrsensorName(): String
setStrsensorName(String): void
<<Java Class>>
GlobalParameters
Main
dataReceiveComplete: boolean
receviedData: String
FinalReceivedData: String
result_String: String
devicename: String
connected: boolean
discoveredDevices: ArrayList<RemoteDevice>
temp_discoveredDevices: ArrayList<RemoteDevice>
temp_sensorRSSI: ArrayList
deviceSearchComplete: boolean
SensorDataStores: ArrayList
addResult(String): void
getResult(): String
~rule_db
0..*
~aldata
0..*
+sensorDevices
0..*
<<Java Class>>
RRULE
Main
frequency: int
desc: String
period: int
duration: int
getFrequency(): int
getDescription(): String
getPeriod(): int
getDuration(): int
setFrequency(int): void
setDescription(String): void
setPeriod(int): void
setDuration(int): void
<<Java Class>>
SensorData
DataCollector
dectemp: double
decvoltage: double
intsample_no: double
getDectemp(): double
setDectemp(double): void
getDecvoltage(): double
setDecvoltage(double): void
getIntsample_no(): double
setIntsample_no(double): void
<<Java Class>>
Sensor
NodeManagement
mstrSensorName: String
mstrSensorDescription: String
mstrSensorGroup: String
mdecSensorResuidalEnergy: double
mstrSensorRemaningData: Integer
mintSensorLocation: String
mintLifeTime: Integer
mboolLifeTimeCurrent: boolean
mDevAddress: String
mboolUpdated: boolean
mdecThreshold: double
connectionString: String
isMboolLifeTimeCurrent(): boolean
setMboolLifeTimeCurrent(boolean): void
isMboolUpdated(): boolean
setMboolUpdated(boolean): void
getMdecSensorResuidalEnergy(): double
setMdecSensorResuidalEnergy(double): void
getMdecThreshold(): double
setMdecThreshold(double): void
getMDevAddress(): String
setMDevAddress(String): void
getMintLifeTime(): Integer
setMintLifeTime(Integer): void
getMintSensorLocation(): String
setMintSensorLocation(String): void
getMstrSensorDescription(): String
setMstrSensorDescription(String): void
getMstrSensorGroup(): String
setMstrSensorGroup(String): void
getMstrSensorName(): String
setMstrSensorName(String): void
getMstrSensorRemaningData(): Integer
setMstrSensorRemaningData(Integer): void
getConnectionString(): String
setConnectionString(String): void
~_cs
0..1
<<Java Class>>
Client
SinkManager
s: Socket
sockaddr: SocketAddress
sockInput: InputStream
sockOutput: OutputStream
connected: boolean
Client(GUI)
connect2server(): boolean
closeConnection(): void
register(MDC): void
deregister(MDC): void
sendSensorInfo(Sensor): void
sendSensorData(SensorDataStore): void
+screen
0..1
<<Java Class>>
MyGui
GUI
img: Image
x: int
y: int
newCoordinates: boolean
reDrawImage: boolean
offImage: Image
offGraphics: Graphics
offDimension: Dimension
MyGui(Image)
setCoordinates(int,int): void
paint(Graphics): void
update(Graphics): void
paintFrame(Graphics): void
setImage(Image): void

Figure A2: Detailed Class Diagram - Implementation on Mobile Robot (Part 2)

