In-network database query processing for wireless sensor networks

This item was submitted to Loughborough University’s Institutional Repository by the author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/36226

Publisher: © Noura Yahya Salim Al-Hoqani

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
In-Network Database Query Processing for Wireless Sensor Networks

by

Noura Yahya Salim Al-Hoqani

A Doctoral Thesis

Submitted in partial fulfillment of the requirement for the award of

Doctor of Philosophy of Loughborough University

October 2018

Copyright 2018 Noura Yahya Salim Al-Hoqani
Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

Noura Al-Hoqani
Abstract

In the past research, smart sensor devices have become mature enough for large, distributed networks of such sensors to start to be deployed. Such networks can include tens or hundreds of independent nodes that can perform their functions without human interactions such as recharging of batteries, the configuration of network routes and others. Each of the sensors in the wireless sensor network is considered as microsystem, which consists of memory, processor, transducers and low bandwidth as well as a low range radio transceiver.

This study investigates an adaptive sampling strategy for WSS aimed at reducing the number of data samples by sensing data only when a significant change in these processes is detected. This detection strategy is based on an extension to Holt's Method and statistical model. To investigate this strategy, the water consumption in a household is used as a case study.

A query distribution approach is proposed, which is presented in detail in chapter 5. Our developed wireless sensor query engine is programmed on Sensinode testbed cc2430. The implemented model used on the wireless sensor platform and the architecture of the model is presented in chapters six, seven, and eight. This thesis presents a contribution by designing the experimental simulation setup and by developing the required database interface GUI sensing system, which enables the end user to send the inquiries to the sensor’s network whenever needed, the On-Demand Query Sensing system ODQS is enhanced with a probabilistic model for the purpose of sensing only when the system is insufficient to answer the user queries. Moreover, a dynamic aggregation methodology is integrated so as to make the system more adaptive to query message costs.

Dynamic on-demand approach for aggregated queries is implemented, based in a wireless sensor network by integrating the dynamic programming technique for the most optimal query decision, the optimality factor in our experiment is the query cost. In-network query processing of wireless sensor networks is discussed in detail in order to develop a more energy efficient approach to query processing. Initially, a survey of the research on existing WSN query processing approaches is presented. Building on this background, novel primary achievements includes an adaptive sampling mechanism and a dynamic query optimiser. These new approaches are extremely helpful when existing statistics are not sufficient to generate an optimal plan. There are two distinct aspects in query processing optimisation; query dynamic adaptive plans, which focus on improving the initial execution of a query, and dynamic adaptive
statistics, which provide the best query execution plan to improve subsequent executions of the aggregation of on-demand queries requested by multiple end-users.

In-network query processing is attractive to researchers developing user-friendly sensing systems. Since the sensors are a limited resource and battery powered devices, more robust features are recommended to limit the communication access to the sensor nodes in order to maximise the sensor lifetime. For this reason, a new architecture that combines a probability modelling technique with dynamic programming (DP) query processing to optimise the communication cost of queries is proposed. In this thesis, a dynamic technique to enhance the query engine for the interactive sensing system interface is developed. The probability technique is responsible for reducing communication costs for each query executed outside the wireless sensor networks.

As remote sensors have limited resources and rely on battery power, control strategies should limit communication access to sensor nodes to maximise battery life. We propose an energy-efficient data acquisition system to extend the battery life of nodes in wireless sensor networks. The system considers a graph-based network structure, evaluates multiple query execution plans, and selects the best plan with the lowest cost obtained from an energy consumption model. Also, a genetic algorithm is used to analyse the performance of the approach.

Experimental testing are provided to demonstrate the proposed on-demand sensing system capabilities to successfully predict the query answer injected by the on-demand sensing system end-user based-on a sensor network architecture and input query statement attributes and the query engine ability to determine the best and close to the optimal execution plan, given specific constraints of these query attributes. As a result of the above, the thesis contributes to the state-of-art in a network distributed wireless sensor network query design, implementation, analysis, evaluation, performance and optimisation.

**Keywords**

Wireless Sensor Networks, In-network query processing, Adaptive sampling, Dynamic query, Probabilistic query model, Query optimisation, Tree-based, Graph-based, Dijkstra query, On-demand query sensing, query plan, Sensinode cc2430, and aggregated queries.
Acknowledgements

First of all, I would like to thank Allah (God) for giving me strength and ability to complete this thesis special appreciation goes to great people who have helped me to overcome all the difficulties and the obstacles that any researchers might face in his/her workforce.

I would like to express my thanks to my supervisor, Professor Shuang-Hua Yang who is a very supportive person. He guided me in novelty the best research direction to make it worth to be a PhD Thesis.

My thanks also go to those who support me in developing the sensing system via query interface data gathering from the different sensors required for my experiment, Mr. Daniel Perez Fiadzeawu and Mr. Hua Yan. Furthermore, we would like to show our gratitude to the IT services team in computer science department at Loughborough University for their patience and help during the implementation of this project.

I would also like to thank my sisters, Zahra, Maryam, Jalila and Hanan, who was very supportive and always available when I asked for assistance. Also, thanks goes to my husband who helped me organize my lifestyle to have the correct schedule for my family and my research, furthermore, gain more time and more knowledge as well.

THANK YOU

Noura Al-Hoqani
Dedication

To my mother and father for their sincere prayers and patience; to my best friend, my husband Labeed Al-Amri for his continuous support and love;

To my future, my heart, my lovely children; Firas, Luqman, Numyr and Falah for the life they bring to our life.
Table of Contents

ABSTRACT ................................................................................................................................. III

CHAPTER 1 INTRODUCTION .................................................................................................... 1

1.1 RESEARCH MOTIVATION ................................................................................................. 1

1.2 PROBLEM STATEMENT ........................................................................................................ 3

1.3 RESEARCH QUESTIONS ........................................................................................................ 5

1.4 RESEARCH OBJECTIVES ..................................................................................................... 5

1.5 THESIS SCOPE AND CONTRIBUTIONS ........................................................................... 8

1.6 THESIS STRUCTURE ............................................................................................................ 10

1.7 ASSOCIATED PUBLICATIONS ............................................................................................. 12

CHAPTER 2 RESEARCH BACKGROUND ...................................................................................... 13

2.1 INTRODUCTION .................................................................................................................. 13

2.2 WIRELESS SENSOR NETWORKS AS A DATABASE ............................................................ 15

2.2.1 In-network Data Acquisition ......................................................................................... 15

2.2.2 Distributed Query Processing ....................................................................................... 17

2.3 QUERIES AND QUERY OPTIMISATION .......................................................................... 18

2.3.1 Ad-hoc Queries .............................................................................................................. 19

2.4 ARCHITECTURE OF A WIRELESS SENSOR NETWORK QUERY PROCESSOR .................. 20

2.5 CONCLUSION ...................................................................................................................... 21

CHAPTER 3 LITERATURE REVIEW ............................................................................................. 22

3.1 INTRODUCTION .................................................................................................................. 22

3.2 SURVEY OF SAMPLING TECHNIQUES IN WIRELESS SENSOR NETWORKS .................. 23

3.2.1 Data reduction techniques ............................................................................................ 23

3.2.2 Data Acquisition Techniques ....................................................................................... 29

3.3 SURVEY OF PROBABILITY-BASED QUERY PROCESSING ............................................... 34

3.3.1 Approximation Techniques ........................................................................................... 35
3.3.2 Aggregation Techniques ................................................................. 37

3.4 Survey of Query-Based Processing Techniques .................................... 38

3.4.1 Push-based and Pull-Based Data Acquisition .................................. 39

3.4.2 Packet Merging .............................................................................. 40

3.4.3 Execution Query Cost ..................................................................... 41

3.4.4 Join Query ..................................................................................... 41

3.5 Survey of Distributed Query Techniques ........................................... 42

3.5.1 TinyDB Project .............................................................................. 42

3.5.2 ADAGA Project ............................................................................ 45

3.5.3 ADAGA-P .................................................................................... 46

3.5.4 TiNA Project ................................................................................ 46

3.5.5 SNEE .......................................................................................... 47

3.5.6 SenQ ............................................................................................ 47

3.5.7 Corona .......................................................................................... 47

3.6 Survey of Query-Based Optimisation Techniques .............................. 50

3.6.1 Query-based data gathering model ................................................. 51

3.6.2 Base Station Optimisation ............................................................... 51

3.6.3 Heuristic approach ....................................................................... 52

3.6.4 Cost-Base Optimiser ..................................................................... 53

3.6.5 In-network Query Optimisation for Wireless Sensor Database ......... 54

3.7 Conclusion ....................................................................................... 60

CHAPTER 4 Adaptive Sampling for Wireless Sensor Networks .......... 61

4.1 Introduction ...................................................................................... 61

4.2 Time Series Prediction Model ............................................................. 64

4.3 TCP Congestion Control .................................................................... 66

4.4 AS-TCP Algorithm Phases ................................................................. 68

VIII
4.5 DESIGN OF THE ADAPTIVE SAMPLING ALGORITHM .............................................................. 70
   4.5.1 Algorithm design ............................................................................................................ 70
   4.5.2 Raw Dataset ..................................................................................................................... 71
   4.5.3 AS-TCP Algorithm ......................................................................................................... 72
   4.5.4 Performance metrics ..................................................................................................... 73
4.6 EXPERIMENTAL RESULTS ............................................................................................. 74
   4.6.1 Test results on the experimental dataset ........................................................................ 74
   4.6.2 Relationship between critical parameters and performance metrics ......................... 75
   4.6.3 Comparison with existing algorithms .......................................................................... 76
4.7 CONCLUSIONS .................................................................................................................. 78

CHAPTER 5 IN-NETWORK ON-DEMAND QUERY-BASED SENSING SYSTEM FOR WIRELESS SENSOR NETWORKS ................................................................. 79
5.1 INTRODUCTION .................................................................................................................. 79
5.2 ON-DEMAND PROCESSING .......................................................................................... 81
   5.2.1 Query processing Framework ...................................................................................... 81
   5.2.2 Propagation of Query Statements ............................................................................... 84
5.3 QUERY EXPERIMENTAL ENVIRONMENT .................................................................... 85
   5.3.1 Contiki Operating System ........................................................................................... 85
   5.3.2 Sensor Tree Structure Technology ............................................................................... 86
   5.3.3 Wireless Sensor Networks Design ............................................................................. 86
   5.3.4 Sensor Node Hardware ............................................................................................... 87
   5.3.5 Architecture of the WSN in the Setup ........................................................................ 88
5.4 SENSOR NETWORK DATABASE MODEL ....................................................................... 89
   5.4.1 Sink Aggregation code ............................................................................................... 90
   5.4.2 Router Aggregation codes ........................................................................................... 90
5.5 QUERY-BASED SENSING SYSTEM ............................................................................... 91
5.5.1 Sensing System Interface ................................................................. 91
5.5.2 Sensor Storage Database ................................................................. 94

5.6 Dynamic Programming Methodology for On-demand Query Processing ..... 95
  5.6.1 Query Communication Cost ............................................................. 96
  5.6.2 Energy Consumption ..................................................................... 96

5.7 Evaluation and Performance ............................................................... 97
  5.7.1 Aggregation ................................................................................... 97
  5.7.2 Merging ........................................................................................ 97
  5.7.3 Query lifetime clause .................................................................... 100
  5.7.4 Sampling interval clause ............................................................... 100
  5.7.5 Dual aggregation level .................................................................. 100

5.8 ODQS Architecture ............................................................................. 101
  5.8.1 ODQS Sensing Testbed Design Architecture .................................... 101
  5.8.2 Query Processing Engine Design .................................................. 102

5.9 Implementation .................................................................................... 102
  5.9.1 On demand Sensing System Development ...................................... 103
  5.9.2 Query Processing Functions ......................................................... 104
  5.9.3 Query Plans Structure ................................................................. 105

5.10 Experimental Methods ....................................................................... 106
  5.10.1 Query Processing Methodology .................................................... 106
  5.10.2 Data Aggregation Functions ......................................................... 108
  5.10.3 Merging Queries .......................................................................... 109
  5.10.4 Query communication cost ........................................................ 110

5.11 Results ............................................................................................... 111
  5.11.1 Discussion of query engine outcomes .......................................... 111
  5.11.2 Routing Aggregations Results .................................................... 112
7.2.2 Network initialization ........................................................................................................... 138
7.2.3 Query execution plan optimization ....................................................................................... 139

7.3 Graph Theory Optimization ..................................................................................................... 140
7.3.1 Tree-based search using BFS (Breadth-First Search) algorithm ........................................... 140
7.3.2 Graph-based search using Dijkstra algorithm ........................................................................... 141
7.3.3 Query elapsed time ............................................................................................................... 144

7.4 Experimental setup .................................................................................................................. 145
7.4.1 WSN structure and query processing ..................................................................................... 145
7.4.2 Multiple query sharing ......................................................................................................... 145
7.4.3 Cost-based model ................................................................................................................ 148
7.4.4 Genetic algorithm presentation ............................................................................................. 149

7.5 Query Database Evaluation Model ........................................................................................... 150
7.5.1 Execution cost fitness function ............................................................................................ 150
7.5.2 Genetic algorithm ............................................................................................................... 151

7.6 Results and Discussion ............................................................................................................. 152
7.7 Conclusion ............................................................................................................................... 154

CHAPTER 8 Thesis Summary and Future Work .............................................................................. 155
8.1 Research Contributions ............................................................................................................. 155
8.2 Achievement of Thesis Objectives .......................................................................................... 155
8.3 Thesis Summary ....................................................................................................................... 156
8.4 Future Work ............................................................................................................................. 157

References ........................................................................................................................................ 160
List of Figures

FIGURE 1-1 TAXONOMY FOR WIRELESS SENSOR NETWORKS .................................................. 4
FIGURE 2-1 DATA PROCESSING APPROACH ................................................................. 17
FIGURE 2-2 QUERY PROCESSING ARCHITECTURE .......................................................... 20
FIGURE 3-1 ENERGY CONSERVATION TECHNIQUE CATEGORIES .................................. 23
FIGURE 3-2 DIFFERENT TYPES OF ADAPTIVE SENSING .................................................. 29
FIGURE 3-3 AGGREGATION STEPS OF SENSOR READINGS IN A SENSOR NETWORK DIALLO ET AL. (2013) ............................................................................................................................................ 43
FIGURE 3-4 QUERY PLAN AT A SOURCE SENSOR IN THE COUGAR APPROACH (YAO AND GHERKE ON COUGAR) .............................................................................................................................................. 45
FIGURE 3-5 IN-NETWORK AGGREGATION QUERY USING TiNA (DIALLO ET AL., 2015) ........ 46
FIGURE 3-6 CORONA QUERY PROCESSOR ON SUN SPOT (KHOURY ET AL., 2010) ............... 47
FIGURE 3-7 IN-NETWORK QUERY OPTIMISATION FOR SENSOR NETWORK DATABASE ........ 55
FIGURE 4-1 TCP CONGESTION CONTROL AIMD MECHANISM ............................................. 67
FIGURE 4-2 AS-TCP ALGORITHM PROCESS DESIGN ......................................................... 71
FIGURE 4-3 AS-TCP PSEUDO CODE ..................................................................................... 72
FIGURE 4-4 COMPARISON OF SENSED DATA, MISSED DATA, PREDICTED DATA WITH THE ACTUAL DATASET .................................................................................................................. 73
FIGURE 4-5 SAMPLING FRACTION AND MISSING RATIO IN DIFFERENT ERROR TOLERANCE ... 75
FIGURE 4-6 SAMPLING FRACTION AND MISSING RATIO IN DIFFERENT MAXIMUM STEP SIZE ...... 75
FIGURE 4-7 SAMPLING PERFORMANCE IN DIFFERENT MAXIMUM STEP SIZE AND ERROR TOLERANCE .............................................................................................................................................. 76
FIGURE 4-8 SAMPLING PERFORMANCE COMPARISON WITH E-SENSE AND EDSAS ............ 77
FIGURE 5-1 FRAMEWORK FOR QUERY PROCESSING IN WSN ............................................. 82
FIGURE 5-2 SENSINODE WITH TI BOARD CC2430 ............................................................. 88
FIGURE 5-3 ON-DEMAND TREE-BASED QUERY ARCHITECTURE FOR WSN ........................ 89
FIGURE 5-4 ON-DEMAND GRAPH-BASED QUERY ARCHITECTURE FOR WSN ................... 89
FIGURE 5-5 ON DEMAND QUERY INTERFACE ........................................................................ 92
FIGURE 5-6 QUERY RESULT FORM FOR TEMPERATURE GRAPH .......................................... 93
FIGURE 5-7 QUERY RESULT FORM FOR LIGHT GRAPH ....................................................... 93
FIGURE 5-8 QUERY RESULT FORM FOR BATTERY CONSUMPTION GRAPH ..................... 94
FIGURE 5-9 QUERY RESULT FORM FOR COMPARING ALL MONITORED DATA .................. 94
FIGURE 5-10 JSON FORMAT QUERY-BASED STORAGE ...................................................... 95
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE 1-1</td>
<td>Research Questions and Thesis Contributions</td>
<td>7</td>
</tr>
<tr>
<td>TABLE 3-1</td>
<td>Summary of Some Data Reduction and Adaptive Sampling Used Techniques in WSN</td>
<td>28</td>
</tr>
<tr>
<td>TABLE 3-2</td>
<td>A Description of Some SQL Aggregation Functions</td>
<td>38</td>
</tr>
<tr>
<td>TABLE 3-3</td>
<td>Summary of the Existing Distributed Query Sensing Projects</td>
<td>49</td>
</tr>
<tr>
<td>TABLE 3-4</td>
<td>Summary of In-Network Query Optimisation Research</td>
<td>59</td>
</tr>
<tr>
<td>TABLE 4-1</td>
<td>Performance Metrics when $S_{max}$ is Between 500 and 1200 and $\Delta=0.99$</td>
<td>74</td>
</tr>
<tr>
<td>TABLE 4-2</td>
<td>Performance Metrics when $S_{max}=600$ and $\Delta$ is Between 0.30 and 0.99</td>
<td>74</td>
</tr>
<tr>
<td>TABLE 4-3</td>
<td>Comparison with E-Sense and EDSAS</td>
<td>77</td>
</tr>
<tr>
<td>TABLE 5-1</td>
<td>Parameters for Query Cost</td>
<td>96</td>
</tr>
<tr>
<td>TABLE 5-2</td>
<td>ODQS Implemented Query Functions</td>
<td>104</td>
</tr>
<tr>
<td>TABLE 5-3</td>
<td>ODQS None-Aggregated Outcomes</td>
<td>110</td>
</tr>
<tr>
<td>TABLE 5-4</td>
<td>ODQS Aggregated Outcomes</td>
<td>111</td>
</tr>
<tr>
<td>TABLE 6-1</td>
<td>ODQS-DP Model Parameters</td>
<td>125</td>
</tr>
<tr>
<td>TABLE 7-1</td>
<td>Adjacent List for a Graph Query Network Architecture</td>
<td>140</td>
</tr>
<tr>
<td>TABLE 7-2</td>
<td>Fitness Ratio to Evaluate Query Results</td>
<td>141</td>
</tr>
<tr>
<td>TABLE 7-3</td>
<td>Sharing Data for Complex Query Approach</td>
<td>146</td>
</tr>
<tr>
<td>TABLE 7-4</td>
<td>Performances of Different Query Algorithms</td>
<td>147</td>
</tr>
<tr>
<td>TABLE 7-5</td>
<td>Measured Energy Model Parameters for the TMote Sky Platform</td>
<td>148</td>
</tr>
<tr>
<td>TABLE 7-6</td>
<td>Genetic Algorithm Parameters</td>
<td>152</td>
</tr>
</tbody>
</table>
### List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>Adaptive Model Selection</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>ASAP</td>
<td>Adaptive Sampling Approach</td>
</tr>
<tr>
<td>AODV</td>
<td>Ad hoc On-Demand Distance Vector</td>
</tr>
<tr>
<td>BBQ</td>
<td>Barbie-Q A Tiny Model Query System</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>CH</td>
<td>Cluster Head</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management Systems</td>
</tr>
<tr>
<td>DDBMS</td>
<td>Distributed Database Management System</td>
</tr>
<tr>
<td>DP</td>
<td>Dynamic Programming</td>
</tr>
<tr>
<td>EDCA</td>
<td>Efficient Data Collection Approach</td>
</tr>
<tr>
<td>EDS</td>
<td>Exponential Double Smoothing</td>
</tr>
<tr>
<td>EDSAS</td>
<td>Exponential Double Smoothing Based Adaptive Sampling</td>
</tr>
<tr>
<td>EDSAS-S</td>
<td>Exponential Double Smoothing Based Adaptive Sampling - Spatial</td>
</tr>
<tr>
<td>EDSAS-T</td>
<td>Exponential Double Smoothing Based Adaptive Sampling - Temporal</td>
</tr>
<tr>
<td>EEDC</td>
<td>Energy Efficient Data Collection</td>
</tr>
<tr>
<td>ES</td>
<td>Exponential Smoothing</td>
</tr>
<tr>
<td>EWMA</td>
<td>Exponentially Weighted Moving Average</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Prediction</td>
</tr>
<tr>
<td>MA</td>
<td>Moving Average</td>
</tr>
<tr>
<td>MODSS</td>
<td>Multiple On-Demand Sensing System</td>
</tr>
<tr>
<td>MR</td>
<td>Miss Ratio</td>
</tr>
<tr>
<td>ODQS</td>
<td>On-Demand Query-based Sensing system.</td>
</tr>
<tr>
<td>PAQ</td>
<td>Probabilistic Adaptable Query System</td>
</tr>
<tr>
<td>SES</td>
<td>Simple Exponential Smoothing</td>
</tr>
<tr>
<td>Abbr.</td>
<td>Term</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>SF</td>
<td>Sampling Fraction</td>
</tr>
<tr>
<td>SN</td>
<td>Sensor Nodes</td>
</tr>
<tr>
<td>SNQP</td>
<td>Sensor Network Query Processing</td>
</tr>
<tr>
<td>SP</td>
<td>Sampling Performance</td>
</tr>
<tr>
<td>SQL</td>
<td>Structural Query Language</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
<tr>
<td>WSS</td>
<td>Wireless Sensing Systems</td>
</tr>
</tbody>
</table>
Chapter 1 Introduction

This chapter presents the research motivations. It also introduces the research scope and contributions and sets out in detail the problem statement and research objectives.

1.1 Research Motivation

A wireless sensor network (WSN) is composed of a collection of sensor nodes placed in numerous geographical locations in order to understand and control the state of the environment in which it operates (Stankovic, Wood & He, 2011). These sensor nodes each contain basic sensing components, data processing components and communication components (Diallo et al., 2015). A diverse range of fields benefit from the ability to sense and collect data from numerous geographical locations, such as air pollution and water quality monitoring, and monitoring of machinery conditions and structural health, and Potdar et al. (2009) suggest that, shortly, wireless sensor networks will also contribute immensely to developments in space exploration, chemical processing and disaster relief.

According to Oliveira & Rodrigues (2011), there are still some challenges that need to be addressed in respect to the practical operation of WSNs, such as power management, which is essential for their long-term operation, especially in the context of sensors monitoring remote environments. In addition, scalability for large WSN networks, so that development continues to grow theoretically, as well as practical solutions, are ready for real WSN. Moreover, they suggest that remote management systems should be designed to enable WSNs to be operated via remote access standards so that they can be configured and managed more securely. According to the Shrivastava & Pokle (2014), the major constraint of any sensor is that they are battery operated devices, which means that they have a limited power supply. With this in mind, they state that the communication module is the dominant energy consumer in a sensor. Consequently, it is essential for sensors to collaborate and minimise communication activities, as well as utilises the sensor’s resources. As a result, the choice of the most suitable approach for collecting the data, depending on the environmental situation, is an important one. Furthermore, as Chong & Gaber (2011) pointed out, the data generated by sensor nodes in a WSN needs to undergo some analysis and processing if it is to provide meaningful information for users. Given the limited memory and power resources of sensor nodes, however, it is essential that processing techniques are developed that are able to cope with these limitations
if the lifetime of the network is to be extended and efficiency to be improved. Specific areas to explore in terms of the management of WSNs include:

1. Wireless sensor networks are used for applications such as monitoring environmental such as light, temperature, humidity and sound. During the daytime, the servers are expected to be empty because there are fewer disasters during daytime, when a lot of people are around for monitoring, and thus we can have a low sampling rate. In summary, sensors need to be managed in such a way that they are only used when there is a significant need to collect data from them, thereby minimising power consumption and maximising the lifetime of the sensors.

2. In some sensor applications, such as where multiple cameras are mounted at a location to monitor the activity of vehicles or people, for example in parking areas, cameras need to detect any behavior like speeding or random circling. The sampling rates of those specific nodes should be increased, and the other nodes’ sampling rates decreased for the same reason of saving energy and getting more accurate readings.

3. While database management systems often have powerful abilities in terms of maintaining data availability, and in their scalability, adaptability and sustainability, data management in the context of wireless sensor networks is complicated by the limited technical specifications of their microprocessors. This makes it difficult to operate WSNs as DBMS with high accuracy, as well as high accessibility and availability.

While many researchers have focused on reducing data transmissions by as much as possible in order to conserve energy (Wong & Zhu, 2014), they have tended to neglect the data sampling aspects of WSN management. In practice, however, sampling or data acquisition consumes considerably more energy, and, therefore, some researchers have now started investigating energy-efficient sampling acquisition and sensing systems, which will be discussed in later chapters. Xiang's (2011) research thesis found that the design of ad-hoc queries is essential for reducing the number of messages and this will affect sensor nodes’ query transmission time. Moreover, it is stated that if the query is region-based or node-based, the set of answer nodes can be identified in advance of the query response. This direction of research is very interesting since it offers the potential to respond to most wireless sensor communication queries while minimising the energy costs involved.
1.2 Problem Statement

The problem addressed in this thesis is the description of the different characteristics of query optimisation and the proposal of a novel solution for query optimisation for real-time databases in the context of WSNs. The data collected by WSNs must closely report the current state of the targeted environment. To this end, WSNs continuously collect data in different discrete seconds of time. Since the environment keeps fluctuating, however, and different important events might occur at any moment, there is a risk that with discrete data collection inaccuracies may creep in. In order to overcome this problem a task is needed to alert the user when the environment is no longer being accurately replicates the state of the environments (Diallo, 2014). A real-time approach to data collection, however, raises significant issues in terms of latency and energy-efficiency due to the resource limitations of WSNs.

The inspiration of this thesis came from the database management administrator’s community. As the database administrator communities are capable to querying their databases only on request to save further space in their CPU’s memory, the sensor’s database has a very low memory, and therefore needs to be consumed in a very intelligent ways. Moreover, we aim to ensure the lifetime of the sensor is maximised as a result of the best adapted methodology. The particular concern is the problem of selecting the less costly energy for any query request by multiple sensor database management end-users.

Although Xiang (2011) covered the database challenges in WSN query processing, a query propagation algorithm for value-based queries was not covered. They did, however, advise on the design of a probabilistic model to predict the data distribution and thus to guide the query broadcasting process. This forms part of our proposed model in which we use the ODQS system to save energy by requiring nodes to be visited for data only when the model is not sufficient to answer the query. From another point of view, the number of transmitted messages is not reduced, where it is very essential to reduce the number of sensed data by generating a query from the sensor’s network database only at specific time intervals.

Figure 1-1 shows how wireless sensor networks encapsulate many technologies, each of which attracts researchers. Some technologies mentioned in Zhang and Zhang (2012) are routing protocols, time-synchronisation, localisation, data aggregation, power management, security and administration. WSNs also have broad applications which Rault et al. (2014) categorise as falling under environment and agriculture, public safety and military systems,
healthcare and industry, with the authors giving specific instances of these as listed in the Figure 1-1. Having considered many aspects of the wireless sensor network, this project focuses on in-network data processing for data acquisition, with an emphasis on structuring on-demand queries in a way that saves energy compared to other query techniques.

In-network data processing involves various ways to route packets in order to combine data from different sources that are directed to the same destination (Fasolo et al., 2007). This technique aims at reducing energy consumption and improving the lifespan of sensor networks (Chen et al., 2006). In this technique, an intermediate node, called a proxy node, is chosen to perform a data transformation function, such as data aggregation, on samples received from sensing nodes before forwarding them to the destination, sink node (Chen et al., 2006).