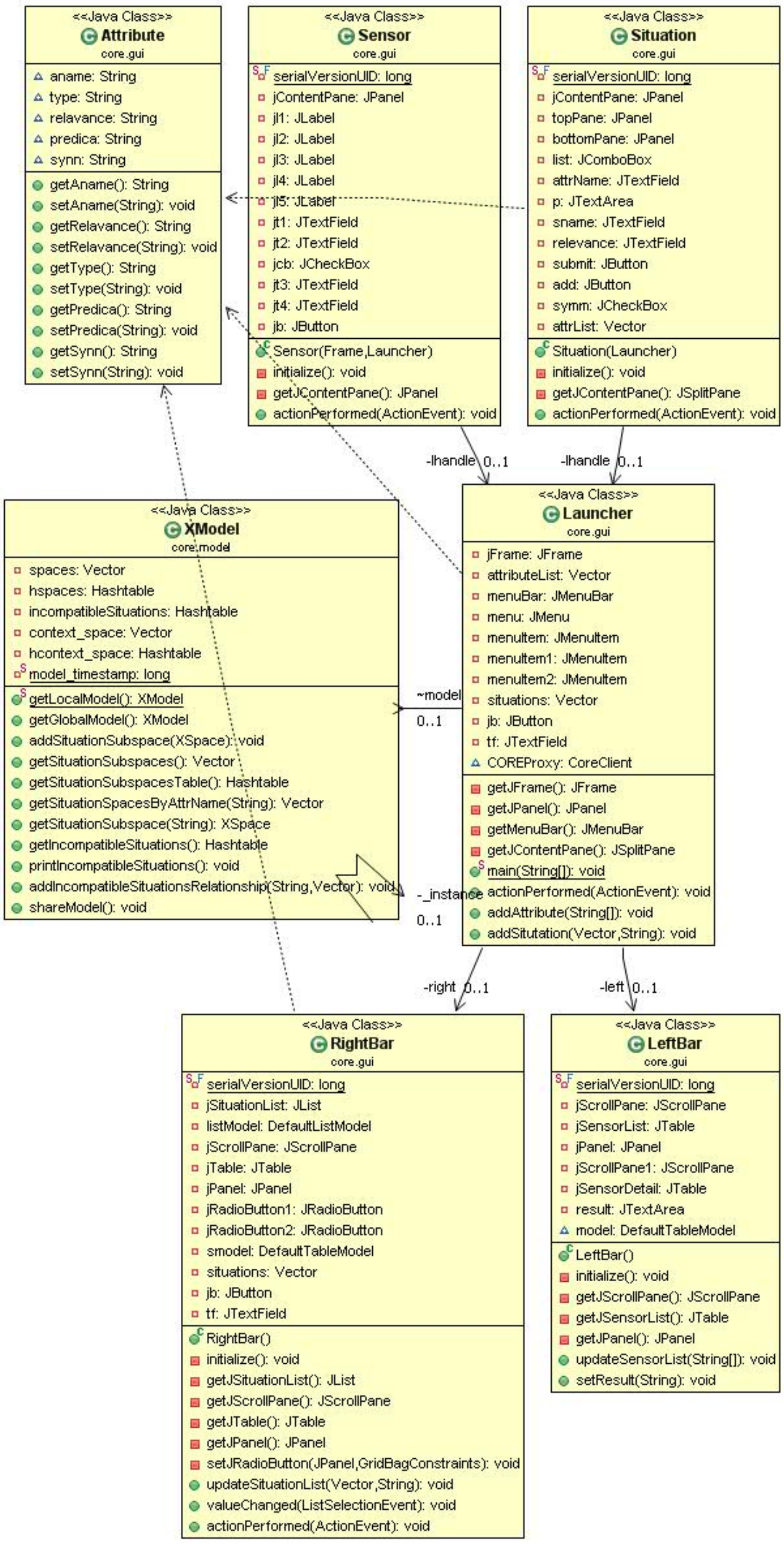
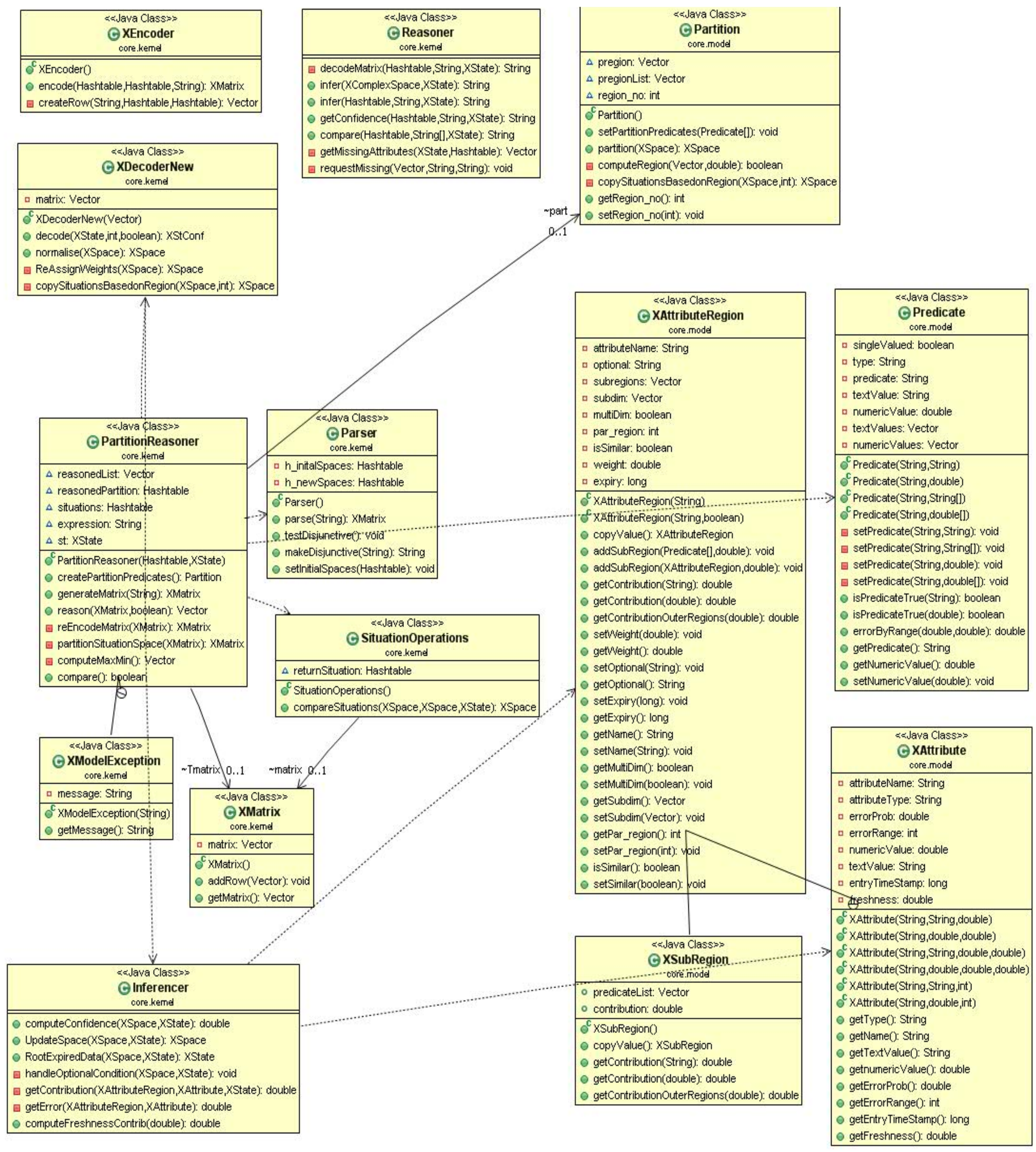