In summary, according to the research literature, the data collected by sensors are sent periodically to the base station where real-time communications using many statements and transactions are processed. This is very resource intensive and, therefore, a new approach to the

Figure 1-1 Wireless sensor networks usages
optimisation of query processing is required to improve the important aspects of query latency and response time to the query, as well as the wireless sensor network lifetime.

1.3 Research Questions

For the introduced topic above, we set a set of question, which we are looking forward to contribute in this research thesis, they are as follow: -

1. What is the difference between the data from the sampling sensors ‘sampling approach’ and the data from querying sensors ‘querying approach’ to data collection in WSNs? Moreover, what are the different query types that can be used in sensor network databases to save communication energy?

2. Why is sampling being so essential when collecting data from sensors? Also, why are adaptive sampling strategies used to reduce the data samples when sensing data?

3. Why does data processing consume more energy than other WSN components? Moreover, what is the best practice for ad hoc historical/random queries in dynamic sensing systems?

4. What techniques are suitable for an ODQS dynamic approach instead of a static approach? Moreover, what strategies can save the most energy using statistical approaches?

5. What is required to transmit queries in any network structure without limit to the tree-based database? Moreover, what enhancements are required for complex database query processing in WSNs?

Table 1-1 illustrates the research questions and contribution of the thesis for more clarity. The bullet points above are organised in the table.

1.4 Research Objectives

In-network on-demand query-based sensing systems and sensor’s database optimisation need to be tightly aligned if data is to be processed efficiently. Below are the objectives.

- Building on the research questions, conduct a literature review to identify the direction of the research.
- To identify research gaps with respect to query processing in WSNs.
• To construct a comprehensive technical view about the methodology used in previous research.

• To identify and evaluate the algorithm to be used for adaptive query optimisation.

• To evaluate sampling performance and the sampling ratio, as well as the missing rate enhancement.

• To develop and evaluate an On-Demand Query Sensing system and test the aggregation methodologies used, such as averaging, maximum, minimum, and then to merge the data from different query sensors using the interface designed.

• To evaluate the deployed hypothesis for sensor energy efficiency and lifetime maximisation.

• To introduce a graph-based mathematical model to share readings among all the different resources in the in-network sensor databases.

• To develop a genetic algorithm model using the MATLAB tool and to integrate this with the available on-demand query sensing (ODQS) system to achieve the best evaluation of our query-based system.

This thesis delivered an in-network on-demand query-based sensing system that includes two highly energy efficient methodologies to respond to the end users using an on-demand interface. The work in this thesis can be considered as a three-tier architecture since it involves three parties: the historical database as a data warehouse; the wireless network cache memory; and the wireless sensor network.
<table>
<thead>
<tr>
<th>Research Questions (RQ)</th>
<th>Research Method</th>
<th>Chapter/Published paper</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RQ1:</strong> What is the difference between the data from the sampling sensors and the data from querying sensors? Moreover, what are the different query types that can be used in sensor network databases to save communication energy?</td>
<td>Literature review</td>
<td>Chapter 3</td>
<td>1. We outline the challenges in conducting in-network query processing in WSNs, compare the existing query processing and define the limitations of the different researches approaches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. A literature review is conducted to identify gaps in the current research and thus opportunities for further scientific improvement.</td>
</tr>
<tr>
<td><strong>RQ2:</strong> Why is sampling so important when collecting data from sensors? Also, why are adaptive sampling strategies used to reduce the data samples when sensing data?</td>
<td>Experimental testing</td>
<td>Conference published</td>
<td>1. We propose a TCP congestion control based adaptive sampling technique that produces better performance better than previous approaches. The TCP-based technique is tested using sampling ratio and performance measurements.</td>
</tr>
<tr>
<td><strong>RQ3:</strong> Why does data processing consume more energy than other WSN components? Moreover, what is the best practice for ad-hoc historical/random queries in dynamic sensing systems?</td>
<td>Textbed programming and methodology testing</td>
<td>Conference published</td>
<td>1. We develop a tree-based ODQS and present how on-demand query processing in Wireless Sensing System (WSS) is controlled in a sensor network query processing database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. The proposed ODQS enables a flexible enquiry mechanism for the end user, applying a combination of interval, period and various aggregation algorithms to enhance sensing efficiency and energy savings.</td>
</tr>
<tr>
<td><strong>RQ4:</strong> What techniques are suitable for an ODQS dynamic approach instead of a static approach? Moreover, what strategies can save the most energy using statistical approaches?</td>
<td>Experimental testing</td>
<td>Conference published</td>
<td>1. We propose a probability approach and a dynamic programming method for query optimisation so that the ODQS approach is able to make the best use of the historic data and thus save energy.</td>
</tr>
<tr>
<td><strong>RQ5:</strong> What is required to transmit queries in any network structure without limit to the tree-based database? Moreover, what enhancements are required for complex database query processing in WSNs?</td>
<td>Optimisation techniques using graph theory and genetic algorithm</td>
<td>Chapter 7 Journal submitted</td>
<td>1. A model-based query optimum planning method is proposed for the processing of multiple queries. In detail, the Dijkstra algorithm is used to select the best aggregation node and the GA is used to compare the results from the graph-based algorithm and the sensor database system for wireless sensor networks.</td>
</tr>
</tbody>
</table>

Table 1-1 Research questions and thesis contributions
1.5 Thesis Scope and Contributions

WSN applications have been the focus of much recent research. To provide such applications, and given the appropriate WSN hardware, researchers first need to design the protocols that can be employed in WSNs. After implementing these protocols, the researchers should test and verify the protocols to eliminate as many problems as possible. Then, the WSN applications can be deployed in the real field. Nonetheless, the after-deployment maintenance is very crucial for keeping the WSN applications running normally. The on-demand in-network query process for wireless sensor network is shown in Figure 1-1. This thesis focuses on three areas of this process, namely, on-demand query sensing system framework design using the most appropriate network AODV protocol, In-network on-demand sensing system testing, and application analysis identification, to obtain dependable WSNs. Specifically, the work in this thesis is shown in the testbed block in Figure 1-1, which shows tree-based network structure. The research scope is also extended to include a graph-based network structure as shown in Figure 1-2 for disseminating queries using our On-demand Query based Sensing System (ODQS) proposed new approaches for high-end query optimisation. Moreover, the work in this thesis deploys added features, which is suggested to be higher energy efficiency when sensing the data according to user’s requirements. The statistical model is integrated at the system access level for prediction or query estimation answer, on the other hand, the dynamic programming for iteration algorithm is integrated at the sink level to ensure that the data are aggregated at the best network structure level, such as pc-level, sink-level, router-level, or sensor, level. This approach helps to reach to more dynamic query retrieval solutions for multiple queries from multiple sensors as well as complicated queries.

![Figure 1-2 Tree-based sensing system for in-network query of WSN](image)

8
The main contributions of this thesis are as follows: -

1. We outline the challenges in conducting in-network query processing in WSN and identify different researches approaches, thereby underscoring the limitations of the existing sensor network query processing (SNQP) approach.

2. In the field of sampling sensor data, we propose a Transmission Control Protocol (TCP) congestion control-based approach as an adaptive sampling technique, arguing that this produces better performance than the techniques adopted in previous research.

3. We present the architecture of in-network On-Demand Query Sensing (ODQS) system query processing framework’s, and a sensor database system for wireless sensor networks when adapted using AODV protocol. This system introduced in chapter five, which has a historical oriented database features, which keeps copy updated when sensing occurs.

4. We develop a real-time sensing system with an on-demand feature to enable flexible end-user enquiries for some or all of the field’s sensors. This is presented in chapter six.

5. We tailor the above system in chapter seven with optimal combinations of intervals, periods and aggregations, and provide an estimation statistical and dynamic algorithm on the on-demand sensing system to achieve efficiency and energy savings.

6. In chapter seven, we propose a probability model and a dynamic programming model for ODQS system optimisation in in-network query processing using the historical model with more records updates at each sensing process.
7. Finally, a multiple query mathematical model is proposed for in-network query processing using the Dijkstra algorithm to find the best aggregation node. Moreover, in chapter eight, we use a genetic algorithm method to compare the query engine decisions.

1.6 Thesis Structure

This section sets out the structure of this thesis. The thesis is structured into nine chapters:

1. Chapter one, ‘Research Introduction’. This chapter introduces the research topic and the research motivations. It also includes the scope of the thesis and contributions made by the research outcomes. The problem statement, research question and research objectives are illustrated here as well.

2. Chapter two, ‘Research Background’. This chapter introduces the background knowledge on WSNs as a form of a database, and the different architectures employed in sensor query processing in current sensing systems. This is followed by an introduction to different query optimisation mechanisms and techniques on related works.

3. Chapter three, ‘Literature review’. This chapter provides a literature review of important concepts related to WSNs, relevant to the work in this thesis. Five different surveys are conducted and discussed. Various conclusions from the existing research are identified and their limitations discussed in order to establish an appropriate direction for this research.

4. Chapter four, ‘Adaptive Sampling for Wireless Sensor Networks’. This chapter provides a study on Adaptive Sampling techniques for the purpose of energy efficiency. For this study, household water consumption is used as a case study. The implementation is performed using the TCP method as a basis for an algorithm.

5. Chapter five, ‘In-network On-demand Query based Sensing System for WSNs’. In this chapter, the in-network query processing for distributed query systems is presented. Many query processing techniques are discussed, reviewed, and various architectures and models are introduced in detail to determine their output advantages. The proposed ODQS system is designed for the purpose of experimental testing and the evaluation of the different aggregation methodologies and merges hypotheses. A sensor database base model is introduced with a tree-based network structure. The chapter presents a new aggregation and merges approach for query optimisation. The implementation and the evaluation of the proposed technique’s functions and strategies are described. The tree-based model
architecture is also presented in detail. Sensing system comparisons are given for merge and non-merge, aggregate and non-aggregate queries using the designed ODQS system. The sensed data are available for reusability to save energy from the sensor for freshness data factor because injected queries are always analysed.

6. Chapter six, 'Probability Model-based On-demand Query Processing for Wireless Sensor Networks Database'. This chapter proposes a probabilistic model to adapt the ODQS system for use in tandem with the historical database so that sensing is undertaken only when the model is not sufficient to answer the query. This chapter integrates this novel approach to aggregation methodology for the end user requirements. The dynamic programming algorithm is also incorporated to find the last sensed queries at a lower cost and retrieve these for the end user.

7. Chapter seven, ‘Graph-based Complex Query processing and Genetic Algorithm for Wireless Sensor Network’s Database’. This chapter tackles the problem of multiple complex queries in a graph-based network structure with the aim of providing the best aggregation node that saves the most energy by generating the most accurate query answer with the fewest hops around the network. We then implement the Dijkstra algorithm and evaluate the output using the well-known genetic algorithm so as to select the best sensor node for the best resulting query plan. In this chapter, we look at both the aggregation queries and the non-aggregation queries. Furthermore, our model also analyses whether the query can be answered by the historical database or the sensor cache database, or whether it needs to be disseminated to the network to wait for an answer.

8. Chapter eight concludes the thesis with a summary and a plan for future work. This chapter highlights the most important conclusions and contributions of this thesis and pinpoints directions for further research work.
1.7 Associated Publications

The following papers have been published/accepted for publication:

**Conference Papers**


**Journal Papers**

Submitted

5. Graph-based Complex Query and Genetic Algorithm for Wireless Sensor Networks Database.  *International Journal of Automation and Computing*, March 2018
Chapter 2 Research Background

This chapter describes the background of wireless sensor networks and gives a general overview of wireless sensors from a database perspective by identifying the different techniques described in the current literature. It then sets out the foundation of our study on query optimisation in wireless sensor network databases.

2.1 Introduction

In a paper by Razzaque et al. (2013), it was stated that the energy consumption of a node is expressed as the total of the sampling energy, the computational energy, the energy of switching and the communication energy. The sensor communication cost is more expensive than other sensor lifetime components, which affects the overall energy consumption.

In the past few years, interest in WSNs has increased considerably from both the networks and the database communities. As described by Aggarwal (2013), sensors can perform the following tasks:

- Collect readings from the environment
- Receive data from other sensors in its network
- Perform some computations on the received and collected data
- Transmit data to the WSN
- Be active whenever it is required to be, otherwise staying in sleep mode.

The typical query directed to a network sensor contains the following information, which is retrieved upon the user’s request:

- The sampling frequency (how often samples are collected).
- Which attributes are to be sampled (e.g. temperature readings)
- Specified constraints on the returned values for the readings (condition applied)

Databases are considered as the authoritative and comprehensive sources of gathering information. It is noticed that wireless sensor networks are influenced by database technologies extensively. One specific area associated with this is sensor net querying which demonstrates the opportunity present for researchers of the database in the process of application of their expertise in this field. However, for programming for the whole network of sensor nodes instead of programming for individual nodes, declarative querying is observed as the most powerful.
With the help of database query engine approach, the answer related to a query is dependent on the availability and accessibility of the data (Brayner, et al., 2008; Zhao & Guibas, 2003).

It should be noted that wireless sensor networks are extensively influenced by database technologies. One specific area associated with this is sensor net querying, which is the process of detecting the existence of a static event, it is also considered as the initiate data collection process as discussed by Can & Demirbas, (2012) in their research. Querying a node obtained all node’s information, which includes query timestamp, sensor readings such as temperature, light, humidity. According to Ren & Liang (2007), there are two conventional ways of handling queries, warehousing and on-demand approaches. The warehousing approach is similar to the database management approach where the base station collects and store the data periodically, whereas, the on-demand approach collects the data based on user’s requests. For programming the whole network of sensor nodes instead of just individual nodes, however, database community considered declarative querying s to be the most powerful approach. According to research Brayner et al., with the help of a database query engine approach, answers to queries are dependent on the availability and accessibility of the data (Brayner, et al., 2008).

It is found that a sensor node may have one or more devices linked to it that are attached to the physical world. Temperature and light sensors are an important example of these devices that can evaluate the incidence of actions such as the presence of an object in their locality. All sensors are therefore a distinct information source with associated data histories linked to numerous fields such as the identification and place of the instrument that produced the interpretation, a time point, the sensor category, and the amount of the reading (Klein, 2004; Buratti, et al., 2009). Records from similar sensor kinds of sensors from diverse nodes that have a similar plan, and communally form a disseminated table and sensor structure might, therefore, be measured across a broadly disseminated database structure containing numerous tables of diverse kinds of sensors. This offers the potential to attain outcomes that are more appropriate by combining information from different sensors. Ex extractions or collections of raw sensor information are consequently more valuable to sensor presentations than distinct sensor interpretations. For instance, when assessing the focus of a hazardous chemical in a zone, one probable enquiry is to evaluate the normal value of all sensor interpretations in that area, and report when it is greater than some predefined inception (Mitchell, 2007; Coman, 2007)
2.2 Wireless Sensor Networks as a Database

Itl Education Solutions Limited (2010) notes that database systems are judged by their capability to spontaneously locate the assessment strategies for queries that are declaratively stated. Depending on the intricacy of a query, the availed query is processed in a variety of techniques. Hence, the query optimizer in use identifies and builds the best query evaluation plan; say on the basis of the cost. So, depending on the expression that has been provided, the query optimizer produces alternate strategies, which yield a similar outcome. The query optimizer then identifies the one with the minimal cost or the most cost-effective. Hence, there is the cost-based query optimisation technique. As it is indicated, under the cost-based query optimisation technique, from the availed query, the optimizer produces several query evaluation plans. It does this through the use of several equivalence guidelines. Based on these guidelines, the most cost-effective query is selected.

In database management, the term data aggregation refers to any process by which information can be gathered and then expressed in a summary form. Aggregation presents data in a form that reveals more information about particular groups based on specific characteristics or variables (TechTarget, 2015). In traditional database systems, queries are demonstrated as the rational set of data that is of importance for the user. This type of database, however, does not illustrate the specific procedures and software components or operatives that are used by the system in order to gather the answer set. For any sort of logical query, the system has the ability to determine from different operator orderings and plans. In queries, the process of choosing the best potential plan is known as query optimisation, which is considered to be important as it assists in the implementation process of query application (Qingchun Ren & Liang, 2007) (Coman, 2007).

2.2.1 In-network Data Acquisition

Data Acquisition is the task responsible for efficiently acquiring samples from the sensors in sensor networks. The acquisition approaches are classified into two major types: pull-based and push-based approach. In the pull-based approach, data are only acquired at a user level input interface to read historic records, which we can call it data warehouse, whereas in the push-based approach, the sensor and the base station agree that sensors are only send data to the base station when an expected behavior occurs (Aggarwal, 2013), which we can call it sensor’s network database. In addition to main objective, which is acquiring the most accurate
and useful data of the sensor data acquisition task, another objective is to achieve energy efficiency, which can be energy consumption used to minimise the number of samples obtained from the sensor or communication cost to communicate the sensed value to the base station. Researchers have proposed various solutions to tackle the energy consumption issue but are still working to achieve the most powerful result. In respect to the communication cost issue, again there has been much research, but optimal solutions have yet to be found.

According to the state-of-the-art survey by Jabeen & Nawaz (2015), data acquisition approaches are further detailed and categorised into different types as below:

- Classical approaches: which allow the user to retrieve the data they are interested in by posting declarative queries in languages such as Structure Query Language (SQL) or Object Query Language (OQL).
- Distributed approaches: the query plan needs to be generated by the query optimizer that specifies the sensors to participate in the query execution plan, the optimizer needs to allocate and schedule plan to specific sensor nodes.
- Adaptive query processing approaches: this approach is also used to handle network dynamics such as resource limitations, large amount of data streams, and data sending time based on memory consumption and energy usage.
- Stream query processing approaches: in a WSN, each sensing device on a node can generate a data stream into tuples for specific time interval and sampling period. This stream represents real world events, this data is continuously pushed to the query processor.
- Time synchronization approaches: this is very critical requirement for correlation among all nodes during in-network processing. This will ensure timely completion of processing any query execution plan that should be highly accurate.

Moreover, by 2016 a research survey by V. Jindal, et al, (2016) states that the query based approach is the most widely accepted approaches for data extraction. In addition, they also categories the agent based approach and the macro-programming based approach. They discussed also the constraints of limited energy and bandwidth data optimisation and transmission optimisation are the main area of research. Data processing is one of the primary technique that need advanced and higher motivated for future WSN economical consideration and technological capabilities to play significant role in data optimisation.
2.2.2 Distributed Query Processing

In the field of sensor networks, multiple data streams are generated from remote sources with the help of web usage logs and internet traffic analysis, which means a completely sensing system interface platform to support the user requirement from the sensor networks fields. Communication costs are reduced through distributed optimisation strategies that re-order query operators across locations and execute simple query functions locally at a sensor or network router (Gauger, 2010; Amato, et al., 2010). For instance, if each node pre-aggregates its consequences by distribution to the central node, the manager may then take the increasing sum and growing count and calculate the total average. A similar technique consists of the process of giving updates to the central node in the event that new data values obtained are different from those already present (Zheng & Jamalipour, 2009; Clements-Croome, 2004). The different data processing approaches that have been proposed in this regard are presented in figure 2-1.

![Data Processing Approaches Diagram](image)

**Figure 2-1 Data Processing Approach**

While on the other hand, some of the techniques of optimisation have been developed particularly for ad-hoc wireless sensor networks with the aim to minimise the cost of
communication along with the extension of battery life and contract with pitiable wireless connectivity. However, all these practices make use of the fact that dissemination of query and collection of the result in a wireless sensor network continue along a routing tree through a common wireless channel (Tahir, 2008; Wu, 2005). For instance, this could be evaluated from the fact that if a sensor heard its maximum local value as x in answer to a MAX query, the response might be achieved from an adjacent sensor that overhears this communication. This might happen in the case local maximum is smaller than x. In addition to this, terminated copies of its maximum values can be announced through the sensor in relation to all the neighbors to reduce the likelihoods of packet loss (Anastasi, et al., 2009; Phoha, et al., 2006).

From another survey by Raut, Gaharwar, Jade, Bais, & Virsen, (2018), they stated that the data aggregation is one of the best methods to reduce amount of readings to sense to the base station. Moreover, it is the procedure of either one or more sensors at any point to gather the readings from the network architecture. The network comprises of three sorts of hubs names straightforward hubs, aggregator hub, and queried hub.

In the context of big data, from another survey paper (Cao, Liu, Cheng, & Shen, 2018), saving energy is a critical demand. The key problem for wireless sensor networks energy efficient is the tradeoff between energy consumption and achieved performance. The data acquisition is also one of the biggest challenging tasks in energy-efficient wireless networks especially in more dynamic networks architecture. To support the big data in WSNs, coverage and connectivity should be ensured at the planning stage or system initialization process.

2.3 Queries and Query Optimisation

Some optimisation techniques have been developed particularly for ad-hoc wireless sensor networks with the aim of minimising the cost of communication and extending the battery life and minimising the use of poor wireless connectivity. Such techniques make use of the fact that dissemination of queries and collection of the results in a wireless sensor network continue along a routing tree through a common wireless channel (Tahir, 2008; Wu, 2005). For instance, this could be evaluated from the fact that if a sensor returned a maximum local value of x in answer to a MAX query, the response might be achieved from an adjacent or neighbor sensor that overhears this communication, because of the memory it is sharing the resources among other neighbors sensors.
A Sensor Network Query Processor (SNQP), also called SensorDB, is a user-friendly interface for programming and running applications that translate instructions from a declarative programming language with high-level instructions to the low-level instructions understood by the operating system. Indeed, the basic idea of SNQP is the addition of a layer modelling the WSN as a distributed database searchable by a query language similar to SQL (Gehrke & Madden, 2004).

Centralised querying has been the common mode of querying in WSNs. For this mode of operation, the base station acts as the point where the query is introduced, and the results gathered. In the query processing architecture, queries are entered at the server by users in a simple SQL-like language. This language describes the data that needs to be collected and the way in which it could be gathered, combined, changed and summarised. The SQL variant that is used for this purpose is different from the traditional SQL in the sense that in this process the queries are periodic and continuous (Sohraby, et al., 2007; Amato, et al., 2010). According to this, an interest is recorded by users in different types of sensor readings, and the results are issued by the system to the user. Each of the periods in which consequences are created is an epoch. The sample period or the epoch duration of a query is known as the amount of time that is recorded in different succeeding sampling models.

2.3.1 Ad-hoc Queries

Ad-Hoc Queries are queries that cannot be determined or defined before the moment the query is triggered. They are created to retrieve information when the need arises for any end-users and are comprised of dynamically constructed SQL languages. As defined by Sanghun et al. (2008) in their research, continuous queries are not the best practice for reducing the number of messages, whereas ad-hoc queries can reduce communication and energy overheads, thereby achieving more energy efficient WSNs (Sanghun, Haengrae, & Xingcheng, 2008).

From a paper by Reina et al., (2015), the ad-hoc networks is based on defining a multihop communication route between two or numerous nodes in the network. The static nature on the ad hoc network has made it very challenging to sense data at the communication stage. The ad hoc networks approach has been anticipated as an appealing communication technology to deal with the unexpected conditions emerging during and/or after the occurrence of any unexpected events or disasters. A real-time experimentation is recommended by the
author to tackle the available challenges of static nature, as well as more to the simulation phase for any communications components.

2.4 Architecture of a Wireless Sensor Network Query Processor

Sensor networks provide a challenging computing and programming environment. These devices are prone to failure resulting from their limitations as well as small and therefore their operating system is not able to mitigate failures. Few of the LEDs on the devices help in conducting debugging. Moreover, the programs are extremely dispersed, and it is necessary for them to manage radio and energy bandwidth when distributing information and processing (Zheng & Jamalipour, 2009). The query processing architecture is composed of six processes blocks, each block has its own functionality, as listed below and presented in Figure 2-2:

The query processing components are processed in order. Once the user selects the query to the query sensing interface, the parser analyses it and type checking is performed. This results in some internal illustration, such as abstract syntax tree that is passed to the query optimizer. After that, the query optimizer aims to select the appropriate query evaluation plan among various plans. The plan calculates different execution cost while it must make sure the best and precisely how the query is to be executed, which can be represented in a form of tree using algebraic operators such as selections, projections, and joins. At this point, the optimizer must associate a cost with a query execution plan using mathematical model which is the cost function. Then the logical optimizer performs optimisations which includes the table size, fragment, existence, whereas the physical stage will generate the physical query plan from the optimized query logical plan. The optimizer maps each logical plan with the physical plan. The above choices depend on various factors such as available memory, relation size, stored or not. Last stage is for the optimizer to allocate and schedule plan fragments to each specific node and then send the result of the query to the end user (Jabeen & Nawaz, 2015).

![Diagram of Query Processing Architecture](image)
This challenging computing environment and the associated device limitations mean that wireless data collection systems have an unusual set of software requirements. Some of these are as follows:

- Resources such as power must be managed carefully. Power consumption is dominated by the use of sensing and communication according to the data operations and data size difficulties inherent in sensor networks. Moreover, Moore’s law states that the energy charge per CPU cycle would decrease as the transistors and voltage reduce. While on the other hand, the physical restrictions associated with battery technology mean that the energy required to communicate information by radio will continue to outstrip the energy density of the batteries (Amato, et al., 2010; Yick, et al., 2008).

- The transient nature of sensor networks must be managed effectively by query processing as nodes come and go, signal strengths present between the devices fluctuate, batteries run out and interference patterns change. Despite all these challenges, the process of data collection should be disrupted as little as possible.

- Users need to be supplied with the tools needed to manage and understand the status of a network in order easily to add nodes with new types of sensors and capabilities (Canete, et al., 2013; Chand, 2007).

2.5 Conclusion

This chapter has reviewed the background of this thesis. It has discussed the different in-network query processing techniques. It introduced the different query models that can be used for different purposes. The chapter also presented the architecture of wireless sensor network query processors and concluded by surveying the different query optimization techniques.
Chapter 3 Literature Review

This chapter reviews the concepts of in-network data sampling techniques and data processing. The chapter surveys the literature on the various current sampling, distributed query processing and query-based optimisation techniques. The primary challenges related to data processing are discussed in the context of seeking to minimise the limitations of Wireless Sensor Networks, especially with respect to their energy consumption.

3.1 Introduction

Data gathering is a key operation for wireless sensor networks (WSNs), and includes data collection with aggregation and data collection without aggregation. Data collection without aggregation is referred to simply as ‘data collection’ (Ji et al., 2014). Over last decade, many techniques for these two applications have been proposed with different motivations, such as accuracy, reliability and time complexity.

A survey by Mukherjee & Mukherjee (2013) on the different approaches for energy conservation in wireless sensor networks revealed the common techniques available for energy conservation in sensor networks, such as data cycling, data-driven and mobility-based approaches. Details are presented below in Figure 3-1.

The highlighted part from the Figure 3-1 is the focused part on in this research, namely the data-driven approach. This is related to minimising the amount of sampled data and keeping the sensing accuracy within an acceptable range. Different factors can be considered in this approach, such as unwanted samples. There are three categories presented in this diagram related to data reduction. The first one is in-network processing, which performs the function of aggregations between the source and the sink. The second classification is data compression. This scheme is more suitable for a WSN because is used to reduce the amount of data sent to the node. The last type is the data prediction category, which works by reducing the number of packets of information sent by the source nodes to the base station, minimising the energy required for communication.
3.2 Survey of Sampling Techniques in Wireless Sensor Networks

In summary, in this thesis we will emphasize the data-driven approach, as it is closer to data collection and generation. The discussion in the next sections will present the three data-driven approaches, which are the data reduction approach and energy efficient data acquisition

3.2.1 Data reduction techniques

This model is capable of predicting the value sensed by sensor nodes both at the sensor and at the sink. If the predicted values are within the range of a specified accuracy, then the data are requested by the users and then evaluated at the sink node, without the need to get the raw data from the nodes (Heinzelman, Chandrakasan, & Balakrishnan, 2000).

According to Anastasi et al. (2009), data prediction schemes can be split into three main approaches: stochastic approaches, time series approaches and algorithmic approaches. Stochastic approaches exploit a characteristic of arbitrary processes so that a probabilistic model can be used to predict sensed values. A time series is a typical method of representing the time, and it can be used by Moving Average (MA), Auto-Regressive (AR) or Auto
Regressive Moving Average (ARMA) models (Gupta et al., 2011). These models are reasonably easy to use in many practical circumstances, with respectable accuracy. More sophisticated models have also been developed such as ARIMA and many others, but their complexity does not make them suitable for wireless sensor networks. The time series approach will be discussed in a later section. Anastasi et al. have grouped many other prediction models within the overall categorisation of algorithmic approaches, wherein algorithms are used to get predictions for a set of values. This approach is more application specific and can be considered to be a way of improving the results gained from one application using the different algorithms in association with other application’s findings by reducing sensing to save as much energy as possible. In our research, for data prediction, we will be using different time series forecasting techniques, as this shows more accurate results from previous research such as Manik Gupta et al. (Gupta et al., 2011). Furthermore, we will compare this approach with other approaches, which will be mentioned in the coming sections.

As shown by the energy conservation categories in Figure 3-1, there are a set of reduction techniques which many authors have developed, as we will be discussing below. These are stochastic approaches; time series approaches and algorithmic approaches.

3.2.1.1 Stochastic Approaches

E-Sense is a stochastic scheduling algorithm that combines the probabilistic data stream prediction with a module for data quality to produce a sampling schedule for each sensor node. In this approach, data measurements are made only when a state change is happening. Moreover, for the missing data, they applied false misses or false hits for missed state changes when they occurred. There are many similarities between e-Sense and our proposed algorithm such as the way it works to sample the data by using a data quality module to express the tolerance level. Ultimately, e-Sense is a mechanism used to save energy, but more improvements are required to gain better outcomes in relation to the objectives stated (Liu et al. 2006)

3.2.1.2 Time Series Forecasting Approaches

In some papers, for instance, that by Gupta et al. (2011), the research work was on monitoring pollution levels due to car exhaust gases. They implemented their work using novel sampling algorithms for the design and evaluation of the datasets. They used the EDSAS
(Exponential Double Smoothing based Adaptive Sampling) algorithm, which is considered one of the time series techniques. The EDSAS were deployed as a data reduction technique, based on prediction for an irregular sampled time series known as, Wright’s extension to Holt’s method. It also incorporates the use of EWMA (Exponential Weighted Moving Average). Chong & Gaber (2011) present the current approaches to reducing the amount of data communicated in-network. These techniques include compression, approximation and prediction. With regards to prediction techniques, there is the Markov model which is used to calculate the probabilities of the required data falling into different time series intervals. At the same time, the algorithms approach is also used. In many research papers, this is referred to as the parametric Expectation-Maximisation (EM) algorithm, which is a clustering algorithm. In addition, a classification model and a prediction model can also be deployed to report any data point of interest. Prediction based monitoring (PREMON) can be used in WSNs to predict the spatiotemporal correlation models. The Barbie-Q system called BBQ system proposed a data acquisition technique based on a time-varying multivariate Gaussian probabilistic model (Aggarwal, 2013), another approach that employs kernel linear regression in order to model the sensed values by capturing the spatial-temporal correlations (Guestrin, Bodik, Thibaux, Paskin, & Madden, 2004).

Yet another paper by Tulone & Madden (2006b) presents a way to collect large amounts of high-fidelity information about a remote location. They rely on an autoregressive model built at each sensor to predict the local readings. They focus on improving the performance of these data collection applications using a probabilistic approach (PAQ) by employing a statistical technique known as time series forecasting. This is generally used to detect progress over time and uses the recent history of readings to predict the expected future values. They use the (AR) model because it is computationally tractable on modern generation sensor networks.

Many works focus on querying, clustering or gathering issues. One particular work by Tulone & Madden used a time series model and approximated the times for which reason we believe building a prediction model is more accurate. By Tulone & Madden in another paper, (Tulone & Madden, 2006) propose a SAF (Similarity-based Adaptive Framework) that uses a simple time series forecasting models to predict sensor readings. Their approach provides a mechanism to detect data similarities between nodes and thence organise nodes into clusters at the sink, at no additional communication cost. In their approach, queries are answered using lightweight linear time series models, built by each node from a small number of readings and
stored at the sink. This approach works to reduce the amount of communication between the nodes and the sink. Their experiments prove that this is an energy-efficient approach.

The research proposed by Gupta et al. (2011) is based on both temporal and spatial data reduction methods, both of which have shown good performance across various experiments. The researchers designed an algorithm called the Exponential Double Smoothing based Adaptive Sampling (EDSAS) It is a further extension of Wright’s extension, which is an extension of Exponential Double Smoothing (EDS). Since the datasets used in the experiment by Gupta et al. are temporal and have linear trends, time series forecasting was used to predict changes so as to avoid unnecessary sampling. An introduction to the main mathematical concepts of time series, EDS and so on will be provided in later sections. The forecasts of EDS are based on the current level and trends in the time series and are generated by simple recursive equations. The data are sampled at regular time intervals; consequently, sampling a lot of data during periods of significant change would cause wastage. Wright’s extension, therefore, forecasts at irregular time intervals. A change detection mechanism is also included in their research, which we will cover in chapter 4. From their research papers, we can sum up that the nodes with higher predictability are sampled at lower sampling intervals compared to those with lower predictabilities, which can be sampled at higher sampling intervals. They also used an ASAP (Adaptive Sampling Algorithm Protocol) for their spatial sampling algorithms.

Since data reduction techniques are very important, many have been proposed. Abdel-Ali and Ramadan (2011) investigate new approaches to data reduction in single and multimodal WSNs. The proposed approaches are based on exponential smoothing predictors due to their simplicity; they require less memory and offer high accuracy. The mean squared error (MSE) for temperature values is calculated to specify the two smoothing parameters (α=0.6, β=0.4) because using smoothing parameter they produce better MSE. The experiment showed that it was possible to increase the lifetime hundreds-fold when the threshold is large.

3.2.1.3 Algorithmic Approach

Table 3-1 lists the different algorithms used for different technologies, comparing these according to the algorithm name; technique type; events detected, and the energy saved.

The first row in Table 3-1 lists the e-sampling technique used for high-frequency events. Bhuiyan et al. (2013) proposed a novel scheme termed “event-sensitive” adaptive sampling and low-cost monitoring. E-sampling works by reducing the resource usage of WSNs on two sides,
they are sampling rate and weight of frequency, which means the total frequent continues sensing, in a decentralised manner. Jain & Chang (2004) used a novel adaptive sampling technique. This study showed that e-sampling can save up to 87% of energy use compared to previous research in the same field. The e-sampling is also an optimal estimation scheme for assessing data arrival features, which is based on the (KF) Kalman Filter estimation of errors. This deals with the new samples to be collected and is similar to our approach: i.e. when the desired sampling rate violates, or the sample expired, a specified range, a new sample rate is requested from the server. In a research by RM Willett et al. (2004), they develop efficient algorithms to reduce the request messages between the server and the source, although they recognised that the system performance needed more testing on real-life data sets.

Another adaptive sampling scenario, presented by Cheng et al. (2010), as they all focused on the efficiency of energy consumption. Their work proposed a novel approach with a new matrix using EDCA (Efficient Data Collection Approach) to lower the sampling rate and to make sure fewer packets are transmitted. This matrix makes use of the benefits of spatial and temporal correlation in WSNs to reduce energy wastage by choosing the node and the time at which to sample data arbitrarily. The scheme has a decentralised base which directly forwards the data to the sink; this node is called a self-organised node. Their algorithm was able to reduce energy consumption significantly, and thus extend the lifetime of sensor networks.
<table>
<thead>
<tr>
<th>Technology Name</th>
<th>Algorithm Used</th>
<th>Parameters (ci, K, T, n)</th>
<th>Technique Type</th>
<th>Event Detected</th>
<th>Reference</th>
<th>Energy saves/consume</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-sampling</td>
<td>Event-sensitive adaptive sampling and low-cost monitoring scheme</td>
<td>Ci=0.5</td>
<td>Data reduction</td>
<td>Radio frequency</td>
<td>(Bhuiyan et al., 2013)</td>
<td>55% - 70%</td>
</tr>
<tr>
<td>Multi-sensor</td>
<td>Extended Kalman filter</td>
<td>Sample interval 0.1:0.2:0.3:0.4</td>
<td>Adaptive Sampling</td>
<td>Predicting tracking accuracy</td>
<td>(Lin, Xiao, Lewis, &amp; Xie, 2009)</td>
<td>67.47% saving</td>
</tr>
<tr>
<td>Flood Net</td>
<td>Decentralised Adaptive algorithm</td>
<td>CI (confidence interval = 95%)</td>
<td>Adaptive Sampling</td>
<td>Provide early warning of flooding</td>
<td>(Kho, Rogers, &amp; Jennings, 2009)</td>
<td>-</td>
</tr>
<tr>
<td>SHM system</td>
<td>Hierarchical decentralised QAS algorithm</td>
<td>-</td>
<td>Adaptive Sampling</td>
<td>Damage identification and localisation</td>
<td>(Hackmann &amp; Dyke, 2014)</td>
<td>78.9% energy saving</td>
</tr>
<tr>
<td>LMAC</td>
<td>MAC Protocol</td>
<td>93% sampling reduction</td>
<td>Data reduction</td>
<td>Environmental monitoring application</td>
<td>(Chatterjea &amp; Havinga, 2008)</td>
<td>Up to 87%</td>
</tr>
<tr>
<td>Entropy Sampling</td>
<td>Non-linear programming, Masked spectral bound</td>
<td>K=5, n=3600,</td>
<td>Adaptive Sampling</td>
<td>Monitoring design context</td>
<td>(Burer &amp; Lee, 2007)</td>
<td>-</td>
</tr>
<tr>
<td>Back casting</td>
<td>preview step &amp; refinement step MSE (mean Square Error)</td>
<td>n=10,000 sensors</td>
<td>Adaptive Sampling</td>
<td>To provide a constant estimate of the field being sensed</td>
<td>(Willett et al., 2004)</td>
<td>68.7% saving energy</td>
</tr>
<tr>
<td>USAC</td>
<td>Bayesian Linear Model</td>
<td>Ci=95%</td>
<td>Adaptive Sampling</td>
<td>Collect environmental data related to glaciers</td>
<td>(Padhy, Dash, Martinez, &amp; Jennings, 2010)</td>
<td>79% efficiency</td>
</tr>
<tr>
<td>SORA</td>
<td>Market-oriented programming</td>
<td>ε =0.05, α =0.2</td>
<td>Data reduction</td>
<td>the virtual market, nodes sell goods in response to global price information</td>
<td>(Mainland, Parkes, &amp; Welsh, 2005)</td>
<td>66%</td>
</tr>
<tr>
<td>Kalman-Filter</td>
<td>KF mathematical formulation and prediction</td>
<td>Sampling interval</td>
<td>Data reduction</td>
<td>Distribute the bandwidth automatically</td>
<td>(Jain &amp; Chang, 2004)</td>
<td>Minimise fractional error</td>
</tr>
<tr>
<td>e-Sense</td>
<td>Stochastic Scheduling Algorithm</td>
<td>False misses, false hits</td>
<td>Adaptive Sampling</td>
<td>Predict the likelihood of an event occurring in the future</td>
<td>(Liu et al., 2006)</td>
<td>Less than 55%</td>
</tr>
<tr>
<td>EDSAS</td>
<td>Adaptive spatial, temporal sampling</td>
<td>δ = 0.10, α = 0.50, β = 0.90</td>
<td>Adaptive Sampling</td>
<td>Reduce the number of samples while not losing data</td>
<td>(Gupta et al., 2011)</td>
<td>Energy saving (50%-70%)</td>
</tr>
</tbody>
</table>

Table 3-1  Summary of some data reduction and adaptive sampling used techniques in WSN
3.2.2 Data Acquisition Techniques

One type of data-driven approach (the bottom of the two in the figure) is energy efficient data acquisition. This type consists of three different approaches. These are the adaptive sampling approach, which we are concentrating on in this research project as well as the hierarchical approach, and the last is the model-driven approach.

3.2.2.1 Adaptive Sensing Techniques

As sensing is one of the mostly used strategies to gather readings from sensor devices, it is important to note down some important techniques related to sensing. From the various sampling techniques, we thought of focusing on the more relevant techniques that could have been used in this thesis but have some drawback to be highly performed. Moving on to the different types of adaptive sensing techniques, as adaptive sampling is one of the most energy-saving strategies, we can classify them into three categories. These are adaptive sampling, hierarchical sampling and model-based active sensing. Adaptive sampling techniques exploit the correlations within the sensed data and the information related to the available energy to adjust the sampling rate dynamically. These can be divided into two main categories; they are activity-driven adaptive sampling and harvest-aware adaptive sampling (see Figure 3-2).

![Different types of adaptive sensing](image-url)
Activity-driven adaptive sampling reduces the amount of data to be sampled through both temporal and spatial approaches. With temporal correlation, if the quantity of interest evolves slowly with time, then the next samples will not change violently, and the sampling rate will be reduced. With spatial correlation, if the sensor nodes are spatially close, the measurements taken from these sensor nodes will not change significantly, again meaning that the sampling rate can be reduced. Harvesting-aware adaptive sensing can also dramatically change the sensing rate. The sensor node used in harvesting-aware adaptive sampling can harvest energy from the surrounding environment by applying specified methodology, therefore adjusting sampling based on the available energy.

Another study by Masoum et al. (2013) aimed to ensure a certain level of data quality as well as energy efficiency. They concentrated on values falling out of the specified range, applying a mechanism which uses spatial-temporal correlation among the sensor nodes to determine which nodes to sample and how often the data should be sensed. The main idea behind their approach is to select a changing subset of sensor nodes to sample and transfer their data. They did this by using a scheduling procedure and finding the nodes of interest, those to be sampled, where values were recorded as having changed and being out of range. The authors implemented a temporal correlation based adaptive sampling algorithm and also forced sampling, which means that only the sampling nodes report their data. Similarly, Gedik et al. (2007) introduced an adaptive sampling that sends its measurements to the cluster head. The non-sampled nodes only collect data during forced sampling periods. Chatterjea & Havinga (2008), meanwhile, used time series forecasting methods to predict the sampling period and transmission rate. Moreover, this algorithm works by the node transmitting a reading to the sink whenever its current value differs from the previous transmission. This is similar to the scheme in the algorithm we propose. In another article (Deligiannakis & Kotidis, 2008), we observed that the authors used temporal correlations as a linear function together with a predefined confidence threshold to find a suitable sampling frequency.

Hierarchical sensing strategies assume that there are different kinds of sensors in WSNs: simple ones and advanced ones. They observe the same event and power consumption but provide different resolutions. Simple sensors are energy efficient but provide low-resolution readings or trigger an event, while in contrast advanced sensor nodes provide more accurate readings but consume more energy. At the same time, the more advanced, higher-level sensor
nodes can be activated to upgrade the lower resolution readings and make them more accurate. The central idea of hierarchical sensing approaches is to dynamically select which of the available sensor nodes must be activated, in order to maintain the readings’ accuracy, saving energy overall. Hierarchical sensing can be divided into triggered sensing and multi-scale sensing (Alippi, Anastasi, Di Francesco, & Roveri, 2009).

Further relevant research has been done by Borgne et al. (2007). They implemented a type of adaptive sampling called the adaptive model selection algorithm. This can be used among a set of different models such as DPS (Dual Prediction Scheme model), CM (Constant Prediction Model), AR (Auto-Regressive time series model), and AMS (Adaptive Model Selection), all of which seek to select the best performance for data prediction. Borgne et al. believed that the CM and AR models have higher performance because they are work in online mode without the need to store the datasets collected. When they set the accuracy threshold to 0.05, the model results in 20% of the data being sent to the sink, whereas the default monitoring system was not as accurate as this (Borgne et al. 2007).

Hernandez et al. (2001) proposed two adaptive sampling methods based on linear and fuzzy prediction, and they compared them with conventional sampling methods by running a different simulation using the internet and video conference traffic patterns. Their work resulted in high accuracy and better performance using both approaches as well as systematic sampling intervals as they retain the same sample interval. They suggested, however, that the fuzzy logic adaptive technique give more flexibility and better performance than LP linear prediction.

**3.2.2.2 Sampling Scheme Types**

Sampath (2005) summarised the different possible sampling schemes. Equal probability sampling is one of the simplest and oldest methods. For this sampling method he used a linear (LLS) trend and an autocorrelation model for their estimations. Unequal probability sampling, meanwhile, uses simple random sampling and systematic sampling. It is called probability proportional, and this type has different methods. Another sampling scheme is stratified sampling, which depends on dividing the population into groups called strata from which sample designs are drawn. Moving on, there is another type called two-phase sampling, which is used for getting the ratio as one phase and product estimations as another phase by choosing
two sample designs. On the other hand, a double sampling scheme can be used to combine different estimator methods for the population into two sampling phases using one scheme type, as mentioned above. Multistage sampling is a two-stage sampling approach with either simple random sampling or unequal probability sampling used in both stages. Sampath stated that the last type is the type with non-sampling errors, i.e. with no response as its data is incomplete (Sampath, 2005). As discussed above, many research projects on environmental monitoring applications have used a variety of different sampling schemes.

Additionally, research by Lin et al. (2009), proposed an adaptive, energy-efficient multisensory scheduling scheme for collaborative target tracking in WSNs. In this research, they calculated the optimal sampling interval to fulfil requirements for predicted tracking accuracy. The authors developed a novel distributed adaptive multisensory scheme based on an extended Kalman Filter. Their scheme is divided into three steps. First, the sampling interval is determined based on the accuracy threshold. Second, a number of sensors are selected to form a short-term tasking cluster. Third, the cluster head is selected according to the predicted energy consumption. They use one-step-ahead prediction in their algorithm, which they assumed to be ideal in terms of more energy saving. The proposed scheme can achieve a much better trade-off between tracking accuracy and energy consumption.

The KF was introduced in 1960 as a recursive solution to the discrete-data linear filtering problem (Jain & Chang, 2004). In their research, the model allows the real-time collection of water depth data to update flood predictions regularly with refreshed water levels. When the model-based probability for the water level exceeds a threshold is less than 5%, the requirement for data transmission from a node is lowered, and otherwise, the requirement for data transmission is raised. The degree of data transmission is related to the importance factor; for example, data with high sampling rates are more important and are associated with critical zones. Upon each iteration of the model, the network changes its behaviour, altering the reporting rate, which is derived from the data's importance using a conversion function for each individual node specified by environmental experts according to the data importance placed by the predictor model.

Another sampling technique is applied to a flood warning system called FloodNet. It collects samples from each sensor by using a centralised flood predictor model, which is used to determine the priorities. The model comprises a random one-dimensional numerical water
power model coupled to an ensemble Kalman Filter (KF) (Kho et al., 2009). Adaptive sampling is proposed by Jain & Chang (2004). It is a statistical protocol in which each node adapts to the characteristics of the streaming data. In this method, the nodes autonomously decide their sampling rates within a given range, using a Kalman Filter (KF) to estimate the error. When the sampling rate violates the range, a new sampling rate is requested from the base station. The base station then determines new sampling rates, taking into account the constraints of the available resources, so, that the KF-estimated error over all the active nodes is minimised. In this approach, the sensors transmit their sensed data to the base station. This is, however, neither scalable nor suitable for distributed implementation, because of the computational complexity of the KF (Jain & Chang, 2004).

### 3.2.2.5 Model Driven Sampling

Model-based sampling techniques build a model of the sensed data and then use this model to predict the next data instead of sampling a large quantity of data. This allows them to save the energy used for data sensing. One example of a model-based adaptive sensing algorithm is e-Sense, proposed by Liu et al. (2006). Using a biased random walk approach, the e-Sense algorithm predicts the probability of an event happening at various time steps into the future. Furthermore, based on the calculated probability, it determines a sampling probability for the gathered instant from the specified sink (Gupta, 2013). Some other authors, such as Gedik et al. (2007), have used Adaptive Sampling Algorithm Protocol (ASAP), and have found that it gives a good sampling performance. Accordingly, Alippi et al., (2010) developed an algorithm for adaptive sampling using an adaptive CUMulative SUM (CUSUM)-based test for signal change detection. This is based on the cumulative sum change detection technique.

Furquim et al. (2014) conducted an analysis of the use of data gathered from urban rivers to forecast future flooding, with a vision of reducing the damage they cause. Their research was able to handle the data collected and provide a better modelling of behaviour making it possible to forecast any future disasters. Many other authors have covered this field of adaptive sampling but with different purposes as well as different implementation plans, for example, Batalin et al. (2004) and Goel & Imielinski (2001).

Our contribution discussed in chapter 4 is to design a data reduction technique to reduce data samples by implementing a time series forecasting approach based on Exponential Double
Smoothing techniques for the temporal purpose of our data (temperature readings), as well as integrating the TCP congestion control idea into the step size mechanism. Furthermore, we will be specifying a threshold that limits our sensed data to only the readings that are outside the set threshold.

3.3 Survey of Probability-based Query Processing

Many researchers carried out work on query processing for various projects. Recently, a methodology presented in (Haroon Malik, Malik and Roy, 2011), who has a novel and interesting query optimisation technique that has to do with the sensory features. In the technique, the researchers signify that they have used principal component analysis (PCA) and a statistical method. One approach, as proposed by in (Zhong, 2007) that each sensor node samples the environment as specified by the query. According to the execution plan, sampled data is sent to the leader node, or together with partially aggregated data received from other nodes, and aggregation operators are applied. The volume of data is decreased by partial or incremental aggregation. After that, the responsibility of the leader node is to combine all the partially aggregated results and report it to the gateway node if the value exceeds the threshold. Another research by (Nehme et al., 2013) presents a query mesh multi-route query plans, which computes multiple routes using classifier model to assign the best route to process incoming tuples. They also proposed a self-routing fabric infrastructure that supports query execution with multiple plans. In (Shirsath et al., 2016), the authors proposed a method to answer the client query using a given network of data aggregator; for fixed queries, which can use the client queries to construct a network of data aggregator optimally. Their intention is to minimise the number of messages between the data source and client.

Another research regarding heuristic scheduling in (Yan et al., 2011), they propose a heuristic scheduling algorithm to solve the problem of multi-regional query scheduling. Also in (Sun et al., 2010) presented a multicast query based data dissemination protocol to reduce energy consumption further. It is found that there is a similarity between our work and the two-tier multiple query optimisations approach presented by Xiang and other in (Xiang, Lim, K.-L. Tan, et al., 2007), from the arguments that it illustrates the sensing system structure; it shows that the user request data from the synthetic queries instead of going to the sensor network immediately, the sensor database is always ready with new queries for updates. They have set a time interval to gather the data by designing a data aggregation model. However, queries,
particularly in TinyDB (Madden et al., 2005b), incorporates SELECT-FROM-WHERE clause for the purpose of collection, connection, forecast and combination. In this prospect, unambiguous sustenance provided for selection, windowing, and sub-queries through embodiment points. By TinyDB, sampling supported extensively. While in the queries, sensor data observed as the single simulated table with one column per sensor category. However, tuples attached by the structures to the table at distinct intermissions identified as query restrictions. From the literature, the optimum approach is considered as a two-tier model according to (Xiang, Lim, K.-L. Tan, et al., 2007) with additional aggregated and sensing dynamically to acquire quality data and further energy efficiency.

3.3.1 Approximation Techniques

In-network approximation-based techniques can help to reduce energy consumption within a wireless sensor network by using statistical or probabilistic models to reduce the amount of communication between nodes and therefore conserve energy. They can typically be used alongside the aggregation-based techniques discussed in the previous section. They can build a time-dependent multivariate Gaussian probabilistic model, inter-sensor and intra-sensor correlation and regression models to determine the likelihood of a given sensor holding a particular value at a particular moment in time. By querying the model created by the system, it is possible to choose to get data from only a small selection or sensor nodes (depending on the user-specified ‘confidence interval’) and thus reduce power consumption by up to an order of magnitude while still providing results that are reliable enough for many applications. As well as conserving energy, some approximation-based techniques can also compensate for other shortcomings of using wireless sensor networks as distributed databases. These potential issues include: Data misrepresentation in conventional wireless networks, sensors within range at any given time may not give a truly random sample of the population; therefore any data retrieved from the non-random sample may not be a true reflection of the real-world (Deshpande et al., 2004). Moreover, Unnecessary Calculations – unless otherwise specified, when collecting data from a wireless sensor network, one may sample from all available nodes. Using approximation-based techniques, however, can often result in suitably accurate results using many fewer samples (Deshpande et al., 2004)
Although many statistical or probabilistic models can be used, the framework in which they are used is generally as follows: a model of the data being recorded by the sensors is created using the sensor readings, the user sends out a probabilistic query which specifies the confidence interval that the user is prepared to accept for the answer, the model is queried and so is the sensor. If the value read from the sensor is close to the value predicted by the model (within a percentage specified by the user) then the value is not transmitted from the sensor node, and so energy has been saved. If, however, the value read by the sensor is outside the tolerance allowed by the user then the sensor reading is transmitted, and its value is used to update the model.

As a means of conveying the concept, the following simple example of three temperature sensors within a room can be considered. Within the imaginary scenario, sensor one was placed near to the window of the room and was generally two to three degrees warmer than the average temperature, sensor two was in the centre of the room and was generally close to the average temperature, while sensor three was located close to the air-conditioning unit and so was generally three to four degrees cooler than the average temperature. If the mean average temperature of the room were to be sensed using a wireless sensor network using the BBQ approach the user would send a query such as:

\[
\text{SELECT nodeId, Temp +/- 0.1oC, conf (0.95)}
\]

\[
\text{WHERE nodeId in \{1…3\}}
\]

The query processor would use its multivariate Gaussian models to calculate the number of actual sensor readings that would be necessary to provide a result that has a 95% probability of being correct. If, perhaps, the model found from previous measurements that temperature sensor two was correlated with measurements from sensors one and three, then the query processor may choose to only query sensor two and use this result to estimate the values of sensors one and three. The model may also have found a correlation between temperature sensor readings and the sensor node’s battery voltage meter. If this is the case, as often with sensor nodes, reading the battery’s voltage meter uses much less power than sensing using the temperature sensor, then the model may choose to query the voltage sensor on sensor two to conserve still more energy.

It is also worth noting that it is possible to be provided with the average temperature of the room with 100% confidence. To achieve this 100% confidence, however, every sensor
in the network would need to be queried and so in this scenario the only energy conserved would be if an in-network aggregation technique was used to gather the data, rather than using the BBQ approximation-based in-network data processing.

### 3.3.2 Aggregation Techniques

Aggregation is generally achieved through the use of aggregate functions in the Structured Query Language (SQL). For example, in a population sample, aggregation could present the average age of the populace, the total income of the populace, etc. In traditional database management systems, aggregation functions are used in the Structured Query Language (SQL) to group together values of multiple data rows to form a single value of more significant meaning or measurement. For example, in a traditional database table that contains the records of hourly temperature readings; obtaining the average temperature for a specific day would be achieved by using the following SQL query:

```
SELECT AVG (temp) FROM Temperature-Table
GROUP BY DAY
```

When used in in-network data processing, data aggregation could refer to the process whereby data which originates from different sources is combined (Fasolo, Rossi, Widmer, & Zorzi, 2007). The initial focus of in-network aggregation techniques was to provide ways of routing packets such that data from different sources that is headed towards a common destination could be combined, this provided an alternative to routing which differed from the traditional ad hoc routing protocols. In addition to the routing problem, recent studies have addressed mechanisms to represent and combine data in a more efficient manner (Fasolo et al., 2007). Aggregation techniques within in-network data processing aim to issue a query to the sensor nodes, obtain a response from the nodes after they have collected, filtered and aggregated the data according to the specification of the user query (Diallo, Rodrigues, Sene, & Lloret, 2015). The queries are similar to the SQL queries used in traditional databases but run continuously in order to capture changes that are recorded by the sensors. This is unlike the approach adopted by traditional database queries which are focused on returning results about the current state of the database. Some common aggregation functions used in SQL are summarised in table 3-2 below (w3schools. 2015):

37
Table 3-2 A description of some SQL aggregation functions

<table>
<thead>
<tr>
<th>Aggregate Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG()</td>
<td>Returns the average value</td>
</tr>
<tr>
<td>COUNT()</td>
<td>Returns the number of rows</td>
</tr>
<tr>
<td>MAX()</td>
<td>Returns the largest value</td>
</tr>
<tr>
<td>MIN()</td>
<td>Returns the smallest value</td>
</tr>
<tr>
<td>SUM()</td>
<td>Returns the sum</td>
</tr>
</tbody>
</table>

Studies have shown that the use of aggregation-based techniques in wireless sensor networks significantly contributes to the conservation of energy within the network, particularly when the number of nodes is large (Intanagonwiwat & Estrin, 2002). This presents a strong motivation for the use of these techniques in wireless sensor networks. Research efforts have led to the development of a number of proposals on aggregation-based techniques for in-network data processing. TinyDB (Madden et al., 2005), COUGAR (Fung, Sun, & Gehrke, 2002) and ADAGA (Brayner, Lopes, Meira, Vasconcelos, & Menezes, 2008a) are some projects that have demonstrated the implementation of aggregation-based techniques. The following subsections present a brief description of these aggregations-based techniques, as well as other techniques.