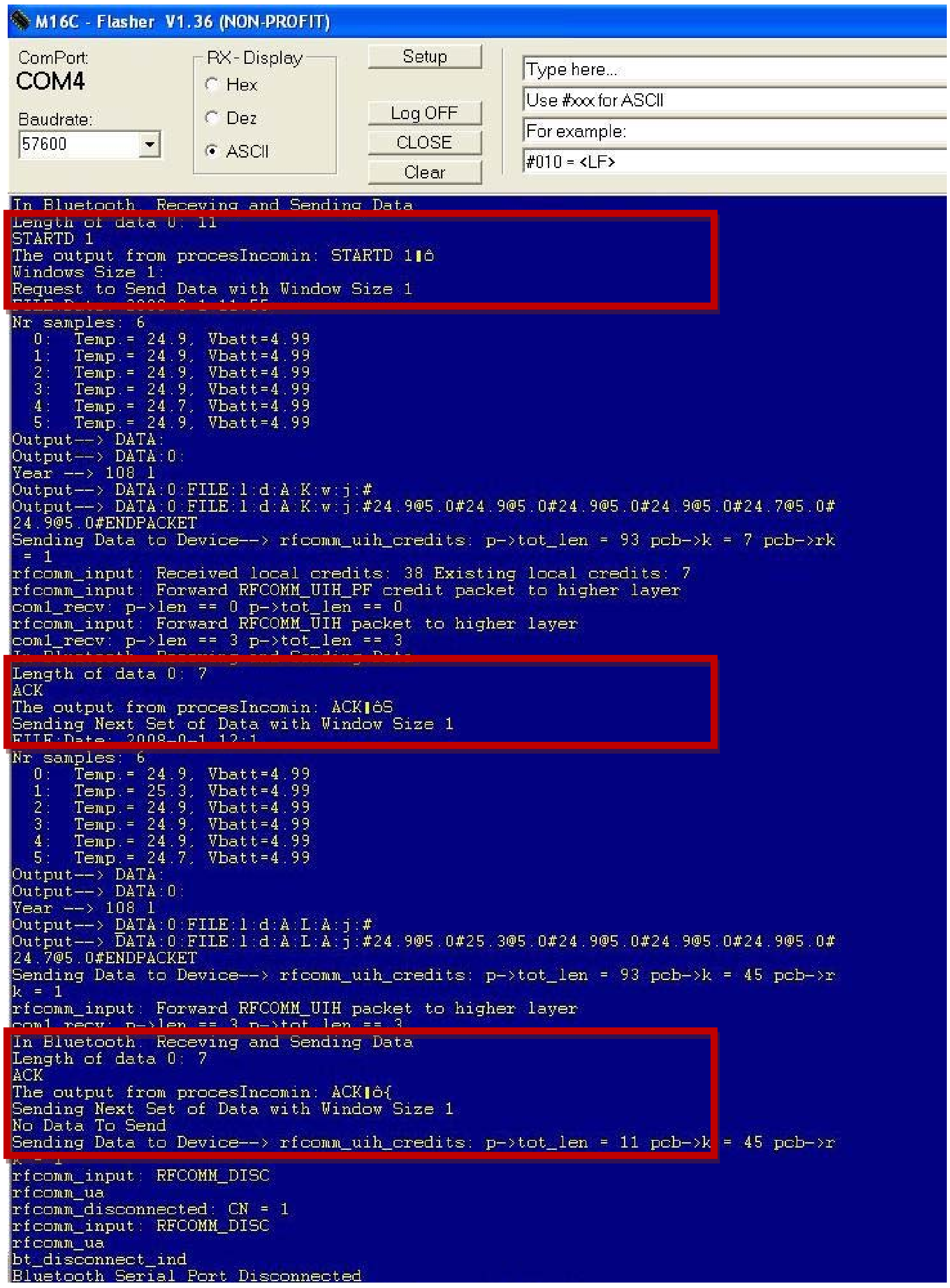
Figure A3: R-CS Implementation - Detailed Class Diagram (Part 1)