3.4 Survey of Query-Based Processing Techniques

Acquiring data in a wireless sensor network can become constrained by limited energy. In solving the energy issues confronting the acquisition of data in the wireless network, various proposals, algorithms and systems have been implemented to reduce the amount of energy consumption and increase the lifetime of wireless sensors, which is a very active research topic, as detailed in(Wang, Chen and Papadias, 2013).

In literature, ongoing work is being done for partial in-network query processing, which needs more attention from researchers. The query could be in a text or declarative format. Existing query systems include TinyQP (Mo et al., 2013), Cougar (Yao and Gehrke, 2002), GSN (Aberer, Hauswirth and Salehi, 2007), SenQ (Wood, Selavo and Stankovic, 2008) and AnduIN (Klan et al., 2011). TinyQP [8] is implemented as a query processing system based on TinyOS (Reusing, 2012). Compared to our approach which is based on ContikiOS, TinyOS provides
in-network complex queries which allow real-time query, historical query, event detection query and aggregation query. A webserver forwards a request to the application server which produces an optimized query scheme. This is forwarded to the sink node. The sink node broadcasts the query command to all nodes according to specifications in the query command. The sink node receives query results from the sensing nodes and returns it to the application server, which processes the data and saves it in the database. Cougar [9] supports aggregation and more complicated computation. A gateway node is responsible for query optimization, which generates and disseminates query plans to all intended nodes. A query proxy layer is added on each node which chooses a query leader responsible for computation. On the other hand, the GSN [10] concept is the virtual sensor, using stream engine to receive the data virtually. In SenQ [11] queries are issued from wearable sensors. The SenQ optimizes queries and reduces network overhead by making use of on-demand buffering and query caching. It is made of a high-level declarative language similar to SQL and TinyDB (Madden et al., 2005b) for external user queries. In AndulIN [12], they introduce a system for developing, deploying and running complex in-network queries by combining sensor local and data stream engine.

3.4.1 Push-based and Pull-Based Data Acquisition

This approach has been proposed for Cougar, TinyDB and TiNA, entailing the running of SQL-like query language commands to reduce the communication cost and thus prolong the network lifetime. The Cougar database started to investigate how database related techniques might be integrated with sensor networks in 2002. They intended to interrogate sensor networks through declarative database queries from a given a user query, utilising a query optimiser to generate an efficient query plan for in-network query processing which can reduce the resource usage and thus extend the lifetime of the sensor network (Yao & Gehrke, 2002). The TinyDB structure for addressing issues of when, where, and how often data is sampled and which data is delivered in distributed embedded sensing environments, is called Acquisition Query Processing (ACQP). TinyDB runs on a Berkeley mote platform, on top of the TinyOS operating system. TinyDB has many features of a traditional query processor (e.g. the ability to select, join, project and aggregate data). Madden et al. discuss an extension to SQL for controlling data acquisition and show how acquisitional issues influence query optimisation, dissemination and execution (S. Madden, Franklin, Hellerstein, & Hong, 2002; S. R. Madden et al., 2005). After that, TiNA was then proposed to offer more improvement over TinyDB.
Another approach, named CONstain Chaining (CONCH), has been proposed for data acquisition. The main idea is to provide effective spatiotemporal suppression and use a minimum spanning forest of the network for data transmission, based on tree topology. Due to the correlation, the readings of nearby sensors always share the same trend with negligible differences (Silberstein, Braynard, & Yang, 2006). In another paper, a system with advanced query processing techniques was designed; using Sun Small Programmable Object Technology (SPOT) sensor network platform. The system was programmed in Java. This system facilitates time triggered queries that are scheduled in a distributed fashion among sensor nodes (Scholz, Gaber, Dawborn, Khoury, & Tse, 2007).

An algorithm called Efficient Data Collection Aware of Spatio-Temporal Correlation (EAST) has also been proposed. This algorithm uses the shortest routes for forwarding the gathered data towards the sink node and totally exploits both temporal and spatial correlations to perform near real-time data collection in WSN. (Villas et al., 2013). Another algorithm, entitled Quality Guaranteed and Energy-Efficient (QGEE), has been proposed by Ren & Liang (2007) for distributed and heterogeneous Wireless Sensor Database Systems (WSDS). The goals of this approach are reducing disturbance from measurements with extreme error, decreasing information loss from link failure and minimising energy consumption. A probabilistic and confidence interval strategy is employed to attain the stated goals (Q Ren & Liang, 2007).

Finally, the Distributed Nested Loop Join Processing (DNLJP) algorithm was proposed by Vaidehi & Devi (2011) to query and access data generated by sensor nodes. The proposed scheme groups sensors based on geographic locations and RFID data query processing in a SQL-based STREAM Query language to reduce the query response time (Vaidehi & Devi, 2011).

3.4.2 Packet Merging

In a wireless sensor network, a query is commonly used for collecting periodic data from the objects being monitored. The amount of data across WSNs sensed by a query can significantly impact WSNs power consumption and its lifetime. SQL-based query languages, which consist of SELECT-FROM –WHERE clause, are commonly recognised and often used in specifying queries for a sensor network. Sensor networks have their own characteristics;
however, and therefore any required additions modifications must be made to the basic SQL query. To decrease the size of the data that is exchanged between the sensors nodes, various data reduction techniques need to be implemented in this framework, were packet merging is one of them. Packet merging is one technique for reducing the data reduction in addition to other such as compression is another technique as well as data aggregation and data fusion. The main idea here is very straightforward, a small amount of data consume less amount of power. (Malik, Malik and Roy, 2011).

3.4.3 Execution Query Cost

Several query execution costs exist, including the cost of storage, accessing the subordinate storage, memory utilisation, computation and the communication. The produced subordinate and primary catalogues along with the relation magnitudes are obtained from the database management system (DBMS). An evaluation of the statistical data concerning these relations helps approximate the cost of the query evaluation plan.

Robinson et al. (2014) propose a cost-based query optimisation through artificial intelligence planning. Under this technique, they argue that for queries that are relational, artificial intelligence programmed planning can be applied to the query optimisation issue. Most importantly, the technique suggested by Robinson et al. (2014) concentrates on join-order assortment. The focus is so far on queries that are conjunctive. The authors note that it is the responsibility of the join-order assortment to select the data to be utilised in the attainment of the data on the disk. The purpose of this is to ensure that the query organisation is planned in a manner that reduces the execution cost of the plan.

3.4.4 Join Query

A JOIN is a clause in SQL (Structured Query Language) that combines the record from multiple database tables, provided there are common values in a similar field amongst the table. A joint makes it possible to extract data from multiple tables with a single query, rather than writing one query for each table and trying to match the data manually. The capability of a join to match relations between multiple tables based on a condition of commonality among similar field values among the tables can be extended to the application of joins in Wireless Sensor Networks, as presented in Kang (2013), where the scenario of an application that monitors pH values (acidity levels) in a river ecosystem is described.
According to this example, the sample query below may be used to compare pH values from different nodes at different locations in the ecosystem without taking the readings separately.

```
SELECT S1.pH, S1.location, S2.pH, S2.location
FROM Sensors AS S1, Sensors AS S2
WHERE S1.pH < t
AND distance (S1.location, S2.location) < d
```

The use of such a query (join) eliminates the need to query the nodes multiple times in order to extract data for processing. This could serve as a way of saving energy since the number of transmissions would be reduced by combining multiple data requests. The key to implementing join type queries is to control that joins with the condition table and the sensor readings.

1. Parallel Join using Column-Oriented Database: this is a distributed algorithm for processing data in sensor networks based on recently proposed strategy for column-oriented databases where the data table is stored in columns (Deng, Kim, & Kim, 2013).

2. Yang et al. (2007) proposed Two-phase self-join (TPSJ) to evaluate self-join queries for event detection in sensor networks efficiently. The TPSJ scheme takes advantage of the properties of the events and carries out data filtering during in-network processing. While Yang et al. (2007) discuss TPSJ execution with one window; we extend it for continuous event monitoring. Their experimental evaluation results show that the TPSJ scheme is effective in reducing the number of radio transmissions during event detection.

3. An SNJ strategy is an in-network synopsis join (SNJ) strategy to reduce transmission costs by avoiding sending non-candidate tuples. The results show that synopsis joining outperforms centralised join schemes in terms of communication cost, especially for low join selectivities, thus prolonging the lifetime of the sensor network (Kang, 2013). Other techniques are also listed below.

### 3.5 Survey of Distributed Query Techniques

#### 3.5.1 TinyDB Project

In the TinyDB project, which was developed for sensor networks based on the TinyOS operating system; a distributed query processor was designed to run on each of the nodes in a sensor network. This query processor supports a SQL-like query language and has the
capability to process aggregation functions such as MIN, MAX, SUM, COUNT, AVERAGE (Madden et al., 2005). When a user requires information from the sensor nodes, they may issue a query through an interface, specifying the data which is required from the sensor network. The query processor decomposes the query and distributes it across the network. The sensor nodes collect the data, filter it, aggregate it and finally send a response to the user. The intermediate nodes that contain relevant information for the query perform aggregation on the sensor readings as they traverse the communication tree. The large quantity of data that would have been transmitted through the network is thus reduced by this in-network aggregation, thereby preventing the bottlenecks that would have occurred at the root node, and in effect increasing the lifetime of the network (Diallo et al., 2015). In figure 3-3 Diallo et al. (2013) illustrate the aggregation that takes place at intermediate nodes in TinyDB.

![In-network Query Processing](image)

**Figure 3-3** Aggregation steps of sensor readings in a sensor network Diallo et al. (2013)

TinyDB supports the specification of queries in a SQL-like query language which considers the sensor data to be records consisting of several fields which include information like id, location, timestamps, sensor readings, etc. The SQL keywords SELECT, FROM, WHERE and GROUP BY are supported by TinyDB. For example, in order to return the id of each sensor in the network alongside the values of light and temperature readings at intervals of 5 seconds for a period of 30 seconds, the following query may be written:

```
SELECT nodeid, light, temp
FROM sensors
SAMPLE PERIOD 5s FOR 30s
```
Diallo et al. (2013) also present an example of a query that returns aggregate values as follows:

```
SELECT AVG (volume), room FROM sensors
WHERE floor = 6
GROUP BY room
HAVING AVG (volume) > threshold
SAMPLE PERIOD 30s
```

In this example, the user aims to return all sensor locations whereby the average sound sensor reading exceeds a specified threshold value. This query delivers updates every 30 seconds until it is deregistered from the system by the user. TinyDB can support this kind of aggregation through the use of three functions: a merging function, f, an initialiser, i, and an evaluator, e.

**Cougar Project**

Another proposal for performing aggregated queries in sensor networks was presented in the Cougar project (Fung et al., 2002). According to Yao and Gehrke (2002), a clustered approach was used in this method, whereby the entire sensor network was grouped into several clusters, each of which was managed by a cluster head. The child nodes of each cluster send their readings to the cluster head on a periodic basis. The cluster head aggregates the readings which it receives and forwards the computed results to the Front End of the network, which is usually a query optimiser that is located at the gateway node. This query optimiser (gateway node) is responsible for receiving user queries, optimising them and computing the results based on the query parameters and the readings it has received from child nodes (Yao & Gehrke, 2002). An illustration of how aggregation works in the Cougar approach is presented in Figure 3-4.
3.5.2 ADAGA Project

ADAGA (Adaptive Aggregation Algorithm for sensor networks) is an aggregation algorithm that adapts its behaviour based on the availability of energy and memory, by using a dynamic approach to adjust the intervals between data collection and sending. In ADAGA, each sensor node aggregates its local data, as well as data it has received from other nodes and forwards it to the next hop (Brayner, Lopes, Meira, Vasconcelos, & Menezes, 2008b). This process is repeated as the aggregated values traverse the network until they make their way to the base station.

The ADAGA algorithm is a five-stage algorithm that consists of the following stages:

Stage 1: This stage controls the other four stages of the algorithm. Basically, a nested loop that runs \( i \) times (where \( i \) is the number of query executions)

Stage 2: Processes the temporary data which is stored in the processing area

Stage 3: Monitors the use of energy and memory, tailoring data collection to available resources

Stage 4: Stores locally sensed data and aggregates it with packets received from other nodes

Stage 5: Forwards packets to other nodes.
3.5.3 ADAGA-P

The ADaptive AGgregation Algorithm for sensor networks with data Prediction is proposed by Razzaque to introduce an in-network data prediction and aggregation operator into query processing in WSN (Razzaque et al., 2013). This is also based on a linear regression model. This approach also has the property of adjustable accuracy.

3.5.4 TiNA Project

TiNA (Temporal Coherency-Aware In-Network Aggregation) (Diallo et al., 2015, Sharaf & Beaver, 2003) presents an improvement over TinyDB. It introduces an additional clause, VALUES WITHIN tct (temporal coherence tolerance of the query), to the aggregation query used in TinyDB. While TinyDB transmits sensor readings at fixed intervals, TiNA transmits a sensor reading only if the difference between the current reading and the previous reading exceeds an acceptable tolerance value, tct. This value specifies the degree of tolerance of the user to changes in sensor readings. For example, a tct value of 5% indicates that only readings that differ from the previous reading by a margin of more than 5% should be forwarded. This approach reduces the size and number of data transmissions required in the network, thereby minimising energy consumption, although it inevitably introduces some lack of consistency in the data transmitted (Diallo et al., 2015). See Figure 3-5

![In-network aggregation query using TiNA (Diallo et al., 2015)](image-url)
3.5.5 SNEE

Galpin et al. (2008) have fully described the Sensor NEtwork Engine (SNEE) approach of query optimisation on a classical Distributed Query Processing (DQP). This framework met many compliments from the perspective of WSN as databases such as TinyDB. The framework is presented in the form of compiler or optimiser for a continuous declarative query language over the sensed data stream.

3.5.6 SenQ

SenQ is an embedded query system for interactive sensor networks (IWSNs) proposed by Wood, Selavo and Stankovic (2008). It stands for Sensor extensible Query system. They addressed the issue of heterogeneity, deployment dynamics, in-network monitoring, localised aggregation, and resource constraints. It has a layered system design, which supports only snapshot and streaming queries.

3.5.7 Corona

Corona is a distributed query processor implemented on the sunSPOT platform for WSN with full Java tools as an operating system. Corona provides a declarative query interface to users as well as sharing queries among several users. In addition, the platform also provides a technique for shifting query starting times to increase cache usage. Figure 3-6 illustrates the Corona system architecture and user interface. (Khoury, Dawborn, Gafurov, & Pink, 2010)

![Figure 3-6](image_url) Corona query processor on sun SPOT (Khoury et al., 2010)
Summary of Existing Techniques

As pointed out earlier, many distributed query techniques that generally reduces the number of transmissions in a sensor network, in effect, conserving energy within the network such as in-network aggregation. A trade-off is introduced, however, due to the potential delay in transmission at nearer nodes which have to wait for data from farther nodes to be aggregated before they, in turn, can perform aggregation and forward the data (Intanagonwiwat & Estrin, 2002). It is important to note that the amount of delay introduced in this way depends on the aggregation approach used. Table 3-3 summarised the existing distribution techniques in different factors.
<table>
<thead>
<tr>
<th>Factors</th>
<th>Cougar</th>
<th>TinyDB</th>
<th>TiNA</th>
<th>SNEE</th>
<th>ADAGA</th>
<th>RT-DBMS</th>
<th>TTQP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proposed approach</strong></td>
<td>Generate New Query Plan QP</td>
<td>Distributed Acquisitional Query Processing System</td>
<td>Temporal coherence tolerance</td>
<td>Stream Sensed data</td>
<td>Nested Loop</td>
<td>Real-Time Query Statistics</td>
<td>Time-Triggered Query Processing System</td>
</tr>
<tr>
<td><strong>Optimisation metrics</strong></td>
<td>Packet Merging for similar queries</td>
<td>Aggregation functions such as MIN, MAX, SUM, COUNT, AVERAGE</td>
<td>Declarative SQL like queries</td>
<td>Continuous Declarative Query Compiler</td>
<td>Aggregation Algorithm (dynamic approach)</td>
<td>Transmit time, logical, temporal consistency</td>
<td>SQL queries</td>
</tr>
<tr>
<td><strong>Model used</strong></td>
<td>In-network aggregation-based techniques</td>
<td>In-network aggregation query processing model</td>
<td>Classical distributed query processing (DQP)</td>
<td>Aggregation, Prediction model ADAGA-P</td>
<td>Probabilistic model</td>
<td>In-network processing &amp; aggregation</td>
<td></td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td>Simulation PDA class</td>
<td>Simulation Tiny OS</td>
<td>Simulation Tiny OS</td>
<td>Simulation Tiny OS</td>
<td>Simulation</td>
<td>Simulation</td>
<td>Simulation</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Data collector, aggregated and stored for offline querying and analysis</td>
<td>Collects data, filters it, aggregate it and finally send a response to the user.</td>
<td>Declarative SQL like queries</td>
<td>SNEE distributed query processing components with algorithm</td>
<td>SQL-like queries SNQL</td>
<td>Cloud-based distributed architecture</td>
<td>Synchronised merge operation, data compression method</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>(Yao &amp; Gehrke, 2002)</td>
<td>(S. Madden et al., 2002)</td>
<td>(Sharaf et al., 2003)</td>
<td>(Galpin et al., 2011)</td>
<td>(Brayner et al., 2008a)</td>
<td>(Diallo, Rodrigues, &amp; Sene, 2012)</td>
<td>(Scholz et al., 2007)</td>
</tr>
</tbody>
</table>
3.6 Survey of Query based Optimisation Techniques

Extensive research has been carried out on query processing for different applications. Recently, the authors of (Brahmi et al., 2015) addressed the problem of optimal stopping at the node level with a focus on scheduling the transmission of aggregated packets. Their method uses a Markov decision process and an optimisation policy based on a genetic algorithm. In research by Shili Xiang, Lim, K. L. Tan, et al., (2007), the authors aimed to schedule the queries considering the whole system. As a result, their algorithm dynamically determines the route to disseminate the query results and aggregates data involving as few nodes as possible. Moreover, they successfully acquired and transmitted the same data segments satisfying multiple queries, by using the broadcast nature of radio communications. In research by Nehme et al., (2013), a query mesh for multi-route query plans was presented. This approach selects the best among multiple routes using a classifier to process incoming queries. Furthermore, the authors propose a self-routing infrastructure that supports query execution following multiple plans. In research by Shirsath et al., (2016), the authors propose a method to execute and answer queries using an optimal network of data aggregators for fixed queries. This method aims to minimise the number of messages communicated between the data source and the final user. Optimal query allocation is also presented in (Mitici et al., 2015). The authors of this study propose continuous- and discrete-time Markov decision processes as a trade-off between communication delays and updated data. The processes are focused on the query cost. Other methods include a heuristic scheduling algorithm for multi-regional query scheduling (Yan et al., 2011) and a multicast query-based data dissemination protocol to reduce energy consumption (Sun et al., 2010).

As presented by Venkata (2014), there have been numerous research studies that have been undertaken regarding query optimisation techniques for queries using SQL in wireless sensor databases. As the author says, the foundational concept of these studies is to decrease the price of a single query. Due to this, varying query optimisation techniques are used to assess the best means of formulating a query that will bring down its corresponding price. The author opines all query optimisation techniques are used to cut down the query processing cost, which is fundamental because information around the assembly of the wireless network along with the interactions that exist in between and among the sensor nodes is needed. It has also been noted that query optimisation is important in the reduction of the amount of consumed power (Xu, et al., 2012).
3.6.1 Query-based data gathering model

Malik et al. (2011) presented a novel and interesting query optimisation technique that has to do with the sensory features. In this technique, the researchers signify that they have used principal component analysis (PCA) and a statistical method. Under the PCA, they assess the sensory data for any historical traces. In this way they are able spontaneously to recognise vital features that exist among those traces that are interlinked; hence, the overhead of the procedure is decreased. They do this in several stages. In the first stage, the SQL is fed into the system where the management of the data and optimisation is to occur. The principal component analysis algorithm is then used to optimise the query by mining the components that are considered to exhibit a high degree of variation from the historical data. Once this is done, the query that has been optimised is entered into the wireless sensor network in order to mine the sensory data. The researchers note that after this is done, the features that originally constituted the data can be obtained from those that have been optimised by reversing the principal component analysis procedure.

Xiang et al. (2007) suggested a technique they term as Two-Tier Multiple Query Optimisation (TTMQP). As its name suggests, this technique constitutes two tiers. Base station optimisation is the initial tier and takes account of costs. Using this methodology, a number of queries are rewritten into a group that is optimised on the basis of similarity while doing away with any excesses that may have been there in the initial queries. In-network optimisation is the following tier. In this tier, the authors state that the outcomes of the query are efficiently availed. The provision of the query outcomes is made by distributing the readings from the sensors in queries that are alike. This is also done through radio channel broadcasting. Their scheme was implemented on top of the TinyDB query processing system by using the packet-level TOSSIM as an emulator.

3.6.2 Base Station Optimisation

Bharatiya and Reddy (2012) are of the idea that query optimisation can be done through an emphasis on the base station. As per their idea, they seek to evoke energy efficacy using optimisation of the queries. Their idea considers the cost of communicating and sensing the data. In their explanation, they allude that once a query is received, it is first evaluated. This first evaluation aims at determining if the query that has been delivered to the base station can be assessed using the outcomes of other queries that are still in process. If the query is found to
be assessable, then a novel query is reconstituted at the base station utilising the ones that are running at that moment. The step is undertaken without having to inject the query into the sensor network. Hence, they opine that this optimises the processing of query networks in the wireless sensor network. Further, they note that the decrease of the queries in the wireless sensor network used in assessing the queries that have been inputted cuts down on the energy costs as per their idea. Besides, using the outcomes of the queries in process at that moment does away with the data requests that match. They also say that if the queries with an expression such as WHERE, that are alike, are to be evaluated, then they can be combined. This reduces energy utilisation by the wireless sensor network (Bharaktya and Reddy, 2012).

3.6.3 Heuristic approach

Compared to the previous optimisation techniques, Kumar et al. (2011) offer a query optimisation technique that is based on a heuristic approach. The costs the study seeks to tackle are those arising from computations such as the time of communication and processing. As explained by the authors, they have suggested an algorithm whereby the query is evaluated through the utilisation of a file that had been stored previously. The reason for this evaluation is to identify any enhancements that have been made to the query following the optimisation techniques that have been used in previous queries. They argue that there is a boost to the enhancement that had been made to the query due to the increased intricacy. The researchers go on to explain how their suggested query optimisation is based on heuristics functions. They say that once the binary tree is formed by the query optimiser, all the dependent variables are placed on one flank of the binary tree’s branch. They indicate that weight is allocated to every variable, which enables them to compute the cost on the basis of weight. Kumar et al. (2011) stated that the period that the operation or variable uses determine the amount of weight that it is allocated. The initiation of the query search triggers the quest for the required entry. At that moment every entry will be at the destination node, can be called the leaf. To explain their idea, the researchers give the example of a nested query. In the example, they explain that if the cost of running the query is “x” units, and in the nested query they run an estimate of twenty times, it gives a cost of 20 times “x” units. To show how their proposition reduced the cost, they explain that their prognosis is run only once. Hence, the cost will only be 1 times “x” units. Based on heuristics, the researchers opine that in the initial stages there might only be a slight enhancement; however, after a certain period, a greater enhancement will be shown once the information to be used fills the memory. They say that in any database system, similar queries are executed after a specific period. Hence, if they have prior knowledge of where the query
A search is going to occur, then they immediately search in that specified area only. This saves cost and time (Kumar et al., 2011).

Hatmode and Rangdale (2014) also talk about a query optimisation technique that is based on heuristics. The researchers in this study utilise what they call a Heuristic optimiser. As they specify, the Heuristic optimiser seeks to reduce the number of accesses made. It does this by cutting down on the number of columns and tuples that are examined. The study uses Heuristic regulations, which are the early execution of the projections and the selection. Through the early execution of the projections, the numbers of attributes are decreased, while the prior execution of the selection diminishes the tuple number. They note that their query optimisation technique aims at interpreting queries in SQL to mimic the greatly optimised queries in SQL that can be located in numerous databases used for commercial purposes. Their query optimisation is tested in a replica database to examine the execution time and speed with normal none heuristic query.

3.6.4 Cost-Based Optimiser

There has been much research on query processing in the area of distributed data systems. TinyDB (Madden et al., 2005) uses a cost-based optimiser to choose a query plan that has the estimated lowest overall power consumption using the metadata available in the sensor node’s catalogue that describes all events, attributes, functions and procedures as well as the cost of data processing and delivering. The metadata is copied to the root of the network to be used by the query optimiser.

One approach, as proposed by Zhong (2007), is that each sensor node samples the environment as specified by the query. According to the execution plan, sampled data is sent to the leader node, or together with partially aggregated data received from other nodes; aggregation operators are applied. Partial or incremental aggregation decreases the volume of data. After that, the responsibility of the leader node is to combine all the partially aggregated results and report it to the gateway node if the value exceeds the threshold.

From another angle, as proposed by Bagherinia et al. (2010) is to reduce the network consumption energy via executing operators with an optimised order and optimised method for aggregation functions, so that the sensor nodes do not send unimportant data to other nodes. They used declarative queries in network interaction to this end, where these queries play an interface role.
Different types of queries are presented by Chatzimilioudis et al. (2013) as part of their proposed optimised query routing tree using operator replacement. Universal queries need data from all the nodes of the network to be answered and usually involve a single aggregational operator that is applied to the entire data source. Subset queries only need data from a subset of nodes and can involve various operators applied to different groups of the data sources. Snapshot queries are performed if we just need the current readings that are stored at each node in the network. A continuous query is if we need the readings over several epochs of new readings and transmission.

3.6.5 In-network Query Optimisation for Wireless Sensor Database

In the last decade, there are various sensor network query processors (SNQPs) that provide data aggregation, data fusion, data merge, data compression, data prediction and filtering. These SNQPs view the WSNs as a distributed database over which declarative query processor could be used to program a WSNs application with much less effort (Jabeen and Nawaz, 2015). This part surveys several novel approaches of handling query processing by the current SNQP literature. Some topics in this field show in Figure 3-7.

Al-Hoqani (2017) proposed the algorithm “On Demand Query Sensing (ODQS)”, a query-based engine that transfers the sensed data into aggregation functions at both the routing and the sink aggregator. This algorithm optimizes the query because data is only sent when requested and irrelevant data is eliminated through aggregation within intermediate node. This has improved node’s lifetime in WSNs.

For optimizing the queries processed over the WSNs, a new algorithm named “Artificial Immune System (AIS)” in Modern Query Optimization (MQO) technique is introduced (Chakravarthi and Bhushan, 2015). This technique works in three different levels which include monitoring the path and the query, tracking the data based on the node information and choosing optimal nodes for data gathering according to query’s lifetime. Some issues such as time, cost and energy consumption are eliminated by the proposed MQO approach, which is efficient for query optimization.

For extending WSNs lifetime, carefully balancing energy, delay and accuracy is of importance and essential (Brahmi et al., 2015). The first step is to adopt Markov Decision Process (MDP) whereby a reward function is defined and then Genetic Algorithm (GA) is used to find a set of optimal decisions which could ensures the best trade-off among energy, delay
and accuracy of the received data based on their priority level. This yields a quite excellent performance.

Operator placement problem is relevant for in-network query processing over WSNs. The operator placement assignment defines on which node of the network each query will be hosted and executed (Chatzimilioudis, et al., 2013). An optimal algorithm named “Distributed Fermat Node Search algorithm (DFNS)” is present and it finds the optimal node to host the operator with minimum communication cost. Meanwhile, Operator Placement Adaptation (OPA) and Operator Tree Adaption (OTA) are two novel solutions which are capable of taking into account adaptive properties of operator placement problem. The results show that the cost overhead of this algorithm is reduced by 50%~100% compared to previously proposed techniques.