<<Java Class>>
XEncoder
core.kernel
XEncoder()
encode(Hashtable,Hashtable,String): XMatrix
createRow(String,Hashtable,Hashtable): Vector
<<Java Class>>
Reasoner
core.kernel
decodeMatrix(Hashtable,String,XState): String
infer(XComplexSpace,XState): String
infer(Hashtable,String,XState): String
getConfidence(Hashtable,String,XState): String
compare(Hashtable,String[],XState): String
getMissingAttributes(XState,Hashtable): Vector
requestMissing(Vector,String,String): void
<<Java Class>>
Partition
core.model
pregion: Vector
pregionList: Vector
region_no: int
Partition()
setPartitionPredicates(Predicate[]): void
partition(XSpace): XSpace
computeRegion(Vector,double): boolean
copySituationsBasedonRegion(XSpace,int): XSpace
getRegion_no(): int
setRegion_no(int): void
<<Java Class>>
XDecoderNew
core.kernel
matrix: Vector
XDecoderNew(Vector)
decode(XState,int,boolean): XStConf
normalise(XSpace): XSpace
ReAssignWeights(XSpace): XSpace
copySituationsBasedonRegion(XSpace,int): XSpace
~part
0..1
<<Java Class>>
XAttributeRegion
core.model
attributeName: String
optional: String
subregions: Vector
subdim: Vector
multiDim: boolean
par_region: int
isSimilar: boolean
weight: double
expiry: long
XAttributeRegion(String)
XAttributeRegion(String,boolean)
copyValue(): XAttributeRegion
addSubRegion(Predicate[],double): void
addSubRegion(XAttributeRegion,double): void
getContribution(String): double
getContribution(double): double
getContributionOuterRegions(double): double
setWeight(double): void
getWeight(): double
setOptional(String): void
getOptional(): String
setExpiry(long): void
getExpiry(): long
getName(): String
setName(String): void
getMultiDim(): boolean
setMultiDim(boolean): void
getSubdim(): Vector
setSubdim(Vector): void
getPar_region(): int
setPar_region(int): void
isSimilar(): boolean
setSimilar(boolean): void
<<Java Class>>
Predicate
core.model
singleValued: boolean
type: String
predicate: String
textValue: String
numericValue: double
textValues: Vector
numericValues: Vector
Predicate(String,String)
Predicate(String,double)
Predicate(String,String[])
Predicate(String,double[])
setPredicate(String,String): void
setPredicate(String,String[]): void
setPredicate(String,double): void
setPredicate(String,double[]): void
isPredicateTrue(String): boolean
isPredicateTrue(double): boolean
errorByRange(double,double): double
getPredicate(): String
getNumericValue(): double
setNumericValue(double): void
<<Java Class>>
PartitionReasoner
core.kernel
reasonedList: Vector
reasonedPartition: Hashtable
situations: Hashtable
expression: String
st: XState
PartitionReasoner(Hashtable,XState)
createPartitionPredicates(): Partition
generateMatrix(String): XMatrix
reason(XMatrix,boolean): Vector
reEncodeMatrix(XMatrix): XMatrix
partitionSituationSpace(XMatrix): XMatrix
computeMaxMin(): Vector
compare(): boolean
<<Java Class>>
Parser
core.kernel
h_initalSpaces: Hashtable
h_newSpaces: Hashtable
Parser()
parse(String): XMatrix
testDisjunctive(): void
makeDisjunctive(String): String
setInitialSpaces(Hashtable): void
<<Java Class>>
SituationOperations
core.kernel
returnSituation: Hashtable
SituationOperations()
compareSituations(XSpace,XSpace,XState): XSpace
<<Java Class>>
XModelException
core.kernel
message: String
XModelException(String)
getMessage(): String
~Tmatrix
0..1
~matrix
0..1
<<Java Class>>
XMatrix
core.kernel
matrix: Vector
XMatrix()
addRow(Vector): void
getMatrix(): Vector
<<Java Class>>
XAttribute
core.model
attributeName: String
attributeType: String
errorProb: double
errorRange: int
numericValue: double
textValue: String
entryTimeStamp: long
freshness: double
XAttribute(String,String,double)
XAttribute(String,double,double)
XAttribute(String,String,double,double)
XAttribute(String,double,double,double)
XAttribute(String,String,int)
XAttribute(String,double,int)
getType(): String
getName(): String
getTextValue(): String
getnumericValue(): double
getErrorProb(): double
getErrorRange(): int
getEntryTimeStamp(): long
getFreshness(): double
<<Java Class>>
XSubRegion
core.model
predicateList: Vector
contribution: double
XSubRegion()
copyValue(): XSubRegion
getContribution(String): double
getContribution(double): double
getContributionOuterRegions(double): double
<<Java Class>>
Inferencer
core.kernel
computeConfidence(XSpace,XState): double
UpdateSpace(XSpace,XState): XSpace
RootExpiredData(XSpace,XState): XState
handleOptionalCondition(XSpace,XState): void
getContribution(XAttributeRegion,XAttribute,XState): double
getError(XAttributeRegion,XAttribute): double
computeFreshnessContrib(double): double

Figure A3: R-CS Implementation - Detailed Class Diagram (Part 2)

Appendix B



ComPort: COM4
Baudrate: 57600
RX-Display: ASCII

```
Type here...
Use #xxx for ASCII
For example:
#010 = <LF>
```

```
In Bluetooth. Receving and Sending Data
Length of data 0: 11
STARTD 1
The output from procesIncomin: STARTD 1▮ô
Windows Size 1:
Request to Send Data with Window Size 1
FILE Date: 2008-0-1 11:55
Nr samples: 6
  0:  Temp.= 24.9, Vbatt=4.99
  1:  Temp.= 24.9, Vbatt=4.99
  2:  Temp.= 24.9, Vbatt=4.99
  3:  Temp.= 24.9, Vbatt=4.99
  4:  Temp.= 24.7, Vbatt=4.99
  5:  Temp.= 24.9, Vbatt=4.99
Output--> DATA:
Output--> DATA:0:
Year --> 108 1
Output--> DATA:0:FILE:1:d:A:K:w:j:#
Output--> DATA:0:FILE:1:d:A:K:w:j:#24.9@5.0#24.9@5.0#24.9@5.0#24.9@5.0#24.7@5.0#
24.9@5.0#ENDPACKET
Sending Data to Device--> rfcomm_uih_credits: p->tot_len = 93 pcb->k = 7 pcb->rk
 = 1
rfcomm_input: Received local credits: 38 Existing local credits: 7
rfcomm_input: Forward RFCOMM_UIH_PF credit packet to higher layer
com1_recv: p->len == 0 p->tot_len == 0
rfcomm_input: Forward RFCOMM_UIH packet to higher layer
com1_recv: p->len == 3 p->tot_len == 3
In Bluetooth. Receving and Sending Data
Length of data 0: 7
ACK
The output from procesIncomin: ACK▮ôS
Sending Next Set of Data with Window Size 1
FILE:Date: 2008-0-1 12:1
Nr samples: 6
  0:  Temp.= 24.9, Vbatt=4.99
  1:  Temp.= 25.3, Vbatt=4.99
  2:  Temp.= 24.9, Vbatt=4.99
  3:  Temp.= 24.9, Vbatt=4.99
  4:  Temp.= 24.9, Vbatt=4.99
  5:  Temp.= 24.7, Vbatt=4.99
Output--> DATA:
Output--> DATA:0:
Year --> 108 1
Output--> DATA:0:FILE:1:d:A:L:A:j:#
Output--> DATA:0:FILE:1:d:A:L:A:j:#24.9@5.0#25.3@5.0#24.9@5.0#24.9@5.0#24.9@5.0#
24.7@5.0#ENDPACKET
Sending Data to Device--> rfcomm_uih_credits: p->tot_len = 93 pcb->k = 45 pcb->r
k = 1
rfcomm_input: Forward RFCOMM_UIH packet to higher layer
com1_recv: p->len == 3 p->tot_len == 3
In Bluetooth. Receving and Sending Data
Length of data 0: 7
ACK
The output from procesIncomin: ACK▮ô{
Sending Next Set of Data with Window Size 1
No Data To Send
Sending Data to Device--> rfcomm_uih_credits: p->tot_len = 11 pcb->k = 45 pcb->r
k = 1
rfcomm_input: RFCOMM_DISC
rfcomm_ua
rfcomm_disconnected: CN = 1
rfcomm_input: RFCOMM_DISC
rfcomm_ua
bt_disconnect_ind
Bluetooth Serial Port Disconnected
```

Figure B1: Window-based Data Collection: Mulle Sensor Dumps

Glossary

.NET CF: Microsoft .NET Compact Framework platform supporting software development for mobile devices

EPE: Ekahau Positioning Engine

ER1: A robot platform built by Evolution Robotics

GSM: Global System for Mobile Communications

GPRS: General Packet Radio Service

GPS: Global Positioning System

LAP: LAN access profile is a Bluetooth profile that enables Bluetooth devices to access a LAN, WAN or internet via another device

LAN: Local Area Network

Locomotion: The term **locomotion** means movement or travel

Mobile Data Mule: A mobile device that is used as a vehicle to collect and deliver sensor data.

Mulle: A Bluetooth-based sensor node developed at EISLab, Lulea, Sweden.

PDA: Personal Digital Assistant

Piconet: A ad-hoc computer network linking one Bluetooth master to seven slave devices

Smart Phone: A mobile phone with advanced computing features than traditional mobile phones. For example internet-enabled, powerful processing, social networking, etc.

Sink/Base Station: The sink/base station is a centralised location to which collected sensor data is delivered for further processing.

Scatternet: A collection of one or more piconets.

SPP: Serial port profile is a Bluetooth profile that emulates a serial cable.

SDK: Software development kit

UMTS: Universal Mobile Telecommunications System

Wireless Sensor Node: A tiny battery powered resource constrained device that has the capability to sense, process, store and wireless transfer data.

Wireless Sensor Network: A collection of wireless sensor nodes, data sinks, and intermediate mobile node deployed within an area to achieve single or multiple goals.

WLAN: Wireless Local Area Network

WAN: Wide Area Network

WiMAX: Worldwide Interoperability for Microwave Access

Wi-Fi: A trademark of the Wi-Fi Alliance used by manufactures to brand WLAN based devices based on IEEE 802.11 standard

XML: Extensible Markup Language is a set of rules to encode documents in machine readable format

Zigbee: A suite of communication protocols for small low-powered devices based on IEEE 802.15.4