![In-network query optimisation for sensor network database](image-url)

**Figure 3-7** In-network query optimisation for sensor network database
The proposed optimizer takes a constant stream of queries as input and dynamically determines what is the best execution plan to run at the sensors while could still answer user queries (Muller and Alonso, 2012). Optimization of concurrent queries in WSNs was proposed and data model, query operators, cost model, data stream operator and query optimization strategies were introduced. This shows different strategies have various effects on the total execution cost. Specially, the min-execution cost strategy performs well across a wide range of multi-programming levels.

Because the WSNs consists of limited battery and low-cost nodes, energy efficiency must be employed for data aggregation and gathering for extending the lifetime of WSNs (Ramanan and Baburaj, 2010). Data fusion and aggregation techniques were used to minimize the total energy per round so as to get balance among per node. This also explored general networks lifetime in WSNs and made an extensive study to categorize available data gathering techniques and analyse possible network lifetime on time. Communication traffic is an activity with a highly energy cost and it drains out limited energy with a node thereby rendering it useless, therefore, efficient use of energy is the main focus to extend the lifetime of a WSN (Jindal, Verma and Bawa, 2015). Reduction of communication traffic thus is the most effective method for achieving energy efficiency. Optimization of multiple queries at the base station and data compression at individual nodes result in quantitative reduction in communication traffic. This method is more general and does not introduce any loss of packet.

Another promising solution to achieve network lifetime is caching useful data closer to the requesting node because data access latency could be reduced by caching (Srivastava and Sudarshan, 2014). Genetic Algorithm (GA) was carried out and showed the helps in selecting sensor nodes to implement caching. The Scaled Power Community Index Cooperative Caching scheme (scaPCICC) was run on the optimized network and showed an excellent performance in reducing energy consumption. This concludes that a WSN with a significant number of active sensors consumes less energy than a dense network with few sensors by comparing GA and SCAppcicc. Deshpande (2017) provided an overview of the work in adaptive query processing, which identifying its common themes, laying out the space of query plans, and discuss why it is needed, how it is being implemented, where it is most appropriately used. Reusing sensory data for answering concurrent application is another solution (Zhou et al., 2016). They argued that the cooperative caching mechanism is a trend nowadays.
Some query processing and optimization techniques, such as data compressing, data fusion, data aggregation and data prediction, are discussed by Jindal, Verma and Bawa (2016). In view of this survey, though there is a plethora of information extraction approach from WSNs, each having its own pros and cons. It mainly focuses on the data optimization among sensor nodes. In fact, it is a local optimization. In summary, Table 3-4 provide an overall summary of this research area.
<table>
<thead>
<tr>
<th>Name of Project/ Authors</th>
<th>Authors</th>
<th>Proposal approach</th>
<th>Research achievement</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Network On-Demand Query-Based Sensing System for Wireless Sensor Networks</td>
<td>(Al-Hoqani et al., 2017)</td>
<td>On Demand Query Sensing</td>
<td>An improvement in energy efficiency by using aggregation function in some node.</td>
</tr>
<tr>
<td>Planning &amp; acting: Optimal Markov decision scheduling of aggregated data in WSNs by genetic algorithm</td>
<td>(Brahmi et al., 2017)</td>
<td>Markov Decision Process (MDP) and Genetic Algorithm (GA)</td>
<td>A best trade-off between energy saving, delay and accuracy and the optimization shows an excellent enhancement up to 20%.</td>
</tr>
<tr>
<td>Distributed Database Management Techniques for Wireless Sensor Networks</td>
<td>(Diallo et al., 2015)</td>
<td>Classification of state of the art of the techniques used to manage data and queries.</td>
<td>A guideline for further contributions</td>
</tr>
<tr>
<td>In-network wireless sensor network query processors: State of the art, challenges and future directions</td>
<td>(Jabeen and Nawaz, 2015)</td>
<td>Sensor network query processors (SNQPs)</td>
<td>Challenges and opportunities of this research</td>
</tr>
<tr>
<td>Quantitative Reduction in Communication Load for Energy Efficiency in WSN</td>
<td>(Jindal, Verma and Bawa, 2015)</td>
<td>Reduction of communication traffic</td>
<td>employs optimization of multiple queries at the base station and data compression at individual nodes resulting in quantitative reduction in communication traffic</td>
</tr>
<tr>
<td>A Survey on Query Processing &amp; Optimization Techniques in WSN</td>
<td>(Jindal, Verma and Bawa, 2016)</td>
<td>Query based approach Agent based approach Macro programming-based approach</td>
<td>Data optimization and transmission optimization are the main areas of research.</td>
</tr>
<tr>
<td>Topic</td>
<td>Author(s)</td>
<td>Details</td>
<td>Benefits</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Optimization of concurrent queries in wireless sensor networks</td>
<td>(Müller and Alonso, 2012)</td>
<td>takes a constant stream of queries as input and dynamically determines what is the best execution plan</td>
<td>data model, query operators, cost models, query optimization strategies, and data stream operators</td>
</tr>
<tr>
<td>Data Gathering Algorithms for Wireless Sensor Networks: A Survey</td>
<td>(Ramanan and Baburaj, 2010)</td>
<td>An extensive study to categorize available data gathering techniques</td>
<td>Efficient use of energy in WSNs</td>
</tr>
<tr>
<td>Probability Model-based On-demand Query Processing for WSNs Database</td>
<td>(Al-Hoqani et al., 2017)</td>
<td>Dynamic programming (DP)</td>
<td>Massive saving of energy</td>
</tr>
<tr>
<td>Energy-efficient cache node placement using genetic algorithm in wireless sensor networks</td>
<td>(Srivastava and Sudarshan, 2014)</td>
<td>Genetic Algorithm (GA), Scaled Power Community Index Cooperative Caching scheme (scaPCICC)</td>
<td>reducing the number of messages; reducing the data access latency and the energy consumption Increased the lifetime of WSNs</td>
</tr>
<tr>
<td>An Energy-Aware Spatial Index Tree for Multi-Region Attribute Query Aggregation Processing in Wireless Sensor Networks</td>
<td>(Tang et al., 2017)</td>
<td>novel energy-efficient heuristic density-based clustering model multi-attribute spatial index tree</td>
<td>Reduced the energy consumption, query time, and increased the network lifetime.</td>
</tr>
<tr>
<td>Cache-Aware Query Optimization in Multiapplication Sharing Wireless Sensor Networks</td>
<td>(Zhou et al., 2016)</td>
<td>energy-efficient query optimization mechanism gray model GM (1, 1)</td>
<td>reduce the energy consumption significantly, improve the network capacity to an extent</td>
</tr>
<tr>
<td>A novel query optimization method for wireless sensor networks</td>
<td>(Sun, 2007)</td>
<td>Novel query optimization</td>
<td>Find the optimal combinations of sending sensor data.</td>
</tr>
</tbody>
</table>

Table 3-4 Summary of in-network query optimisation research
3.7 Conclusion

This chapter discussed the literature review for the research objectives covered in this thesis. Different surveys have been conducted for data sampling and query processing. The techniques which we would consider for this thesis are data gathering techniques, data reduction techniques and aggregation techniques. Moreover, other techniques have also been discussed, including approximation-based techniques and join techniques and presented.

In conclusion, from the above-discussed literature, the research approach undertaken is novel. In our approach, we integrated aggregation techniques as well as merge techniques using the historical approach to on-demand query-based methods. These are found in more detail in chapters five and six of this thesis.
Chapter 4 Adaptive Sampling for Wireless Sensor Networks

This chapter discusses the concept of adaptive sampling for wireless sensors using household water consumption monitoring as a case study. In the chapter, a TCP-congestion control idea is proposed for the purpose of adaptive sampling algorithm generation. The chapter presents the designed algorithm methodology and its performance metrics used to make sure the over performance is achieved.

4.1 Introduction

Wireless sensing systems (WSS) are currently gaining wide use in data collection in various applications such as environment monitoring, energy consumption monitoring and control, and industrial condition monitoring. The WSS systems are predominantly battery driven with low data rates; therefore, it is undesirable to gather all available data without considering the dynamics of the monitored environments or processes. This study investigates an adaptive sampling strategy for WSS aimed at reducing the number of data samples by sensing data only when a significant change in these processes is detected. This detection strategy is based on an extension to Holt's Method and statistical model. To investigate this strategy, the water consumption in a household is used as a case study. A number of performance metrics are used to evaluate the adaptive sampling strategy, including sampling fraction, missing ratio and sampling performance. The experimental results show that the proposed strategy over-performs compared to two existing sampling algorithms.

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Smoothing parameter $0 &lt; \alpha \leq 1, t &gt; 0$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Smoothing parameter $0 &lt; \beta \leq 1, t &gt; 0$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Detection ratio</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Error tolerance</td>
</tr>
<tr>
<td>$S_{t+1}$</td>
<td>Estimate at instant $t+1$</td>
</tr>
<tr>
<td>$S_t$:</td>
<td>Current estimate</td>
</tr>
<tr>
<td>$y_t$:</td>
<td>Current Reading</td>
</tr>
<tr>
<td>$\epsilon_t$</td>
<td>Forecast Error</td>
</tr>
<tr>
<td>$b_{t+1}$</td>
<td>Trend at instant $t+1$</td>
</tr>
<tr>
<td>$y_{t+k}$</td>
<td>Estimated reading at instant $t+k$</td>
</tr>
</tbody>
</table>
Wireless sensing systems have recently become a topic of interest to many researchers, especially in terms of in-network data collection. Wireless sensor networks (WSN) are considered one of the reliable environmental monitoring systems that are constructed using various techniques and numerous algorithms, which are studied and analysed according to the deployment plan. Courtesy of its targeted sensing accuracy cost reduction, and energy consumption, the technology has been used in studies and experiments that have been conducted and implemented with great precision. Although many researchers are continuously working in this field, it appears less than satisfactory for the rapid development and technological improvements of the future. For example, for event detection; adaptive sampling is used to balance the sample rate when an abnormal behaviour or event in the collected data is detected.

This chapter addresses the problem of capturing the essential details from the measurement of household water temperature while minimising the energy consumption of the sensor’s battery. The existing adaptive sampling techniques proposed for data acquisition has critical limitations; particularly in the concern of energy consumption. Hence, we propose an adaptive sampling algorithm based on time series statistics and the concept of TCP Reno congestion control, that user a network congestion-avoidance algorithm. This algorithm's factors adjust the transmission rate limit according to the received rate and increase by one always, if it detects any changes it will drop by half of the current windows size, other wise to increase it by double all the time according to received packet's bytes. Our approach is designed to generate data from the source node only when a significant change is detected.

Gupta et al., (2011) carried out research work on monitoring pollution levels from car exhaust gases. They implemented their work in a novel sampling algorithm for the design and
evaluation of the datasets. This team used the EDSAS (Exponential Double Smoothing based Adaptive Sampling) algorithm, which is considered as a time series technique. EDSAS was deployed as a data reduction technique based on predictions for an irregular sampled time series known as Wright’s Extension to Holt’s Method. It also incorporated the use of EWMA (Exponential Weighted Moving Average). Furthermore, Borgne et al., (2007) proposed an adaptive model selection technique for time series prediction.

The technique was considered as a lightweight, online algorithm for a temperature time series acquired from a sensor deployed in the real world. It was implemented to adaptively choose the best performing model for satisfying data prediction algorithm Borgne et al., (2007). The selected model called Algorithm Model Selection (AMS) was based on an Auto Regression (AR) model, whose parameters can be updated in real-time. The authors stated that the prediction model allows the detection of outliers to be recovered with more possible testing values. However, other authors (de Aquino et al., 2007); (Huang, Zhang and Xu, 2011) concentrated on using different mechanisms for the purposes of monitoring increases in network lifetime and reducing both delays and energy consumption. Data stream and prediction filtering schemes for sampling are used respectively to solve the energy wastage problem. An adaptive data collection strategy was designed to allocate a number of updates that were allowed to be directed to the sink. Even though the lifetime-constrained strategy proposed was successful, missing data was recorded as having a more negative influence on data accuracy (Tang and Xu, 2008).

Other researchers have proposed adaptive sampling techniques that focus on the reduction of the resources used by WSNs in a decentralised manner. The technique proposed by Bhuiyan and Wang, (2013) and Jain, Chang and Wang, (2004), involved new samples to be collected. Notably, when the sampling rate violates a set range, a new one is requested from the server. (Masoum, Meratnia and Havinga, 2013), targeted applications that can tolerate changes in sensor values as long as measurements fall outside a specific range. Cheng et al., (2010), also presented another adaptive sampling approach that is focused on the efficiency of energy consumption. Their work proposed an approach using a new matrix called EDCA (Efficient Data Collection Approach) to lower the sampling rate and make sure fewer packets are transmitted. Moreover, Arici and Altunbasak, (2004) proposed an adaptive sensing method that offers a compromise between system lifetime and distortion of the reconstructed image.
The main idea in their method is to keep the sensors more active at intervals when the measurements show small variations. Zhou and De Roure, (2007) proposed yet another sampling technique that is applied to flood warning systems. Jin et al., (2010), designed and proposed a water environment monitoring system that was based on WSNs. This system can easily be configured as a random constraint or parameter monitoring network. Additionally, another system of remote water quality detection measuring, and monitoring based on WSNs has been proposed by Wang et al., (2012). It was designed mainly for the purposes of reducing energy consumption and ensuring effective information acquisition in WSNs. Most of the above water applications based on WSNs were designed to detect the quality of water or water pollution.

This thesis motivation is to save the energy consumption from the sensors resources, this work is driven from a water system project for research management. This chapter presents part of the research outcomes from an ongoing EU-funded project Integrated Support System for Efficient Water Usage and resources management (ISS-EWATUS) (http://issewatus.eu). We aim at achieving the best possible energy saving algorithm, as well as the accuracy of sensing readings. The experiment was conducted using a household’s water temperature as a case study.

4.2 Time Series Prediction Model

Single Exponential Smoothing (SES) method is used here as a time series prediction model. SES forecasting uses an SES formula whose values $S_{t+1}$ are obtained using a smoothing parameter ($\alpha$) as the highest for the current reading $y_t$, and a decreasing weight ($1-\alpha$) at the distant observation $S_t$, while $S_{t+1}$ is a single step estimate and $S_t$ is the current estimate. This SES forecasting is presented in Eq. (4-1) as Exponential Weighted Moving Average (EWMA)

$$S_{t+1} = \alpha y_t + (1 - \alpha)S_t \quad 0 < \alpha \leq 1$$

(4-1)

Eq. (4-1) can be rewritten as Eq. (4-2) by introducing a forecast error $\varepsilon_t$ for estimated instant t. Where: $\varepsilon_t = y_t - S_t$

$$S_{t+1} = \alpha \varepsilon_t + S_t \quad 0 < \alpha \leq 1$$

(4-2)
SES does not work well with datasets that exhibit a trend. The trend $b_t$ itself can be presented in an SES formula as shown in Eq. (4-3) where $b_t$ represents the trend at instant $t$ and $\beta$ is a smoothing parameter used for the linear trend. Eq. (4-3) updates the trend with the difference between the two previous consecutive estimates. Introducing the trend $b_t$ into the SES forecasting to adjust the previous observation $S_t$, the SES become exponential double smoothing (EDS) as shown in Eq. (4-4). This method, which uses trends, is known as Holt’s Method. The $k_t$ step estimate $\tilde{Y}_{t+k_t}$ is given in Eq. (4-5) (Wright, 1986) by assuming the trend is maintained at a same level for $k_t$ steps where $k_t$ being positive integer ($k_t = 1, 2, ...$).

$$b_{t+1} = \beta(S_{t+1} - S_t) + (1 - \beta)b_t \quad 0 < \beta \leq 1$$  \hspace{1cm} (4-3)  

$$S_{t+1} = \alpha y_t + (1 - \alpha)(S_t + b_t) \quad 0 < \alpha \leq 1$$  \hspace{1cm} (4-4)  

$$\tilde{Y}_{t+k_t} = S_t + k_t b_t, k_t \geq 1, t > 0 \quad (k_t = 1, 2, ....)$$  \hspace{1cm} (4-5)  

Eq. (4-1) to (4-5) are normally used as a prediction model for constant sampling interval situation, i.e. the interval between two sampling points, instant $t$ and $t+1$, is a constant. If the sampling interval is the multiple of a basic sampling interval in instant $t$ Eq. (4-3) to (4-5) can be re-written as Eq. (4-6) to (4-8), for the explanation of reading the sensed data $S_t$ at time $t$ or data ($S_{t-1}$) at time $(t - 1)$ when data available at earlier time, instead of sensing the data at later time $(t + 1)$ . The first item in Eq. (4-6) represents the average contribution of the latest change in the prediction to the trend; the second item is the moving average. The second item in Eq. (4-7) introduces the adjusting to the previous prediction by taking consideration of the trend; the first item is the moving average. Eq. (4-8) is used for prediction only if there is no current reading available.

$$b_t = \beta \frac{(S_t - S_{t-1})}{k_{t-1}} + (1 - \beta)(b_{t-1})$$  \hspace{1cm} (4-6)  

$$S_t = \alpha y_{t-1} + (1 - \alpha)(S_{t-1} + k_{t-1} b_{t-1})$$  \hspace{1cm} (4-7)  

$$\tilde{Y}_{t+1} = S_t + k_t b_t, k_t \geq 1, t > 0$$  \hspace{1cm} (4-8)  

Eq. (4-8) can be used for predicting long-term moving average ($S_{long}$) and short-term moving average ($S_{short}$) with two users defined smoothing parameters $S_{long}$ and $S_{short}$, that are defined and tested as more sufficient statistics for the used data. By considering a value for both
averages and evaluated at each parameters point from $\alpha$ and $\beta$, which is between 0 and 1, at some points when the values of the predicted and sensed data ratio are at a specific threshold.

This ratio $\eta$ of the short term moving average ($S_{\text{short}}$) and the long term moving average ($S_{\text{long}}$) provides an indication of a sudden change or the occurrence of a rare event when the ratio $\eta$ exceeds a user specified threshold (1.0 in this study)

$$\text{Ratio } \eta = \frac{S_{\text{short}}}{S_{\text{long}}}$$  \hspace{1cm} (4.9)$$

4.3 TCP Congestion control

When too many packets appear in the network, the performance of the network begins to decline; this is called congestion. In order to prevent network congestion, TCP proposed a series of congestion control mechanisms. As the most widely used end to end transport protocol on the internet, TCP is a congestion control mechanism mainly based Additive Increase Multiplicative Decrease (AIMD). Stevens et al. (1999) developed the TCP congestion control mechanism, which is composed of ‘Slow Start’ and ‘Congestion Avoidance’. Even though many years have passed, TCP congestion control mechanisms are still very common in use. Moreover, frequent large and small improvements have been made in this area, it has already become of a hot area of network research. TCP congestion control algorithms try to balance between both sides and choose the most suitable value, thus allowing the network to get maximum throughput without congestion.

The process of Additive Increase Multiplicative Decrease (AIMD) can be divided into two phases. First is the additive increase: each time the source end receives a confirmation acknowledgement (ACK) from the receiving end, the congestion window will increase by two. This is the additive increase phase. The second phase is the multiplicative decrease: when a data transmission time-out occurs, TCP assumes that there is congestion on the route and begins to decrease the transmission rate of the source data. Every time a time-out occurs, the source end will recalculate the value of the window and cut it in half: an explanation of our algorithm, based on TCP congestion control, follows in Figure 4-1.
Initially, TCP will send a large number of data packets to the network once the connection is established, which can easily lead to the exhaustion to network router cache space, which causes congestion. So, the newly established connection cannot start sending a large number of data packets. In order to avoid the occurrence of congestion, TCP raises the amount of data to be sent each time gradually, according to the network situation. This phase is called the Slow Start. It is also known as sender-based flow control, which is a mechanism used by the sender to control the transmission rate. That means that the sender’s transmission rate is determined by the rate of ACKs from the receiver. In the very beginning, when a TCP connection first begins, the Slow Start algorithm initialises a congestion window (cwnd) to one segment; the source end can only have sent one segment each time. Every time an ACK arrives, a congestion window known as cwnd is incremented.

The cwnd is effectively doubled per round trip time (RTT). In reality, a Slow Start is not very slow when the network is not congested. For instance, the cwnd increases to two segments after the first successful transmission and ACK of a TCP segment. Then, when the transmission of these two segments and ACKs is completed, the cwnd will increase to four, then eight segments, then sixteen segments, and so on. In this way, it continues doubling from there on until it reaches the maximum cwnd size advertised by the receiver or until congestion occurs.
4.4 AS-TCP Algorithm phases

For this project, we decided to design a novel adaptive sampling algorithm with reference to this technique’s basic concept. Because of this, we need to have a basic understanding of TCP Congestion Control. A detailed explanation will be given through our algorithm. The TCP Congestion Control algorithm was designed with many variables. When it comes to the data sensing process in WSNs, if the sensors sense all the data at each sensing point, it will consume too much energy and largely shorten the lifetime of sensor nodes. Here we have provided a basic outline of TCP; the AIMD, slow start and congestion avoidance parts of the TCP Congestion Control strategy. These basic ideas can be used in the design of adaptive sampling algorithms as shown in the next steps. We named this adaptive sampling algorithm as AS-TCP as it is based on TCP congestion control principle. As discussed earlier, but in more clear steps.

Step 1:

In the first step, the data trend and data level are calculated to find the SES (Simple Exponential Smoothing) using the Alpha (α) parameter, then the EDSAS is calculated by specifying the Ebase value (SES) and the trend (using the Beta β parameter value). After that, the forecast value is found by adding the trend value to the EDSAS value. The forecast accuracy measurement is calculated between the predicted data values and the actual data values at each sampling instant, this is called the forecast accuracy error. As the forecast error (Fe) indicates the error range between the predicted value and the actual value, only a positive number makes sense because the negative is unmeasurable and at both case of negative and positive it will be detected as an up normal event. However, the difference value between them can be positive or negative, so we find the absolute (ABS) value of the forecast error (Fe) based on the step size prediction. The step size prediction is calculated depending on the Fe. In the Excel table available in the appendices, the ABS error column is the absolute value of the forecast error. That the (Fe) a user-defined error will be compared with the estimated error tolerance level ε.

Step2:

The step size can be defined as the time interval between two consecutive sampled data points. Originally, the step size is set to one or the real dataset. However, the data value may stay the same, or nearly the same, for several consecutive sampling intervals, in which case these values do not need to be sampled. So, in the AS-TCP algorithm, the step size will change,
and only sense data when a significant change happens. The process of defining the step size, we call it K-Step Prediction. The K-Step Prediction process here draws lessons from the TCP Congestion Control mechanisms; use the basic concepts of Slow Start and congestion avoidance. The sampling interval is computed depending on the relationship between the $F_e$ and a user-defined error tolerance level $\epsilon$. If the $F_e$ is greater than error tolerance $\epsilon$, a change of maximum step size and sampling interval will be triggered. The sampling interval which we call step size (K) here is initialized to one, and there is also a user-defined maximum step size (Smax).

**Step 3:**

At this stage, we will implement the AS-TCP technique, which will be discussed in detail in this chapter. If the forecast error is greater than the error tolerance, then we will cut the K interval by half. If it is less than the value specified for error tolerance, then we will double the interval. The step size K parameter is initialised to one and can be incremented to a user defined maximum step size, S-max (600s, which is 16 minutes). At every sample instant, the step size is doubled, as long as the forecast accuracy measure ($F_e$) stays below the user defined error tolerance level ($\delta=0.99$). This is repeated until it reaches the Smax step size. On the other hand, if the forecast accuracy measure ($F_e$) stays above the user defined error tolerance level ($\delta=0.99$), then the step size is decreased by half. This is repeated until it stays at one. This step we call step size modification for the proposed algorithm. As it was easily described and simplified by figure 4-1.

Therefore, if ($F_e$) is smaller than the defined error tolerance, the step size will be doubled; this multiplicative increase phase is like the Slow Start phase. This process will be repeated unless K reaches the Smax. When it reaches the maximum step size (Smax), the step size (K) increasing speed will slow down it will change to an additive increase, increased by one each time. Otherwise, if the ($F_e$) is greater than the defined error tolerance, the Smax will be sharply decreased to half of the current step size, while at the same time, the step size (K) will be reduced directly to one. This phase imitates the congestion avoidance algorithm. A schematic of K-step prediction is shown in the resulted figures below.

**Step 4**

An adjustment feedback model needs to be triggered, which will make sure to maintain data fidelity and minimize the number of false misses. False misses are important events or
changes happening in the environment, as well as guarantees on the accuracy of the data that we have gained from the sampling technique used. For adjustment feedback, we used exponential weighted moving averages (EWMA) using the following equation where $\alpha$ is the parameter to be used for forecasting the sensor readings and $t$ is for time where ($0 < \alpha > 1$ and $t > 0$). The $L_{t+1}$ is used as a notification for the new moving average weight for the forecasted values.

$$L_{t+1} = \alpha y_t + (1 - \alpha)L_t$$  \hspace{1cm} (4-10)

To use this mechanism, we need to calculate EWMA using the two parameters, $L_{\text{short}}$ and $L_{\text{long}}$ for the sensed readings using time series forecasting formulae. With the two smoothing parameters ($L_{\text{short}}$) and ($L_{\text{long}}$) at a specific time interval, the below equation will give the sudden change when the value exceeds the threshold (Werner-Allen et al., 2005). The values for the two parameters are 0.0250 and 0.0011 respectively. In this experiment $L_{\text{short}}$ is the EWMA at parameter $L_{\text{short}}$ and $L_{\text{long}}$ is the EWMA at parameter $L_{\text{long}}$ for the forecasted reading $L_{t+1}$ at specific time.

$$\text{Ratio } \eta = \frac{L_{\text{short}}}{L_{\text{long}}}$$  \hspace{1cm} (4-11)

After calculating the EWMA using $L_{\text{short}}$ and $L_{\text{long}}$ the ratio is measured and then compared with the threshold (1.0ppm) to find out if a sudden change is happening. After that, if outliers are also detected then the trend adjustment feedback needs to be calculated for the false data with the forecasted accuracy error, using the statistical theory which will be discussed in section 4.5, and that gives us an indication that, the less missing rates, the better the accuracy results.

4.5 Design of the Adaptive Sampling Algorithm

4.5.1 Algorithm design

In order to reduce the number of data samples and avoid unnecessary energy consumption, an energy efficient adaptive sampling algorithm is designed in this study. The algorithm is based on the implementation of time series forecasting described in the previous section to predict the future value and detect possible changes. In addition, it uses the basic concept of the TCP Reno congestion control to adjust the dynamic sampling interval, i.e. the
step size and its maximum value; therefore, we name this new adaptive sampling algorithm as AS-TCP.

Apart from saving energy, we must ensure that the important changes that happen in the environment are not missed. Therefore, EDS forecasting, which uses current temperature and the trend to generate the forecasted temperature, will be used. After implementation, the evaluation of this algorithm will be worked out using a real water temperature dataset. The amount of sensed data, the energy saved by the sensor and data accuracy levels will be evaluated with various metrics. Figure 4-2 is a flowchart illustrating the steps followed to form the data generation framework for our AS-TCP algorithm.

![Flowchart](image)

**Figure 4-2** AS-TCP Algorithm process design

### 4.5.2 Raw Dataset

Having described the project specifications, we now turn to the raw dataset comes from the temperature of a real water environment from another project. The sensor node senses data at a fixed frequency, which is once per second. Nearly a whole day’s actual temperature data has been extracted, which is 86,177 data points in total. We store them in the Excel, named this Actual column Temperature, numbered the data points using the time in seconds at which they were sensed, and they were ready to be used.

In our experiment, temperature data had been collected from projects by other people in the university. The data collected was for one full day (24 hours), which is 86,400 minutes. The data was collected by another student group working on wireless sensor network many thanks to them. Moreover, the readings were stored in the sink every second. So, we acquired 86,177 readings in excel format. Having the actual readings for 86,177 seconds in a lot date, but it still
means there are 223 seconds from which the data are missing. Imagine that this data is to be collected daily for ever; how much energy is it going to consume? In our project, we proposed to predict the temperature for the rest of the days of a whole year and then to check at the critical times when a change is detected. As a result, the data can be kept in the sink and only when the module gives a sign of a change in the readings, will be communicated to the server.

### 4.5.3 AS-TCP Algorithm

To explain the proposed AS-TCP algorithm, a schematic of slow start and congestion control algorithm is presented in Figure 4-3. In order to prevent network congestion, TCP Reno congestion control adopts Additive Increase Multiplicative Decrease (A MID) congestion control mechanism. Following this TCP Reno idea, the sampling step size is doubled every interval until it reaches the pre-specified maximum step size ($S_{\text{max}}$). The sampling is then halted and maintained at that interval as long as the forecast error is below the allowed error tolerance limit ($\delta$) and no event is detected. During the additive increase phase and the maximum step size maintaining phase the, sampling step size will be halved if the forecast error is beyond the allowed error tolerance limit or an event is detected. This is the multiplicative decrease phase. The AS-TCP pseudo is given in Figure 4-3.

---

Initialize variables $k_t=1$, $\delta$, $S_{\text{max}}$

Initialize EDS parameters $\alpha$, $\beta$

Initialize EWMA parameters $\alpha_{\text{short}}$, $\alpha_{\text{long}}$

% produce data prediction module

While $k_t <$ total data set

$y(t)=$Actual reading

$S(t)=$prediction reading

% evaluate forecast accuracy error

If $(y(t) - S(t)) < \delta$

% k-step size modification If $(k_t < S_{\text{max}})$ then $k_t$ is doubles;

If $(k_t = S_{\text{max}})$ stay at $S_{\text{max}}$ and check for event detection

% detection mechanism

Evaluate $\eta = S_{\text{short}}/S_{\text{long}}$ // as explained in algorithm phases

% feedback adjustment

If $\eta >$ threshold then $k_t$ drop to half of the current interval

Count missed false; 

Sense data;

---

Figure 4-3 AS-TCP pseudo code
4.5.4 Performance metrics

There are different performance metrics used to evaluate the dynamic sampling algorithms. Three of them are selected here for the convenience of comparison with other existing algorithms in next sections. These three metrics are sampling fraction (SF), Miss Rate (MR) and Sampling Performance (SP) described as follows (Gupta et al., 2011).

1. Sampling Fraction (SF): SF is defined as the ratio of the total number of data samples taken by the sampling algorithm and the total number of data points available in the real dataset if a fixed sampling interval is taken. SF represents the sampling fraction of the energy saving achieved by the sampling algorithm.

2. Miss Ratio (MR): MR is the fraction of the number of events/changes that have not been detected by the algorithm as shown in Eq. (4-10) where \( n \) denotes the total number of sampling points, and \( n_f \) denotes the number of miss points.

\[
MR = \frac{n_f}{n} \quad (4-10)
\]

3. Sampling Performance (SP): SP combines SF and MR into a single performance/cost metric as shown in Eq. (4-11). (1-SF) denotes the energy saving obtained by the algorithm and MR denotes the miss ratio. Higher energy saving, and lower data miss will lead to higher values of sampling performance.

\[
SP = \frac{1-SF}{MR} \quad (4-11)
\]

![Figure 4-4](image.png)

Comparison of sensed data, missed data, predicted data with the actual dataset
4.6 Experimental Results

4.6.1 Test results on the experimental dataset

An experimental dataset was generated by a wireless sensing system developed in ISS-EWATUS project for a household water usage over 24 hours. Both water temperature and flow rate were recorded by the system where a fixed sampling interval (1 second in this study) was utilised. Figure 4-4 shows the prediction, missing data points, sensed data points by the adaptive sampling algorithm in comparison with the actual data from the original dataset. Only water temperature is used here for the sake of simplicity. Two critical parameters $S_{\text{max}}$ and $\delta$ are chosen at 600 and 0.5 respectively after various training set have been also tested. There are a number of clearly noticeable outliers as shown in Tables 4-1 and 4-2.

<table>
<thead>
<tr>
<th>$S_{\text{max}}$</th>
<th>SF</th>
<th>SP</th>
<th>MR</th>
<th>Missing data</th>
<th>Sensed data</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.0023</td>
<td>6.1103</td>
<td>0.1633</td>
<td>32</td>
<td>196</td>
</tr>
<tr>
<td>600</td>
<td>0.0019</td>
<td>6.0983</td>
<td>0.1637</td>
<td>27</td>
<td>165</td>
</tr>
<tr>
<td>700</td>
<td>0.0017</td>
<td>6.2055</td>
<td>0.1609</td>
<td>23</td>
<td>143</td>
</tr>
<tr>
<td>800</td>
<td>0.0015</td>
<td>6.6195</td>
<td>0.1508</td>
<td>19</td>
<td>126</td>
</tr>
<tr>
<td>1000</td>
<td>0.0012</td>
<td>6.8547</td>
<td>0.1457</td>
<td>15</td>
<td>103</td>
</tr>
<tr>
<td>1200</td>
<td>0.0010</td>
<td>6.7573</td>
<td>0.1478</td>
<td>13</td>
<td>88</td>
</tr>
</tbody>
</table>

Table 4-2 Performance metrics when Smax=600 and $\delta$ is between 0.30 and 0.99

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>SF</th>
<th>SP</th>
<th>MR</th>
<th>Missing data</th>
<th>Sensed data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>0.0032</td>
<td>1.1988</td>
<td>0.8315</td>
<td>78</td>
<td>200</td>
</tr>
<tr>
<td>0.50</td>
<td>0.0023</td>
<td>2.2934</td>
<td>0.4350</td>
<td>47</td>
<td>178</td>
</tr>
<tr>
<td>0.60</td>
<td>0.0021</td>
<td>3.7791</td>
<td>0.2641</td>
<td>45</td>
<td>177</td>
</tr>
<tr>
<td>0.80</td>
<td>0.0020</td>
<td>4.8754</td>
<td>0.2047</td>
<td>34</td>
<td>170</td>
</tr>
<tr>
<td>0.90</td>
<td>0.0019</td>
<td>6.0983</td>
<td>0.1637</td>
<td>27</td>
<td>165</td>
</tr>
<tr>
<td>0.99</td>
<td>0.0019</td>
<td>6.0983</td>
<td>0.1637</td>
<td>27</td>
<td>165</td>
</tr>
</tbody>
</table>
In order to understand the influence of the critical parameters of the adaptive sampling algorithm on the performance metrics of the adaptive sampling, various experiments have been conducted here. Setting the maximum step size $S_{max}$ at 600 and increasing the error tolerance
\( \delta \) from 0.1 to 1.0, both of the resulting sampling fraction (SF) and missing ratio (MR) are decreasing as shown in Figure, 4-5.

Similarly, setting the error tolerance \( \delta \) at 0.99 and increasing the maximum step size \( S_{\text{max}} \) from 100 to 1100, both of the resulting sampling fraction (SF) and missing the ratio (MR) are decreasing as shown in Figure 4-4. When combing Figure 4-5 and 4-6 together by using the sampling performance in the metric, the overview of the relationship between critical parameters \( S_{\text{max}} \) and \( \delta \) and the SP is obtained as shown in

Figure 4-7 shows that setting \( S_{\text{max}} \) between 500 to 600 or 700 to 1000 could generate sharp increase in the sampling performance with the increase of \( \delta \) from 0.6. This observation could be used as a guideline on how to set the critical parameters in terms of the desired sampling performance.

![3D Plot](image)

Figure 4-7  Sampling performance in different maximum step size and error tolerance

### 4.6.3 Comparison with existing algorithms

Two existing algorithms are chosen for the comparison with the developed AS-TCP. The first one is EDSAS proposed by (Gupta, 2013) and the second one is the e-Sense (Liu, Chandra and Srivastava, 2006). Their principles have been reviewed in the introduction section.
Both of them aim to provide a sampling schedule for sensor node only when a state change is likely to happen, which is in a similar spirit with this research. Table 4-3 gives the comparison of these three sampling algorithms in terms of sampling fraction, missing ratio and energy saving. Figure 4-8 presents their comparison in terms of the sampling performance. The AS-TCP achieves the lowest sampling fraction and a similar missing ratio, therefore expecting the most energy saving. The sampling performance of the AS-TCP is slightly better than e-Sense and EDSAS.

Table 4-3 Comparison with e-Sense and EDSAS

<table>
<thead>
<tr>
<th>Algorithm Used</th>
<th>Error tolerance</th>
<th>Maximum Step Size</th>
<th>Sampling Fraction</th>
<th>Missing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS-TCP</td>
<td>0.90</td>
<td>600s</td>
<td>20%</td>
<td>16%</td>
</tr>
<tr>
<td>EDSAS</td>
<td>0.01</td>
<td>10s</td>
<td>42%</td>
<td>15%</td>
</tr>
<tr>
<td>e-Sense</td>
<td>0.10</td>
<td>20s</td>
<td>61%</td>
<td>10%</td>
</tr>
</tbody>
</table>

The clear significant change among the three researches is noticeable that the AS-TCP is advanced at points where a change is detected, so as a result our algorithm adaptively change to the existing environment and to the fluctuations in various time interval sensor readings. It is clearly noticed that at 2400 seconds in Figure 4-8, the performance was recorded to be the highest since the sampling fraction as shown in Table 4-4 is recorded to be the lowest for AS-TCP algorithm.
4.7 Conclusions

Inspired by computer network congestion control TCP mechanism this chapter proposed an AS-TCP adaptive sampling algorithm and tested it in a domestic water temperature dataset offline. On average, the AS-TCP can achieve 20% sampling fraction and maintain missing ratio around 18%. Comparing with two existing similar algorithms e-Sense and EDSAS, the AS-TCP has lower sampling fraction and similar sampling performance. Therefore, the AS-TCP is expected to achieve more energy saving than e-Sense and EDSAS.
Chapter 5 In-network On-demand Query-based Sensing System for Wireless Sensor Networks

This chapter discusses the concept of on-demand query-based sensing approach for wireless sensors using room monitoring as a case study. The chapter presents the on-demand sensing idea used for sensing data with a high requirement of saving energy, using the different aggregation algorithms and query engine that merge the data. In this chapter, the designed algorithm methodology and its power consumption metrics used is discussed for the purpose of solving the energy saving problem for on-demand query sensing systems and to make sure the query transformation is achieved significantly. The real-time sensing idea used for sensing data where there is a high need to save energy, using the different aggregation algorithms and query engines that merge the data. In this chapter the designed sensing system setup as well as its algorithm methodology is introduced, together with the power consumption metrics used to make sure the query transformation is achieved significantly.

The design and the development of an on-demand data acquisition sensing system called an On-Demand Query-based system (ODQS) is presented. We identify the set of network topology, database and development tools as well as the performance metrics and scaling factors for the evaluation of our experiment. We deployed Sensinode cc2430, which is operated with the Contiki operating system. We express the dynamic programming with interrelated routing protocols and aggregation techniques on a WSN emulation platform.

5.1 Introduction

A wireless sensor network is a large number of static or mobile sensor nodes or devices, which aim at assessing and monitoring environment or processes such as temperature, humidity, light conditions, and noise pollution, among others. Data acquisition is the task responsible for efficiently acquiring samples from the sensors in a sensor network. Energy efficiency could be achieved by minimizing the number of samples obtained from the sensor similar to our work in (N Al-Hoqani and Yang, 2015) or by reducing the frequency of communication of sensed value to the base station (Diallo et al., 2015).

In this chapter, we aimed to design and develop a front-end Graphical User Interface (GUI). The developed on-demand data acquisition sensing system is called On-Demand Query-based Sensing system (ODQS). We identified the set of network topology, database, and development tools first, and then introduced the performance metrics and scaling factors for the
evaluation of our test bed experiment. We deployed Sensinode cc2430 (Wauthy and Schumacher, 2010) in the testbed, which operates with Contiki operating system. We established the self-motivated programming with interrelated routing protocols (Goyal and Tripathy, 2012) and aggregation techniques as surveyed in (Dhasian and Balasubramanian, 2013).

In our WSN designed architecture, users input queries at the client computer front-end system in a natural, SQL-like language that defines their data demand and the way they aspire to combine, transform, and encapsulate these data at routing aggregator. As mentioned in (Diallo, Rodrigues and Sene, 2012), a particular query can trade-off the communication sensing cost to efficient communication and quality query outcomes.

To the best of our knowledge, this is the first work that fully distributes in-network aggregation at two levels of a sensor network routers and a sink node without any involvement of a gateway, streamer or DBMS server, integrated with the on-demand query-based approach. The work is tested in a real sensor network testbed Sensinode devices cc2430.

The WSN’s main purpose is to provide users with access to information of interest from data collected by spatially distributed sensors. In real-world applications, sensors are often deployed in large numbers to ensure full revelation of the monitored environment. Accordingly, such networks are projected to generate enormous amounts of data. The desire to locate and obtain information means the success of WSN applications is largely determined by the accuracy and quality of the extracted information. The principal concerns when extracting information include the timeliness, accuracy, cost and reliability of the extracted information and the methods used for extraction. The Information Extracting (IE) process allows unstructured data to be retrieved and filtered from sensor nodes using sophisticated techniques to discover specific patterns. Practical constraints on sensor node implementation such as power consumption (battery limits), computational capability and maximum memory storage make IE a challenging distributed processing task (Alsbouí & Hammoudeh 2011). Using visual studio development tools as middleware or engines to be able to communicate with the wireless sensor network’s sink to request any data from the connected sensor nodes, we have designed a Context-Aware Applications and Services platform.

In this chapter, we are addressing the problem of sending queries to specific nodes that have a significant effect on the query answers. We combine the Ad-hoc On-demand Distance Vector (AODV) routing features with query-based features to come up with an energy-
efficiency power consumption algorithm. Query-based approaches provide a high-level interface that hides the network topology as well as radio communication from end users. Queries can be sent on demand or at fixed intervals. They provide a solution if the data needs to be retrieved from the entire network. The literature review chapter discussed many limitations of the query-based approaches are, firstly, that most existing query languages do not provide a suitable construction to easily articulate spatial-temporal data characteristics. Secondly, it is difficult to formulate queries using current languages that represent higher-level behaviours or specify a subset of nodes that have a significant effect on the query answer.

5.2 On-demand processing

In this chapter, the concept of data processing with many-to-many information extraction characteristics is anticipated. This concept use query-based factors and data acquisition in in-network data processing structures in order to achieve the research objective. The primary objective is performing an accurate, with high-speed query, while saving energy at the sensor node. Those nodes have many different roles. They may act as a server, which we consider as net database, and it is highly recommended and primarily required. On the other hand it may act as a routing nodes or even a broadcasting nodes. The use of different in-network query processing techniques in handling only one query and generating the answer from one sensor is also considered. An in-network query response relationship category, which is one of the database relations for query management methods, is discussed as a core solution.

5.2.1 Query processing Framework

Chatzimilioudis et al. (2013) designed a query routing tree to determine how data is routed and processed in a network. According to the authors, there are two types of queries: universal queries and subset queries. In our query unit architecture, users input queries at the interface, which will be send to the sink, which acts as a server in a simple SQL-like language that describes the data they are collecting and how they wish to combine, transform and summarise it.

The framework for query processing is illustrated in Figure 5-1. This shows that the optimiser is in the sensor sink as well as a router to perform the aggregation task when requested. Moreover, it is responsible for merging the query packet as one message per query. Moreover, the framework works from sending query selected by users. After that the query attributes are merged into one query that will be ready to be broadcasted to the concern sensor
nodes. Once the query is propagated to the sensor networks, each sensor nodes waits for response to the query and the in-network query optimiser retrieved the desired packets readings and forward it to the sink to be then merge whole the answers into one packet. The last packet will be then sent to the user interface sensing system and saved in the external database for any further related queries as a historic records. The different query steps is explained in the following sections

![Diagram of query processing in WSN](image)

**Figure 5-1 Framework for Query processing in WSN**

### 5.2.1.1 Build Query

The query building function begins with variable declaration, query building, and aggregation operation selection. The aggregation variable such as maximum, minimum, and average stores the appropriate aggregation operation options (min, max, avg) of sensor reading values according to request selected by the user. This is achieved using the binary operator which works like an if-else statement but minimises lines of code and thus improves execution time. After the aggregation option has been stored, the data grid view is looped through with consideration given to only rows with at least one attribute selected. After looping through each row, the address of the sensor, together with its attribute, is stored in the variable declared earlier called query. Multiple rows are separated by a semi-column to depict multiple queries. Two functions help the merging to take place. First is the ‘MergeAddress’, which saves the sensor address and the second, ‘CheckedAttr’, which saves the attributes requested. They both merged together to form one single message.
5.2.1.2 Token Generation

The token generation is a randomised four-digit unique number generated and assigned to each query. The reason behind the generation is to make it easy to identify unique queries saved in the database. A class called Tokenizer, responsible for the token generation, is expressed in pseudocode.

5.2.1.3 Sending a Query

For our experimental monitoring scenario, the users make sure the input sensing system is ready for on-demand sensing, after the user connects, they select sensor attributes required such as readings selected (temperature, light, and battery), the sampling period, sampling interval and aggregation or leave them to their default values, the user then clicks to run the query. Three main things occur behind this button, as listed below. Each item is then explained in more detail.

1. Build Query: This function is responsible for constructing the query in a particular format before sending it to the node is invoked.
2. Token Generation: This is a class that generates and assigns unique IDs to each query. The purpose of this is to be able to track the result of queries stored in the database by their query IDs (Token).
3. Stopwatch and timer enabled: The timer records the round time between sending a query and receiving the result, while the timer provides multi-tasking capabilities as in the case of threads. The reason for using a timer rather than a thread is because the timer is easier to handle without the need for memory resources. (Dunkels, 2007).
4. The query was sent in one-line programming code in the Contiki environment in order to save energy by sending all the requested attributes with their conditions in one line.

5.2.1.4 Receiving answer to a Query

Both sending and receiving is done through the serial line connection, which is possible using the serial line control on visual studio.Net. The serial line communication is activated ones the user connects to a port. It then listens for any form of communication that it writes or reads.
5.2.2 Propagation of Query Statements

In this section, it is essential to explain that the sink sends the request containing the data specified by the end user via the sensing system. As an example of the queries we performed in our experiment, queries Q-1 to Q-4 and four below are query statements include conditions and attributes such as aggregation parameters for sample interval and query lifetime.

The universal queries require queries from all nodes to be answered and usually involve the application of a single aggregation operator to all data sources such as (Q-1), whiles subset queries only need data from a subset of nodes such as (Q-3) in which various operators may be applied to different groups of data sources.

This is a one scenario query for multiple query formats for either aggregation or non-aggregation. The query syntax statement (Q-1) shows the general SQL format for this thesis text-based query engine. This statement is a syntax used for the query to be sent to the sink and broadcast as requested by the end user, allowing the router node to forward the query to the sensor nodes as a destination.

\[
\begin{align*}
\text{SELECT} & \quad \text{sensor, light, temp, battery} \\
\text{FROM} & \quad \text{sensors1, sensor2, sensor3, sensor4} \\
\text{WHERE} & \quad \text{<condition>} \\
\text{SAMPLE} & \quad \text{INTERVAL 10s FOR 30m}
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{MAX (light), MAX (temp), MAX (battery)} \\
\text{FROM} & \quad \text{sensors} \\
\text{WHERE} & \quad \text{SAMPLE INTERVAL 10s} \\
\text{LIFETIME} & \quad \text{IS 100s}
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{MIN (light), MIN (temp), MIN (battery)} \\
\text{FROM} & \quad \text{sensors1, sensor2, sensor3} \\
\text{WHERE} & \quad \text{SAMPLE INTERVAL 10s} \\
\text{LIFETIME} & \quad \text{IS 1hour}
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{AVG (light), AVG (temp), AVG (battery)} \\
\text{FROM} & \quad \text{sensors1, sensor2, sensor3, sensor4} \\
\text{WHERE} & \quad \text{SAMPLE INTERVAL 10s} \\
\text{LIFETIME} & \quad \text{IS 1hour}
\end{align*}
\]
The above queries are triggered via the system interface for simplicity for the end user. For example, (Q-2), consider retrieving maximum light, temperature, and battery readings for all sensors, for 100 seconds, for sample interval or epoch of 10 seconds. Similarly other query statement (Q-3), and (Q-4) The different query engine blocks or models are discussed in different sections for different purposes such as query type (aggregation or not), query mode (merge or not), interval selections, sample rate selections and aggregation type selections. The data are merged using default settings for energy consumption and for quick response times for the queries. At the end, the query is propagated to the different sensor accordingly, to return with users result

5.3 Query Experimental Environment

For the on-demand query information extraction experimental testing, the experiment is designed based on the following environment: -

1. Sensinode hardware cc2430 board; seven nodes are used for different sensor node roles and programming for each.
2. Programming model hardware or dongles and c-like programming language.
3. The sygwin major software subsystems.
4. Contiki OS architecture.

5.3.1 Contiki Operating System

Contiki (Dunkels, Gronvall and Voigt, 2004), is a lightweight open source OS written in C language for WSN sensor nodes. Contiki is a highly portable OS that is built around an event-driven kernel. Contiki provides pre-emptive multitasking that can be used at the individual process level. A typical Contiki configuration consumes 2 kilobytes of RAM and 40 kilobytes of ROM. A full Contiki installation includes features like the multitasking kernel, pre-emptive multithreading, proto-threads, TCP/IP networking, IPv6, Graphical User Interface, web browser, personal web server, simple telnet client, screensaver, and virtual network computing. In the following subsections, we shall explore the Contiki OS in more detail.

Contiki shares the hardware among different applications and users through processes and protocol threads. It is important to understand how processes are scheduled and how communication between processes works, because the development of new applications, or even new low-level protocols, is made using the processes framework. Contiki has two different execution contexts: cooperative and pre-emptive. The first one is used in the context of regular
execution, where functions are sequentially called and executed in the microcontroller. The pre-emptive context corresponds to the execution of interruptions, due to I/O or timers. When executing in a cooperative context, all tasks are executed until their completion. Pre-emptive activities may, however, interrupt the cooperative code at any time.

5.3.2 Sensor Tree Structure Technology

The network model for our work is the most vital part in terms of achieving the requirement for high data quality and a fast query response. This section will cover the approaches to network topology and the tasks performed in the network to accomplish the multi query-processing objective.

Our network database model consists of four layers: Sensor nodes, Router nodes, On-demand query interface, and a sink. When sending a query packet to a WSN, the sink queries a specific area or a point of interest in respect to the sensed data in the sensing field, where sensor nodes are densely and uniformly deployed. According to Yan et al. (2015), query scheduling is one of the most important mechanisms used in query processing. In their work they used a Multi-Regional Query approach. In this, when a query is placed, it is first scanned, parsed and validated. An internal representation of the query is then created, such as a query tree or a query graph. Then alternative execution strategies are devised for retrieving results from the database tables. The process of choosing the most appropriate execution strategy for query processing is called query optimisation.

5.3.3 Wireless Sensor Networks Design

Wireless Sensor Networks are a group of specialised autonomous sensors and actuators with a wireless communications infrastructure, intended to monitor and control physical or environmental conditions (Yang and Cao, 2008). Due to their nature, wireless sensor networks are noted for their low data rate, low energy-consumption and short-range network communication, which make them useful in tracking and monitoring network enabled objects in the physical world. A wireless sensor network is never achieved without the use of wireless sensor nodes. A typical wireless sensor node is made up of the following:

- A computing subsystem, which consists of a microprocessor with memories and a communication subsystem, which consists of a short-range radio for wireless communication.
• A sensing subsystem, which links to the physical world and consists of a group of sensors and actuators.

To perform their functions successfully and efficiently networks need to be well-designed and organised. This section will therefore look at the many factors that must be considered in order to build such a network, such as performance, resilience and scalability. This section of the thesis is essential for the project and its sustainable development. It will discuss in-depth the hardware and software required to build a wireless sensor network. More importantly, the network topology will be clarified in order to understand the role of each device.

5.3.4 Sensor Node Hardware

In computer networks, hardware refers to the networking devices such as routers, switches and cables. In order to build a wireless network, however, cables will no longer be needed. Devices can communicate with each other wirelessly. The standards that are often followed, which are created by the IEEE Standard Association, use radio frequencies to transmit and receive data. Since this project aims to build a wireless sensor network that uses all of the basic principles of wireless networking, such as the receiving, transmitting and relaying of data, it is necessary to have a device that can also perform each of these tasks. Typically, sensor nodes are small, cheap, energy efficient and are able to communicate wirelessly. The Sensinode cc2430 hardware, shown in the figure below, was chosen for this project as it meets these requirements.

Typically, sensor nodes include microprocessors, memory, sensors, power supplies and radio frequency transceivers. A transceiver is a device capable of both transmitting and receiving data. The transceiver within the Sensinode device supports a frequency band of 2.4 GHz using the IEEE 802.15.4 standard. This wireless node provides high performance and low power with 32, 64 or 128 KB in-system programmable ash, 8 KB RAM, 4 KB of which has data retention in all power modes (Texas Instruments, 2011).

The cc2430 device in figure 5-2 has built-in non-volatile program memory, and programmable read and write lock portions of flash memory for software security. In order to bypass the lock, an adapter is needed, which in the Sensinode cc2430 comes in the form of a small black N740 dongle that is plugged into the back side of the device. Finally, a micro USB data cable is required to connect the Sensinode to a PC in order to access the Sensinode’s memory.
The Sensinode cc2430 has some built-in sensors, including a battery monitor, temperature sensor, light sensor and accelerometer. Some of these sensors will be used in the project to ensure that the best performance is being achieved. The Sensinode cc2430 also supports a Received Signal Strength Indicator (RSSI), which will be used in this project to determine the best route for packets to be sent. The following figure provides a functional diagram of the Sensinode cc2430 and how the components are integrated.

The sensor router acts as the mediator between the sink and the end or sensing nodes. Its main function is to route the packet between the sink and the sensing node. Various algorithms aid in the forwarding of the packet to its destination and back to its source.

5.3.5 Architecture of the WSN in the Setup

Our network database model consists of four layers: sensor nodes, router nodes, on-demand query interface, and a sink. By sending a query packet to a Wireless Sensor Network, the sink queries a specific area or a point of interest in the sensing field, where sensor nodes are densely and uniformly deployed, but this doesn’t mean that if sensor nodes are not densely and uniformly deployed, no queries can be received, it is only for the simplicity of the experimental testing. According to Yan et al. (2015), query scheduling is one of the most important mechanisms used in query processing.

Figure 5-3 and 5-4 demonstrate the two different wireless sensor networks architectures that drive the algorithms used in chapters six, seven and eight of this thesis. The first one is a tree-based static network structure, whereas the second one is a graph-based dynamic network structure. The advantages of having both architecture are the different sensing systems requirements. The tree structure is mostly use for small sensor network projects, whereas the graph-based is useful for huge and complicated sensing projects. The disadvantages is that the static nature consumes more energy than the graph-based and it more accurate, on the other hand the graph-base also takes more time to perform the routing and aggregation tasks, but
aggregating makes it less accurate also. Each structure could be applied to a specific scenario of sensing systems, more accuracy or more energy efficiency.

![Figure 5-3](image1.png)  
**Figure 5-3** On-demand tree-based query architecture for WSNs

![Figure 5-4](image2.png)  
**Figure 5-4** On-demand graph-based query architecture for WSNs

### 5.4 Sensor Network Database model

A distributed database is a set of interconnected databases that are distributed over the computer network or internet. A Distributed Database Management System (DDBMS) manages the distributed database and provides mechanisms to make the databases transparent to users. In these systems, data is intentionally distributed among multiple nodes so that all the computing resources of the organisation can be used optimally. The four basic operations in any query on a database are create, retrieve, update and delete. There are similarities between traditional databases and sensor network databases. In this experimental case study, we are
able to create new query whenever user needs to sense a specific sensor reading. The manipulation operations, which includes update or delete any record from the sensor memory. We could also manipulate the sensor net database via systematic sensing. From another point of view, the data are to be collected according to the user inputted parameters and operations. The system is explained more in the next two chapters for intensive discussions and evaluations

5.4.1 Sink Aggregation code

The implementation was carried out on two separate levels. The first level was to aggregate the requested attribute in the sink, as this is the last destination before the data is forwarded to the user. At this stage, the requested data are collected, then the sink performs the users’ selected aggregation function; after that, only the final result will be sent to the end user, and one single packet will be forwarded. Pseudocode for the algorithm implemented for sink aggregation illustrates the steps followed as a result of the users’ selections of aggregation functions.

Real-code 1: Sink node code in Sensinode cc2430

```c
case 1:
    avTemp=0; avlight=0; avbatt=0;
    //Aave(data);
    // printf("counter: %d\n", counter );
    for (m=0; m<MAXROUTES;m++){
        if(qTable[m].address[0]! =0x00){
            printf("node- count: %d\n", (int)qTable[m].nodeCount[0]);
            bSize+=(int)qTable[m].byteSize[0];
            avTemp+=(((qTable[m].temperature[0]<<8)+qTable[m].temperature[1])*qTable[m].nodeCount[0]); //add *count
```

5.4.2 Router Aggregation codes

This section is concerned with the aggregation functions that are performed at the router level (max, min and average). The values to be sensed from the environment include temperature, light and battery consumption.
In this section, we have proposed a query-based data aggregation technique to efficiently aggregate the data from the given set of sensors at the base station and the router level. In our model, the base station queries each sensor to transmit its collected data, thereby reproducing the status of the area that is supervised by the queried sensors. With the order of tasks in each sensor defined as sense, process and transmission, each sensor begins sensing the data after receiving the query from the base station. The worst-case scenario is observed when all the sensors send their collected data to the base station simultaneously, potentially leading to a loss of data. To avoid the overlapping of data transmissions at the base station, we have embedded the query-based data aggregation model at the routing level using dynamic programming for the nodes. Moreover, we incorporated a model to get the data using an inter-query approach. This approach will save more of the sensor’s energy.

We refer to the on-demand query-based system as an ad-hoc query, and also a self-tuning and autonomic optimiser: An on-demand approach is a form of many-to-many relationship in the evaluation of database queries in which several different queries are executed concurrently on multiple processors so as to improve the overall throughput of the system. The ad-hoc querying is similar approach, but it has not been studied in querying sensor data from a user point of view. The user selects query parameters from sensing system interface.

5.5.1 Sensing System Interface

The computer is the source of the query, but software must be created for that purpose. In the project, the software was developed in C# using the Microsoft Visual Studio.Net

```csharp
switch (aggregation){
    int avTemp,avlight,avbatt;
    case 1:
        avTemp=0; avlight=0; avbatt=0;
    for(m=0;m<MAXROUTES;m++){
        if(qTable[m].address[0]!=0x00){
            avTemp+=((qTable[m].temperature[0]<<8)+qTable[m].temperature[1]);
            avlight+=((qTable[m].light[0]<<8)+qTable[m].light[1]);
            avbatt+=((qTable[m].battery[0]<<8)+qTable[m].battery[1]);
```

5.5 Query-based Sensing System

The computer is the source of the query, but software must be created for that purpose. In the project, the software was developed in C# using the Microsoft Visual Studio.Net
framework. The software sends and receives queries to and from the sink node and stores a copy of the result in JavaScript Object Notation (JSON) file format. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, and it can produce different codes such as native or managed interface query engine.

The Sensor Source Form

There are two menus on the user interface, namely sensors and queries, which, when clicked, link to their respective forms. Figure 5-5 shows the query request form that used to query any data. The forms were designed for our experimental purpose; meaning that the queries are controlled and organised in a way that saves much energy. The details of the experiment cost and energy functions are discussed in more detail in chapter 6.

![Figure 5-5 On Demand Query interface](image)

SELECT node id, light, temp
FROM sensors

The end users select the query attributes such as Temperature, Light and Battery. Also, the user needs to specify the sensor name. The sensing system also has an on-demand period schedule as well as an interval scheduler. The user can select 1 second, 10 seconds, 30 seconds, 1 hour, 2 hours, or 1 day. The user can decide whether or not to select an aggregation feature and the different aggregation types are listed (MIN, MAX or AVG). After that, the user clicks on the Run Query button to send the query to the sink node for processing. Since this is a real-time processing, the user will be able to see the data when processed, according to the set criteria as mentioned above.

In Figures 5-6 and 5-7, the Temperature tab and Light tab are used to view the data sensed from the room environment in real time.
In Figure 5-8, similarly, the battery consumptions for different sensors are monitored by the end user. The user can see the data in either a grid or tabular view or a graph view to ease the readings. The designated query is also written for the user to indicate his requested data from the deployed sensor networks.

Figure 5-9 illustrates the data view to compare the monitored environment for all the gathered attributes and sensors. The legend on the right shows each sensor name.
5.5.2 Sensor Storage Database

The storage package for the sensed data or resulted queries is stored simultaneously in a JSON file for historic purposes. The figure 5-11 shows the query result for the sensed records for any selected query by the user in our experiment. To perform the testing for our experiment, the query selection was alternatively selected for different queries by the end-user. Different queries where performed for different parameters according to query statement (Q-1), which results in table 5-1 We sensed more than 2000 data records for different queries and different requirements, such as an aggregated query or normal simple query for the several sensors. We have included very important data in the table such as query start date and time, query end date and time and also the elapsed time the query used to perform the in-network process. In addition, we also added the energy consumed in term of bytes per query, as in figure 5-10.
Figure 5-10 JSON format query-based storage

Figure 5-11 it illustrates the different stages in sequence starting from the stage when it is requested by the end user until it is back to the end user again.

![Diagram showing query result form for comparing all monitored data]

**Figure 5-11** Query result form for comparing all monitored data

### 5.6 Dynamic Programming Methodology for On-demand Query Processing

Dynamic programming, also known as a dynamic optimisation, is a method for solving a complex problem by breaking it down into a collection of simpler sub-problems, solving each of those sub-problems just once, and storing their solutions, ideally, using a memory-based data structure. Dynamic programming is both a mathematical optimisation method and a computer...
programming method. In both contexts, it refers to the process of simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively; Bellman called this the "Principle of Optimality". Likewise, in computer science, a problem that can be solved optimally by breaking it into sub-problems and then recursively finding the optimal solutions to the sub-problems is said to have optimal substructure. If sub-problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problems. In the optimisation literature, this relationship is called the Bellman equation. (Sniedovich, 2010)

5.6.1 Query Communication Cost

First, we focus on region-based aggregation queries, such as SUM, MAX and AVG. More specifically, these belong to the category of distributive and algebraic aggregation queries, as defined in Madden & Franklin (2002). A query region denotes the geographical area of interest for the query. Thus, a region-based query can identify a fixed set of nodes that are involved in the query and hence simplify the cost estimation and sharing estimation per query.

We assume that the sensor nodes are deployed uniformly in a network structure of $n \times n$ two-dimensional grid, with the base station being node 0 and n is being the total number of nodes. The query cost is calculated by the number of bytes consumed at each level the packet is transferred from any number of routing levels, as has been discussed in the previous chapter.

5.6.2 Energy Consumption

In this chapter, we are addressing the problem of sending queries to specific nodes that have a significant effect on the answers. We combine the AODV routing features with the query-based mechanism features to come up with the most energy-efficient power consumption algorithm. Query-based approaches provide a high-level interface that hides the network topology as well as the radio communication details from the end. Queries can be sent on demand or at fixed intervals. They provide a solution if the data needs to be retrieved from the entire network. The limitations of the query-based approach are: firstly, most existing query languages do not provide suitable constructs to easily articulate spatial-temporal sense data characteristics. Secondly, it is difficult to formulate queries using current languages that
represent higher-level behaviours or to specify a subset of nodes that have a significant effect on the query answer (Alsbouï et al., 2012).

5.7 Evaluation and Performance

Apart from the sensor’s performance, we need to evaluate the output according to the techniques used. The first one is the aggregation, and the second is the merging of the query for sending and receiving it as one packet instead of many.

5.7.1 Aggregation

Using the aggregation method for query processing is very effective and leads to value added in terms of energy saving as we have the data in one set instead of many values sent in-line with the query requested. Below are the equations Eq. (5-1) and (5-2) for the aggregation and the non-aggregation functions respectively. Table 5-1 sets out the parameters used to explain the cost formula in next section.

\[
\text{Agg} = \sum (b) \cdot (h) \quad (5-1)
\]
\[
\text{No-Agg} = \sum (b \cdot s) \cdot (h) \quad (5-2)
\]

Table 5-1 Parameters for query cost

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>No. of sensors</td>
</tr>
<tr>
<td>h</td>
<td>No. of hops moves</td>
</tr>
<tr>
<td>b</td>
<td>No. of bytes per packet (30)</td>
</tr>
<tr>
<td>Bytes</td>
<td>30 bytes is the cost of sending a message per query</td>
</tr>
</tbody>
</table>

5.7.2 Merging

Merging the query attributes requested by the user into one packet is the main task that this query engine is undertaking. This means that the data from different sources or sites are combined before the resulting packet gets to the final destination. Merging the data requested and retrieved means that the packet is sent only once instead of sending the different requests individually. In contrast to the non-merged approach, the data is directed forwards and backwards only once. At this stage, we can say that the merge is our default setting for requesting any single query. By specifying this, we are guaranteed some saving to the lifetime
of the sensors, which is merging more than one query set into only one. The contributory chapters will explain this concept more clearly.

The non-merged approach, on the other hand, involves sending the request as a query at a time for the specified sensing period and the specified sampling interval. This will be performed as many times as necessary while waiting for the termination of the query. The assurance of saving more energy in this scenario is lower than in the previous one. This is because of the requested sensing period or the query lifetime. In other words, the lifetime of any query will be considered as one query although it has been executed according to the specified period, such as 1 hour or 1 day.

Summary for the dual technique (aggregation and merge)

We added the aggregation (MAX, MIN and AVG) to the merge as a default selection, in order to obtain the best results. It is very clear from Figure 5-12 that the aggregation query saved more energy than the non-aggregation query. The value for the aggregation is only 2000 bytes compared to the value consumed for the non-aggregation, which was recorded as being 4000 bytes. There was a savings difference of exactly 50% as a result of using aggregation techniques to query the data from the sensor networks than using the non-aggregation query for sensing the data.

Figure 5-12  Energy consumption line graph for Q query

In the query result figure, it is also drawn to our consideration that the total result for bytes consumed is equal to the total query requested packets. The output produced shows about \(47 \times 4 = 188\) bytes were expected to be collected, but we received only 155 total results. To conclude, \(188/155 \approx 6\%\) packets were lost or failed to be sensed, equal to 33 packets lost. From
another angle, approximately 18,900 bytes are consumed, compared to the expected
188*120=22,560, which means we are not wasting energy if packets are lost. This calculation
express that the lost packets didn’t consume any bytes, only response query answer packets
reached to the base station are consumed some energy, according to the above energy equations.

The performance of our query processing was also tested by executing the query using
both the merge and non-merge techniques, and then calculating the differences between them,
then the results are expressed as a percentage. The generated analyses results drawn from
numbers of queries by different query user's request and commencing all the sensor nodes for
all operations proposed in our query sensing system.

In terms of performance, we thought the higher impact would be on the response time
for any performed query. Figure 5-13 offers a comparison between two queries with the same
attributes, but after applying aggregation to the second query. It is very clear that the aggregated
query takes less time than the non-aggregated query. The first query response time is recorded
to be 108 seconds, which is 27% less than the second query response time. On the other hand,
the second query took 121 seconds, which is around 13 seconds more to perform the
aggregation function, which is only 27% higher than the non-aggregated query.

![Figure 5-13](image)

**Figure 5-13** Aggregated and none-aggregated query total elapsed time

From Figure 5-13, it is observed that the total elapsed time per query improved
significantly for both queries. It is noticed that the none-aggregated query was slightly faster
than the aggregated query, at 108 seconds as opposed to just under 121 seconds (i.e. about 11%
quicker). The performed result of nearly equal response time for the optimised query using
aggregated data packet and merge packet as well. After the operations functions are discussed,
the rest of the query clauses to be discussed next.
5.7.3 Query lifetime clause

In the selected query, as any query needs to be terminated at some point we added a requirement for the lifetime for the query to be terminated after 1 hour, where period = (1time*interval, 10min=10*interval, 30min*interval, 1hour*interval, 1day*interval). This will terminate the query after the selected interval, as well as the selected aggregation and merge choices.

5.7.4 Sampling interval clause

In our sensing system, we added the features of the setting of the sampling interval. As a part of this thesis achievement, we have performed the adaptive sampling algorithm which we implemented in Chapter 4 in the form of TCP-based adaptive sampling. Having a higher sampling will of course increase the energy saving, and this will have an impact on the query sensing performance. The major goal is to sense the sensor readings only when the prediction model measures any unexpected event in the AS-TCP model.

5.7.5 Dual aggregation level

In our sensing system experiment, the aggregation is tracked at the routing level nodes as well as at the sink level node. This results in a well-filtered result before it is executed at the sensing system interface. Our aggregation strategy is performed at all routing trees level as long as queries structures move deeper, and data requested are more than from two sensors. This is very clearly a good result in terms of a very high degree of energy saving across all the network nodes and adding more nodes will not increase the consumption of energy. The savings will therefore be higher as we add more nodes.

To sum up according to the tested queries and using the equations discussed, from Figure (5-12) combining the query attributes will save the energy consumed by more than 50%, as it is assumed that only 30 bytes per message will be spent, regardless of how heavy the query is, with all the attributes and condition identified. From these assumptions, we can conclude that the merger will out-perform the non-merge of the sensing system by 50% or more in the long run. After testing and implementation, we are to conclude that the provided above real codes 1 and 2 in section 5.4 are only to be used for a child connected to the router, but not the other second level routers.
5.8 ODQS Architecture

The proposed ODQS system architecture was implemented for demonstrating a real-time data-query technology, energy saving efficiency and the data quality. The architecture used AODV routing protocol because it adapts to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the network (Goyal and Tripathy, 2012).

5.8.1 ODQS Sensing Testbed Design Architecture

The testbed, as shown in Figure 5-14 involved a computer, a sink node, two routers and four end nodes. The sink, routers and end nodes are wireless sensors from Contiki Sensinode cc2430, which can be powered by two 1.5 volts’ batteries. The sink node was attached to a computer, and the routers were connected to the sink and each other. Each router wirelessly connected to two end nodes. Three level tree structures were used for the implementation.

The system works starting from the query interface through the Contiki operating system for sending the data request by the end-user. The data request is sent in one line considering all necessary attributes via the query engine, which transfers the data request into a text-based format. The Contiki sensors database (DB) is connected to the sensor sites, which performs the aggregation functions at the different levels. The results then sent back to the user, processing the query.

Figure 5-14  ODQS Wireless Sensor Networks testbed design.
5.8.2 Query Processing Engine Design

The user sends queries to the sink node through the query engine interface. The sensing devices are attached to our ODQS application interface, which is developed for sensing efficient and quality data determination. Figure 5-15 illustrates the three levels of the ODQS.

![ODQS System Three Query Levels](image)

Figure 5-15  ODQS system three query levels

The ODQS query flow diagram is presented in Figure 5-16. The query is managed by the sink query engine process. The developed sensing functions package is implemented on each node either, sink, router, or sensor node. That will give the nodes the database functionality. The figure explains the ODQS query processing phases in sequence, commencing the query requested by the end user up to response results directed back to the end-user again.

5.9 Implementation

This section focuses on the implementation of the ODQS. We used visual studio as programming tools for the front-end and the C programming to encode the sensor nodes. As a prerequisite for our sensing system, the sensor nodes are registered by their MAC addresses, location, name, and any other details to be able to communicate.
5.9.1 On demand Sensing System Development

The designed GUI is divided into a set of views, as demonstrated in Figure 5-17. It begins with a section indicated by port this displays the available communication ports in the pc connected via sensor nodes. Beneath the port selection is a query builder view which lists the entire sensor’s name and address registered, also lists the attributes to ease the user selections.

Following the query builder is the period view where the user selects the query execution recurrence. Following is the intervals block, where the user can select the time interval at which the sensors should be queried. The default is set to 10 second runs up to 10 recurrences. The next view in the GUI is the aggregation, where the user can choose an aggregation to be performed on the query. The default option is none-aggregation.
The aggregation technique has various options such as maximum, minimum and average. Moreover, merge query technique is the default setting in our query engine. This technique combines query results on the sink node before sending it to the computer. After aggregation comes, the button that runs the query engine for transforming the queries to the ContikiOS. Underneath the button is a text box that displays debug data from the sink node.

On the right-hand side of the interface is a data grid-view that displays the query results, with columns for sensor name, address, and sensing values for temperature, light and battery, as well as a timestamp in seconds within which the execution was made, and the kind of aggregation that was performed on the query. Finally, we have the real-time charts with various tabs for query acquisition monitoring persistence. The ODQS system generates the bar chart on real-time sensing that compares the various attributes of each sensor node, representing them in different colours.

According to S. R. Madden et al., (2005), lifetime-based queries are ways users can select a duration or period for their queries whether in minutes, hours, days, weeks or months and gives an intuitive way by which users can understand how the adjustment of a query’s lifetime influences power consumption. That means a user can influence the interval at which the query should run by making an adjustment on the interval, for instance, running the query at time intervals of 10 seconds for the entire day. In our GUI, there are options available for the user to select both the lifetime of the query as well as the interval at which these queries are to run until its lifetime, period or duration expires. The query termination is performed by the lifetime operation specified.

5.9.2 Query Processing Functions

We designed the interface with regards to sending queries from the sink, dispatched them to the routers, and then to the sensor nodes. Then, we developed efficient query functions at different network levels. The functions are effective at different locations to execute query efficiently at the proposed query engine. Table 5-2 summarised the deployed function as the main contribution to our scheme. The sensor devices are encoded with specified functions as discussed in previous sub sections of section 5.7. For example merging means concatenate the data from more than one sensor nodes to a single line packet by the sink to send it to the end user interface sensing system and then to a historic database.
### Table 5-2 ODQS Implemented Query Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>PC</th>
<th>Sink</th>
<th>Router</th>
<th>Sensing node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Query</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query Processing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aggregation</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Attributes Sensing</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Query result storage</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5.9.3 Query Plans Structure

In this section, we will be working on spanning tree, which is one of the traversing algorithms to implement the best tree query structure that saves energy for our query optimizer. In this experiment, the query tree structure was selected according to the signal strength of the sensor networks. As long as the signal of a node is stronger than the other, then the packet will be forwarded to the next closest level. The short tree path will be selected if the network signal strength is equal, in order to overcome the collision that might occur instead. For example, if two router sensors are connected to a specific sensor node, the query processing engine will select the short path to broadcast the sensed data.

#### Multiple Query Processing

Multiple sensing requests from the end-users, who are sending to multiple sites or many destinations, go back to the same base station. The router has the feature of query resource sharing structures, which means that if the data has been sensed recently and a copy is available in the routing table, the user can still use that, instead of repeatedly re-sensing the data.

#### Merge and Non-Merge Query

The merge and non-merge model used for the presentation of query result from the sink to the computer. It was designed to either merge the result or send them one after the other. By default, all queries are merged and by way of merging the query results by the sink before sending it through the serial port are intended to reduce execution time thereby reducing the amount of energy in transmitting the result one after the other. This was demonstrated in the project where the execution time for merged query result is lesser than non-merged query results.
5.10 Experimental Methods

For the experimental methodology, several data processing operations were introduced and implemented specifically sampling interval, data aggregations, lifetime-based, and merging operations. Our router aggregator is performing the function for all incoming queries.

5.10.1 Query Processing Methodology

Various data processing operations were designed and implemented namely, data aggregation, lifetime-based queries, merging and non-merging. Three data aggregations were implemented, including maximum, minimum and average. The aggregations were performed on the sensed data being temperature, light and / or battery values. Once more, the aggregation was done on both sink and router levels; this was to reduce energy compared to non-aggregated data. The following is a design with brief explanations of the aggregations.

Maximum Aggregation

Maximum: This is the maximum aggregation performed on the incoming query results. In Figure 5-18, the child nodes (end or sensor nodes) send their requested query result to their parent node (router). The routers select the maximum values from the two set values, which came from the child nodes. The routers perform the first aggregation to give a new single set of sensed values, which are transmitted to the sink node. The sink node, upon receiving the aggregated values from the two set query results from the routers, also performs an aggregation to select the maximum sensed values, which is made available to the computer through the serial port and presented on the Graphical User Interface (GUI).

![Diagram of Maximum Aggregation]

Figure 5-18 Maximum aggregation function
**Minimum Aggregation**

Minimum: Just as with the maximum, in Figure 5-19, the child nodes (end or sensor nodes) send their requested query result to their respective parent nodes (routers); the routers select the minimum values from the two set values that came from the child nodes. The routers perform the first aggregation to give a new single set of sensed values, which is transmitted to the sink node. The sink node, upon receiving the aggregated values from the two set query results from the routers, also performs an aggregation to select the minimum sensed values, which is made available to the computer through the serial port and presented on the Graphical User Interface (GUI).

![Figure 5-19 In-network minimum aggregation](image)

**Average Aggregation**

Average: Unlike the maximum and minimum aggregations, the average takes a slight twist in averaging the sensed values. In Figure 5-20, the average aggregation is performed on the sensed values. In the figure, the end nodes present their sensed values to their respective routers. The router performs an average aggregation by adding up each sensed value from each end node (sensor nodes) and dividing it by the number of end nodes (N) whose sensed values were received, which in this perspective is 2 (N=2). After each router performs the aggregation, the resultant values are transmitted to the sink node together with the divisor (N).
The query result is received by the sink node, which also performs average aggregation. Rather than adding up the values and dividing by the number of nodes (routers), there is a uniform formula that intends to favor both an equal and an unequal number of sensing nodes. To achieve this, each sensed value is multiplied by the number of sensor nodes, whose values were acquired from the router, before adding up with subsequent results from other router(s) and finally dividing the result by the total number of sensor nodes (T), who’s sensed values were received. Formulas (5-3), (5-4) used for averaging function in the router and sink node respectively.

\[
\text{Router: } \frac{(v_1 + (v_n))}{N} \\
\text{Sink: } \frac{((v_1) \cdot N) + (v_n \cdot N)}{T}
\]

Where N is the number of nodes whose sensed values were received, T is the total number of sensing nodes whose values were received, v is the value (s), and n is the last value. However, it must be noted that should the tree grow, the formula used in the router only applies to the direct parent (router) of the child nodes (sensing nodes), and that all other levels of the tree will use the formula in the sink node.

5.10.2 Data Aggregation Functions

Three aggregations are implemented in our ODQS include maximum, minimum and average aggregations. These aggregations are performed on the sensed data temperature, light and/or battery values. They are applied to the values at both sink and router levels. This is to reduce energy compared to none-aggregated sensed data. The following is the design with
detailed explanations on the aggregations. The principle of the maximum-minimum and average aggregation function is drawn in Figure 5-21 with different readings deliberated.

\[
\begin{align*}
N = 2, & \quad T = 4 \\
\text{Temp:} \frac{(23.5 + 25.2)}{N} & \quad \text{Light:} \frac{(94.7 + 74.8)}{N} \\
\text{Battery:} & \quad \frac{(2.3 + 3.3)}{N}
\end{align*}
\]

\[
\begin{align*}
N = 2, & \quad T = 4 \\
\text{Temp:} \frac{(29.5 + 25.2)}{N} & \quad \text{Light:} \frac{(94.7 + 74.8)}{N} \\
\text{Battery:} & \quad \frac{(2.3 + 3.3)}{N}
\end{align*}
\]

Figure 5-21 Aggregation Principle

Where \(N\) is the number of nodes whose sensed values were received, \(T\) is the Total number of sensing nodes whose values were received for temperature, light or battery. However, it must be noted that should the tree grow, the formula used in the router only applies to the direct parent or router of the child sensing nodes; all other levels of the tree will use the formula used in the sink node.

5.10.3 Merging Queries

The second data reductions technique is the merge technique, the main task that our query engine is undertaking is to combine the different packets according to the query requested by the end-user and concatenate all queries together into one packet only to save time and energy for sensing the required data from the sensor devices as shown in Figure 5-22. The query attributes requested by the user are consistently concatenated into one single request, and therefore one message packet is transmitted. It means that the data from different sources or sites are combined before the resulting packet gets to the final destination. As a result of merging the data requested and retrieved, the packet is sent only once instead of sending the different requests individually. At this stage, we can say that the merge is our default setting.
applies to every single query demanded. Specifying this, we assure a lifetime saving for the sensors.

<table>
<thead>
<tr>
<th>33.05</th>
<th>24.5</th>
<th>89.7</th>
<th>2.3</th>
<th>33.06</th>
<th>29.3</th>
<th>107.3</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.07</td>
<td>32.5</td>
<td>99.62</td>
<td>3.1</td>
<td>33.08</td>
<td>21.7</td>
<td>69.76</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Figure 5-22 ODQS Query merge model

In contrast to the none-merge technique, the data is directed forwards and backwards only once. The merge out-perform the none-merge of the sensing system with the none aggregated queries, because the router aggregator waits for all attributes to perform the aggregation before unicasting, and that is clearly time-consuming.

5.10.4 Query communication cost

The communication cost in this experiment is dependent on the data type requested and whether or not aggregation is performed; the number of bytes consumed to execute each query will depend mainly on the processing type, and the number of messages varies from one query to another. For instance, sending all temperature readings from all the sensors - assuming there are five sensors - is not sensing only the average reading from all the five sensors. Since we have assumed that the query packet size is 30 bytes because we wanted to include the maximum readings and assume complex query is requested by the end-user also, then the 30 bytes will be consumed at each routing level. Compared to the aggregation and non-aggregation query type, the fewer messages passing from one level to another, the less the waste of energy incurred. The following two equations explain the expected result from the query types.

Aggregation = 30 bytes * no. of routing level

Non-aggregation = (no. of sensors*30 bytes) + (no. of routing level*30 bytes)

In these experiments, we explored the scalability of MODSS (Multiple On-Demand Sensing System). To measure this scalability, we calculated the packet received a ratio of the downward transmission and upward transmission in the following chapters.

This chapter was concerned with wireless sensors networks query processing systems. We presented the typical architecture of such systems. Then, we proposed a new query
processor system call an On-Demand Query based Sensing system (ODQS). Contrary to existing systems, our ODQS system supports the developed protocols, 6lowpan and RPL, which have enabled wireless sensor networks to become reachable on the global internet prototype network. To sum up, regardless of how heavy the query with all attributes and operations was identified, concatenation saved significant energy.

5.11 Results

In our sensing system experimental result, the aggregation is yet pursued at the routing level nodes as well as at the sink level node. Consequently, a well-filtered value was gathered before it is displayed at the GUI. Our aggregation strategy is performed at all routing trees as long as we move deeper in complex network architecture. Furthermore, adding more nodes will not increase the battery consumption because of routing tasks, which is responsibility of aggregating and merging. Therefore, energy efficiency is maintained while increasing sensor nodes using the proposed sensor architecture, while maintaining the accuracy as well. The tree structure distributes the load among the network structure and maintain energy consumption balance among all devices.

5.11.1 Discussion of query engine outcomes

To test the functionality of the developed sensing system, we have generated the query results for none aggregated queries with a fixed sampling interval of 30 seconds and we set the sensing delay 3 seconds in our query engine. Table 5-3 and Table 5-4 show the outcomes with some lost packets. Furthermore, the identified queries are reflected by the broadcast.

<table>
<thead>
<tr>
<th>Total Queries</th>
<th>Sensed Results (Packets)</th>
<th>Expected Results (Packets)</th>
<th>Total Bytes Consumed</th>
<th>Total Packet lost</th>
<th>Packet lost %</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>39</td>
<td>40</td>
<td>18720</td>
<td>1</td>
<td>2.56</td>
</tr>
<tr>
<td>20</td>
<td>77</td>
<td>80</td>
<td>36960</td>
<td>3</td>
<td>3.90</td>
</tr>
<tr>
<td>40</td>
<td>153</td>
<td>160</td>
<td>73440</td>
<td>7</td>
<td>4.58</td>
</tr>
<tr>
<td>60</td>
<td>225</td>
<td>240</td>
<td>108000</td>
<td>15</td>
<td>6.67</td>
</tr>
<tr>
<td>80</td>
<td>296</td>
<td>320</td>
<td>142080</td>
<td>24</td>
<td>8.11</td>
</tr>
<tr>
<td>100</td>
<td>372</td>
<td>400</td>
<td>178560</td>
<td>28</td>
<td>7.53</td>
</tr>
<tr>
<td>120</td>
<td>448</td>
<td>480</td>
<td>215040</td>
<td>32</td>
<td>7.14</td>
</tr>
</tbody>
</table>
The above tables compare the factors for the actual sensed results with the expected number of packets out of each send query to the sensor network according to the users query. From the energy point of view, to measure the energy in joules as usual, we need to reproduce the records by multiply it by the bytes by 0.027 joules to convert the bytes to joules according to our Sensinode device.

5.11.2 Routing Aggregations Results

To measure the cost of the query as a message Eq. (5-5) and Eq. (5-6) presents the equation for the cost in terms of message size per packet for the aggregation and the none-aggregation query respectively. Where H is the number of query hops, K is the number of aggregated query requested, and B is the single bytes consumed per packet, which is equal to 30 bytes per query in our experiment. Hops are calculated by multiplying a number of sensors N and number of routers K.

\[
\text{Aggregation} = \sum ((B \cdot H) - (B \cdot K)) \quad (5-5)
\]

\[
\text{None-Aggregation} = \sum (B \cdot H) \quad (5-6)
\]

At the query cost calculation stage, when the user's request aggregated query, then the routing hops are counted and the number of aggregated query routers is also counted. As a result of subtracting the total number of aggregated queries from the total number of hops, which is mainly only total number of query hops multiplies by the total packet bytes. Whereas for none aggregated queries it is only the sum of all routing hops performed for each query multiplied by packet size.
The query results are plotted in Figure 5-23 to precisely explain that the total numbers of queries requested are very close to the expected answer. The lost packets are recorded to be less than 8.5% only. The packet transmission is calculated in terms of bytes per message or query. Therefore, the higher the packet transmitted, the higher the consumption of the battery. This resulted in saving energy by merging the packets into one query. The consumed bytes are also considered for evaluating the sensor lifetime in the evaluation section.

![Figure 5-23](image)

**Figure 5-23** None-Aggregated query growth versus energy consumed.

Using the aggregation technique for query processing is very effective and leads to value added in terms of energy saving, as we have the data in one set instead of numerous values sent in-line with the query requested.

### 5.12 Evaluation

To evaluate our query processing, we used merge techniques within the query engine as a router and sink aggregator. To measure the performance, we picked the aggregation and none aggregation queries using merge and none merge techniques. We compared the two to find the transformations between using such techniques. The evaluation considered the energy consumed and the elapsed time to perform the query.

#### 5.12.1 Aggregated Query Energy Consumption

In ODQS, we measured the energy to determine the consumed energy using Sensinode cc2430, bearing in mind the packet size in bytes’ unit. The greater the packet size, the higher
the energy consumed as clarified in Eq. (5-5) moreover, Eq. (5-6). The routing level is the crucial argument for our measurement.

It is obvious from Figure 5-24 that the aggregation query saved more energy than the none-aggregation query. The packet size for the aggregation is only 2000 bytes compared to the bytes consumed for the none-aggregation, which was recorded as being 4000 bytes. There were approximately 50% savings due to the use of aggregation techniques in querying the data from the sensor networks.

From the energy perspective, according to (Amiri, 2010) they have use SkyTmote, as it is applicable for ad-hoc network environment comparable to Sensinode device, the energy consumed by a sensor node while sending, receiving or even discarding a packet can be described using a linear equation Eq. (5-7) proposed in (Feeney and Nilsson, 2001).

\[
\text{Energy} = (M \cdot \text{size}) + E \quad (5-7)
\]

Where E is a fixed component related to device state changes and channel acquisition overhead for cc2430, and another incremental component which are associated to the size of the packet (M-size) is the packet size, M is the number of packets, and E. is constant 0.027 joules for sending or receiving 1 byte in cc2430 needs (Amiri, 2010). This means that as the packet size increases, the energy is highly consumed. The energy consumed in our experiment is plotted in Figure. 5-25, and it is compared to none aggregated queries. It is observed that the attained technique performed a huge savings of consumed energy per query as the number of
queries requested by the end user increase. As it can be seen in Figure. (5-25), It is measured to save on average 55% of sensor lifetime, it reaches a peak of 80% energy saving.

For measuring the energy consumed, we have user the total bytes broadcasted instead of the joules unit to measure the electric unit of sensor power. According to Amiri (2010), 0.027 joules for sending or receiving 1 byte in cc2430, this means total bytes should be multiplied by 0.027 joules. Consequently the total consumption in joules can be calculated by equation Eq. (5-8).

\[
\text{Energy} = 0.027 \cdot (M \cdot \text{size}) + E
\]  

(5-8)

![Figure 5-25](image)

**Figure 5-25**  Energy consumed versus number of queries

### 5.12.2 Merge Query Performance

The performance of our query processing was also tested. This was done using merge techniques within the query engine. Merging the query as one input and receiving the result as one output is the new technique we implemented in our query processing for the cc2430 Sensinode device. To calculate the performance of the developed ODQS system, many query-based end user requests are send to the sensor networks. The same query attributes were send to measure the performance of the sensors to aggregate or not as well as to merge or not and for performing both functions as well. For experimental testing, we execute the query using merge and none merge techniques and compare the two to find the percentage differences between them, just to have two different query examples.
In ODQS real-time sensing exposed higher impact on response time for any performed query. Figure 5-26 offers a comparison between two queries (Query ID: 2801 merge, Query ID: 6559 none merge) from ODQS termed as first query and second query respectively. We applied same query conditions, however, achieved merge technique to the first query. It is observed that the merged query took more time than none merged query when it comes to elapsed time to complete the query. None-merged query logged response time was 60% less than the response time for the merged query as showed in Figure 5-26(a). Furthermore, the first query consumed less energy than the second query. The trade-off associated between elapsed time and energy is recorded about 5% increases in this example as in Figure 5-26 (b). That means the query engine was waiting for the values to be merged before performing the execution, with despite the fact that fewer packets are transmitted to the user compared to none merged query.

![Figure 5-26](a) Query Elapsed time  ![Figure 5-26](b) Query Energy consumed

We observed that the query results for aggregated query compared to none aggregated query saved significant energy. With regards to merge methodology on the other hand, it is a built-in function in this experiment which merge all sets of queries to save more energy to sense all the necessary data in one go only. Furthermore, merge technique saved sensor nodes energy with some disregarded time-consuming.

### 5.13 Conclusion

In this chapter, we have addressed the problem of query-based data collection in WSNs using an on-demand query interface. We designed the interface in terms of sending queries
from the sink to be forwarded to the routers, then to the sensor nodes. Then we proposed an
efficient engine for a query-based data collection scheme as the main contribution. The query
engine chooses the optimal query plan to send the query packet and adapts the best routing
mechanism based on the communication-cost metrics.

Our experimental testing and evaluation of the data quality and energy efficiency showed the encouraging impact on the node’s lifetime and other resources in addition to its ease
of procedure structures. Finally, we examined sample query results, indicated energy saving of
in-network aggregated queries and merged queries. The judgement of execution time and
energy for aggregated queries wherein both discussed merged and none merged scenarios
showed their respective percentage of improvements. The next improvement to ODQS system
is to improve the query engine to choose energetically the optimal query plan to send the query
packets and adapt the best routing mechanism based on the communication cost metrics.

Our sensing system is very promising, given the capacities, it offers in the use of sensor
networks. In future work, we will focus on remaining research issues such as the support of
continuous and real-time queries, and the management of heterogeneity and interoperability.
For the next step in our research, we will work on confidence factor as a weight for the
evaluation, and also for high performance for our algorithm.

This chapter is concluded by the evaluation of proposed on-demand query sensing
system for room monitoring using testbed sensors for our purposes. It is very clear that our
algorithm is promising for sensor network database query and backup when it is necessary for
historical purposes, as well as for monitoring functions. This also leads us to further research
on a sampling–based on re-optimise the query using the on-demand interface designed for
sensing any monitoring environment.
Chapter 6 Probability based On-demand Query Processing for Wireless Sensor Databases

This chapter discusses the concept of query optimisation in in-network data processing and how to overcome this research statement to perform high speed for the query as well as save energy of the sensor node that acts a server (database). The use of in-network query processing techniques in handling the large amount of data they generate is also considered. The in-network query optimizer is presented as a core solution which is one of the query management methods.

6.1 Introduction

Wireless sensor networks are being implemented in diverse sorts of environment and essential for different applications. With the help of employment of various kinds of sensors, it becomes convenient to increase the functionality of sensor networks to support different applications. Most of the in-network query requests are prepared to be executed in an appropriate prototype structure. While in other situations, the applications are derived from Graphical User Interface (GUI) developments technology. Hence, based on this, it is noticed that creation of query language for sensors is a tough problem as presented in (Degirmenci, Kharoufeh and Prokopyev, 2014). As fundamental research for the sensor query based, a research statement, introduced in (Al-hoqani et al., 2017) illustrated aggregation and merge techniques for ODQS system. Additionally, an adaptive sensing system for sampling the data was also proposed in (Noura Al-Hoqani and Yang, 2015) that control the missing rate. Extension work will be provided in this chapter for dynamic and energy efficiency research.

According to (Alsbouli et al., 2011), the query-based approaches involve requesting response interactions between the end-user or application components and sensor nodes. End users issue queries in an appropriate language, and then each query is disseminated after evaluated using a statistical model to the networks to retrieve the desired data from the sensors based on the query demanded. With regards to out contribution for in-network tasks, it is built to execute two fundamental tasks: information acquisition/collection at the sink nodes, and dissemination of information to the nodes across the network only if the statistical model is not sufficient to answer the query.

This chapter addresses the problem of massive communication cost consumption at query processing stage. Our approach is a probability model, ODSQ-P, which answers the
user’s inputted queries, proposed, if it satisfies the accuracy condition. However, ODQS-D interface is designed for disseminating queries at different network levels in a tree-based network structure. Our experiment is from the measurement of room monitoring aiming at maximising the lifetime of the sensor energy for two query types aggregated, and none aggregated queries.

The existing sensor nodes have various limitations; particularly in the concern of power consumption. Moreover, they are very static in nature. Therefore, a query processing optimizer based on probability model statistics is proposed, if the end user requests none aggregated query. On the other hand, for the aggregated queries, a dynamic aggregator model based on Dynamic Programming (DP) method to control the aggregation mission in complex sensor networks.

Our approach is designed to generate the data from the source node only if it satisfies three basic conditions. First one is the historical lifetime model is valid; the second is that the probability model is sufficient to answer the query, and third is that the direction to dispatch the data is the shortest. These can be achieved using a dynamic programming technique as well as the probability model technique.

To the best of our knowledge, the algorithm proposed in this chapter is a novel in the wireless sensor field with dynamic context queries. Compared to other mechanisms such as data fusion and compression, our proposed algorithm is the simplest sensing algorithm applied in a real-time sensing system. This is to prove its performance in addition to the probability model construction on an on-demand base. Furthermore, it is more accurate and energy efficient.

6.2 Probability Model

Query framework has been used extensively in ambiguous dynamic systems with on-demand sensing systems approach, for which the user optimise the network resource. In this chapter, a great emphasis on the optimisation is drawn, of a cost function that combines energy consumption, elapsed time, and delay.

6.2.1 Query Probabilistic model

The steps involved in our technique are shown in the block diagram in Figure 6-1 for constructing a query answer for any query requested by users using our ODQS (Al-hoqani et
Our sensing algorithm is built at the PC level, where the historic JSON file can be reached.

The results are calculated in terms of energy consumed for not performing any aggregation in addition to the sensing of the packet received from the closer sensor nodes. Similarly, to ensure that the probability of the query waiting time factor for the query requests, it will be checked as soon as it reaches the sink, if it is needed or not, before it is processed. In our interface, the waiting time is fixed to be 0.35 seconds to get the result to the output interface.

**6.2.2 ODQS-P Probabilistic algorithm**

The pdf probability model with freshness deployed features is used, which present the simplicity and accuracy. The probabilistic model is mainly constructed and depends upon the characteristics of the environmental aspects. In this scenario of query processing, this work is focusing and without being deprived of loss of simplification and for exemplification
determinations, on a model is that comprised of the Gaussian distribution. It has also been realised that the pdf is used to represent the aspects of the query distribution functions from the perspective of the Gaussian distribution. By considering that a query arrives in the system at time $t$ and asks to estimate the values of temperature of the sensor database with an error $\varepsilon$ and $\theta$ is the confidence error $1 - 0 (\theta \in [0, 1])$ (Diallo et al., 2015). The random variable $X$ follows a normal distribution with parameters $\mu$ and $\sigma$ using the probability density equation as introduced in equation (6-1): $\mu$ and $\sigma$ represent the mean and the standard deviation respectively.

$$P(X) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ \hspace{1cm} (6-1)

The late time of the query from the last reading is given by $T = t - t_N$. Where $t$ is the current time and $t_N$ is the time for the last reading available in the historic file. After that the query estimation is computed using pdf equation (6-2), the expected query answer $E(X)$ is given by:

$$E[(X - \mu)^2] = \int_{\mu-\sigma}^{\mu+\sigma} p(x) dx$$ \hspace{1cm} (6-2)

The variance $\sigma^2$ of the temperature between the last reading and the arrival of the query should be $\sigma^2 \leq \varepsilon$; the variance is calculated using equation (6-3).

$$\Delta = \frac{1}{\sum_{i=1}^{N} \xi_i} X \sigma^2$$ \hspace{1cm} (6-3)

The graph could also be called as the bell curve or as a histograms with two parameters $\mu$ and $\sigma$ respectively would be able to represent the mean and standard deviation of $X$. The probability distribution is mainly symmetrical with respect to the aspects of $\mu$, which therefore symbolizes the main tendency, and deals with the aspects of dispersion of the distribution. To produce the expected model, the proposed method is encoded and produce algorithm 1 using MATLAB tools given the data collected from our experiment.

The outputs of Algorithm I present the expected query answer for the injected new query by the end user using the ODQS interface.
Algorithm I: ODQS-P Model Pseudo Code

```
Input x=reading from previous acquisition query (Historic data)
CT=CurrentTime
LT=Last time of the last query from the last reading
T=(CT-LT), m = 48 hours, Er = {0.05};
mu = mean(x)

sd = standard deviation(x)
y= pdf ('normal', x, mu, sd);
z=(x-mu)/sd;

fun=@(x) (1/(sqrt(2*pi).sd). x.exp(-((x-mu)/sd).^2/2));
if T < m time check if data is out of date
    var(x) = (1/sum(x(1,1).x(1,1))) * sigma^2
if var <= Er check for error confidence [0-1]
    query=integral (fun, p, q,'array valued', true); p, q are lower and upper bounds for the integral function
else
    Go to WSN to sense the data
end

Save historic data in JSON file
```

6.3 Query Optimization Methodologies

In the context of SQL statement queries over probabilistic model and distribution techniques, the proposed choice of the optimisation techniques depends on many factors. In addition to the probabilistic model, the choice also depends on the query clause.

In our data model, the readings are selected from a historical database using the previous values stored as the lowest cost per query in terms of bytes. The On-demand query sensing dynamic and probability based (ODQS-DP) query optimizer are demonstrated in figure 6-2. It includes the DP algorithm model at the base station level and the probability density function (pdf) at the interface level.
6.3.1 Base station Query answer Model

For the probability law, to get an answer to the query by constructing probability model, historic data is required. The main idea is to get the probability of collecting values for temperature, light and battery readings from the different sensor, which might be critical for the end user.

The framework of the proposed model in figure 6-2 illustrates the different stages of query processing for our ODQS-DP model. At the first stage, the user submits the SQL statement. After that, a soft parse required for the requested data is checked for syntax, historic data availability, and construct the query answer from a probability model if the data is out of date from the historical data. The third step is for hard parsing called the optimization stage, which generates multiple queries and query plans after cost evaluation. Another step required for row source generation, which generates the query aggregation level dynamically or
randomly according to the specified SQL statement. The last step will be the execution of the selected query from the sensor network. The principal of the dynamic query-based sensing concept is that the user’s specifications determine the sensing request for the required query statement by primary selections if it is aggregation query or not, the aggregated query to be performed dynamically or randomly among network levels. The two query model are to be executed dynamically depends on the user’s inputted queries, which have been developed using c programming language to execute the best in terms or less costly.

6.3.2 Query Sensing Model

The dynamic query model consists of four stages. The stages are the four aggregator levels in our network architecture. The query model iterates to the different network levels, which is four network routing levels in our experiment, and compares it with different level energy consumption in terms of communication cost. Our query model level consists of:

1. Interface pc aggregator.
2. Sink aggregator.
3. Router aggregator.
4. Router and sink aggregator.

In our study, the query engine generates a set of query plans on \( qp \), denoted as \( qq = \{ qp_1, qp_2, qp_m \ldots \} \); where \( m \) is the total number of query plans generated. A query plan \( qp \in qq \) denotes how to perform the query in the sensor network as a substitute of just stipulating how to sample sensors readings from a different node.

On the other hand, the Dynamic strategy is different from the static strategy. The dynamic strategy performs the best plan with less cost and high rewards to enhance the system of the static strategy of query processing. Furthermore, whenever the static strategy failed to execute efficiently, then the random strategy can be implied to provide the best query plan with values. This specific technique has two constituents, operators or query builder that mainly coded into the plan of execution and autonomously adapted to make the run-time changes. The other constituents are reactive optimizers that would be able to reform the plan and measure the different cost due to the fact of query attributes and runtime changes.
6.4 Experimental Test

Our results are based on running experiments over real-world data set that are collected during the past few months. The data are a trace of 49331 readings from 20 sensors in a water tank. In this scenario, MATLAB tools were used for developing the probability query model algorithm. Query builder model analyses attributes from different queries inputted by the users, and then the probability model is constructed using the model means and standard deviation values to estimate the query answer. Furthermore, the dynamic network level feature was provided to make sure of two things. First one is to make sure any SQL declarative query type can be encoded using ODQS. Secondly, the historical data can be constructed into a probability model used as optimization method which is invoked according to the query type injected, to generated query answer without the need to sense unimportant data.

Our approach is tested using a testbed Sensinode cc2430 platform device over a developed on-demand query-based interface, on a network architecture with a user input PC, one sink, two intermediate routers, four sensor nodes to sense the temperature, light, battery values from our classroom. The data collected is stored in a JSON file for historical use in the prediction model. The system runs for the specified period and interval according to the user inputs, each query carried a token number, the real-time monitoring shows the queries at a different time interval, with associated readings and values. The system carries out the cost, elapsed time, and packet losses comparison then generates the best query plan.

Figure 6-3 demonstrates the ODQS system interface. The system has a back-end for historical data storage purposes. The optimizer receives the query and executes the query id if it is available in the historical data return it to the user, else it will request to be generated from sensor database, by determining the sample, and have the statistics collected from the probabilistic model using the SQL statement requested. After that, once the answer is ready, a copy of it is kept in the optimizer, as well as reply to the user with the record demanded.
6.4.1 Probability Model Testing

The developed query sensing model collects all required data includes environmental sensing values, sensing period and interval, and the query operations. While the collected data are general monitoring data, the probability model used depends on the physical phenomenon studies. The values in historical data were tested for normality.

The normal distribution method for the collected temperature data is tested, it shows that most of the readings are in the specific temperature range. The energy consumption at this stage is measured based on the fact that the network is not activated, which means as the number of queries answered is increased, the saving is dramatically increasing. To conclude, this shows the importance of the model construction before sensing the sensor networks.

For the entire relations, the statistics have estimated for a small subset that is because of the more time consumption of the whole relation. The constructed probability model -according to our data are listed in Table 6-1, as it can be seen from the values the new query could be answered by the model where the model variance value was lower than the confidence error. In this experiment, the tolerance error $\varepsilon$ varies between 0 and 1 and the confidence error fixed to 95%.

<table>
<thead>
<tr>
<th>Table 6-1</th>
<th>ODQS-DP model parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$-error tolerance</td>
<td>Temperature</td>
</tr>
<tr>
<td>$\mu$-mean</td>
<td>16.93</td>
</tr>
<tr>
<td>$\sigma$-standard deviation</td>
<td>5.86</td>
</tr>
<tr>
<td>Q-query answer</td>
<td>17.0762</td>
</tr>
</tbody>
</table>
Figure 6-4 illustrates the energy saved for various parameter values resulted from probability model construction per query. It is observed that the best error was measured at 0.01 and confidence interval greater than 95%. The model saved approximately 99% of the sensor lifetime.

![Energy Consumed for different Error Tolerance](image)

Figure 6-4   Energy Consumed for different Error Tolerance

### 6.4.2 Dynamic Query Plans

In this section, the Dynamic query processing is encoded using visual studio C# to implement the best query plan decision that saves energy for our query optimizer. The short tree path will be selected if the network signal strength is equal, to overcome the collision that might occur instead.

For this chapter, two statements are executed in-network to request sensor’s network database. The first statement illustrates the dynamic characteristic, syntax presented in statement I. However; other statement shows the random representation as defined in statement II, to present the enhancements in our model.

**Statement I:**

```sql
SELECT MAX (Temp, Light, Battery)
FROM sensors
WHERE interval is 10s,
LIFETIME 1 day.
```

**Statement II:**

```sql
SELECT AVG (Att1, Att2, Att3)
FROM sensors (S1, S2, S3, S4)
WHERE SI is 10s,
LIFETIME SP 1 day,
```
The developed dynamic aggregation pseudo code is provided in algorithm II where the system generates the best model to perform the query and execute the requested statement to retrieve the values from the sensor node’s database. To sum up, we can say, the new energy expected to be consumed depends on last consumed energy and last operation made at the specified network level.

**Algorithm II: ODQS-D dynamic model Pseudo Code**

```plaintext
Input S1, S2, S3, S4, S5, S6, Att1, Att2, Att3, Sample Period SP, Sample Interval SI, (Query statement), Aggregation (Max, Min, Average), Dynamic (Yes, No) Threshold = query cost, packet loss, response time
Aggregation levels = i;
Where i = 0, 1, 2, 3;
Initial level = 0;
For Query size = SP;
If threshold[i] < threshold[i-1]
    If query cost[i] < query cost[i-1] & packet loss [i] < packet loss [i-1] & response time [i] < response time [i-1]
        While i <= 3;
            then
                Aggregation levels = [i+1]
            Else
                Aggregation levels = 0;
End if
End
```

6.4.3 Cost model

For calculating the cost for performing any number of queries, TTQM scheme (Xiang, Lim, K.-L. Tan, et al., 2007) is followed to compute the overall cost consumed by the sensor node. The main consumption produced from the transmission cost for each query packet is illustrated in equations (6-4) and (6-5).

\[
\text{Aggregation=} \sum (Q_b \cdot Q_h) - (Q_b \cdot Q_r) \quad (6-4)
\]

\[
\text{Non-Aggregation=} \sum (Q_b) \cdot \sum (Q_h) \quad (6-5)
\]

Where, \(Q_b\) the total bytes per query, \(Q_h\) is the total hops from each sink node to source node forward and backward., \(Q_r\) is the total aggregated queries.

Queries in wireless sensor network require a subset of source nodes to investigate and measure collected data. After running the experiment for specific interval and specified attributes, auspicious efficient energy savings is established. From the discussed
methodologies, an expected output panel is analysed in Figure 6-5. The Packet size in our experiment is 30 bytes. The equation to calculate the expected cost depends on overall hops performed in the network architecture to the interface.

![Figure 6-5](image)

**Figure 6- 5** Query dynamic models versus random model

In this section, it is observed that the idea is resulting perfectly on the performance as well as the quality of the data, to proof this; the resulted energy savings using dynamic programming process is plotted in figure 6-5 and discussed in detail in the following section.

### 6.5 Results and Discussion

#### 6.5.1 Optimized Query Performance

Proposing a query processing by incorporating statistical optimizer into a dynamic WSNs is essential to make the best use of sensor lifetime. Optimizer statistics are collections of data describe the database and the attributes specified. Our ODQS-DP optimizer uses statistics to choose the best execution plan for each SQL statement. Being able to gather the appropriate statistics promptly is critical to maintaining acceptable performance on any system (Oracle, 2013)

The optimizer works to generate the best query plan that avoids communication and energy cost for the sensor. It generates multiple query plans and executes the best after comparing them by cost using the statistics collected previously since the last query executed. The threshold is automatically set to ensure minimum cost to execute the query specified via the interface.
Figure 6-6 shows the trade-off between energy consumed by each query and time used to perform it. It is clearly noticed that to save more energy, occupy more time for query processing especially when the request is more complex in additional powerful feature, to the aggregation parameter as well. The percentage of saving is almost 66% of energy saving compared to the time of 1.5 seconds per query, which is equivalent to 44% of the time used by the higher energy saver.

6.5.2 Optimized Query Evaluation

The proposed optimiser works online also. The statistics gathering is prepared during execution phase and stored for next retrieval required. This optimizer will save time and communication cost, as well as energy consumption, as the access to the sensor will be decreased. Therefore, testing many different SQL query types is critical to be more precise on the performance and evaluation stage of the algorithm result.

The resulted data from our experiment is plotted in Figure 6-7; it demonstrates the dynamic versus the randomly change from one level to another to depict the query performing the aggregation action at the specified sensor network level. The energy in this experiment is presented in bytes.
Our approach combines query-based with real-time sensing system with statistical modelling techniques to perform query processing optimisation that uses the dynamic query selections and probabilistic confidence interval as query execution principles. For future improvement to this work, it is also recommended to apply an optimality rules, for the high quality sensed data, consider the time, energy, and delay. It is observed that ODQS-DP algorithm achieved a high rate of energy saving. Moreover, additional automation for dynamic aggregation and data sensing, to acquire quality data further energy efficiency.

6.6 Conclusion

Our approach combines query based with real-time sensing system with statistical modelling techniques to perform query processing optimisation that uses the dynamic query selections and probabilistic confidence interval as query execution principles. For future improvement to this work, it is also recommended to apply an optimality rules, for the high quality sensed data, consider the time, energy, and delay. It is observed that ODQS-DP algorithm achieved a high rate of energy saving. Moreover, additional automation for dynamic aggregation and data sensing to acquire quality data and further energy efficiency is required.

The communication cost in a wireless sensor network is the energy consumed for performing the communication. The total communication cost is the sum of the energy consumed by each node in the network. We propose a fully distributed algorithm, which does.
not require the gathering of network information. Queries instruct sensor nodes which are specified as a database, filtering and processing of the data acquired from the environment. Those data collected are usually highly redundant. Therefore, a well-designed query, which is not reporting the irrelevant, redundant, and highly correlated sensory data to the database node can further increase the impact of data reduction.
Chapter 7 Graph-based Complex Query Processing and Genetic Algorithm for Wireless Sensor Networks Database

This chapter discusses the concept of in-network data processing for graph-based network architecture. This chapter presents the proposal of Dijkstra algorithm for selecting the best aggregation node for multiple sensor queries using the ODQS system. Genetic algorithm is used to evaluate the best plan among set of queries. The resulted output proofs that we have come up with the targeted query optimisation

7.1 Introduction

In-network adaptive query optimisation for wireless sensor network (WSN) databases (DBs) is a promising solution to extend the life of the nodes. In fact, appropriate query execution plans for sensing data transmission could reduce transmission time, communication, and processing requirements, and hence save energy. Therefore, it is necessary to generate the best query execution plan.

Over the last decade, various WSN query processors that provide data aggregation, fusion, merging, compression, prediction, and filtering have been proposed. These processors consider WSNs as distributed DBs to apply declarative query processing and simplify the implementation of WSN applications (Jabeen and Nawaz, 2015). Al-Hoqani et al. (Al-hoqani et al., 2017) proposed an on-demand query sensing framework that consists of a query-based engine transferring the sensor measurement data into aggregate functions at both the routing and sink nodes. This framework optimises the query by sending data when requested and eliminating irrelevant data through aggregation at intermediate nodes. Consequently, it can extend the battery life of WSN nodes.

In this study, we address on-demand complex queries at different WSN levels and a graph-based network structure, such as mesh networks. In-addition, we consider a room-monitoring real-time WSN application and aim to maximise the sensor battery life. Specifically, we attempt to overcome the current limitations of the aggregator nodes, including high power consumption and standard static allocation, which includes all module computation required to send the query to the best path. To control the aggregation process in heterogeneous WSNs, we propose a graph-based query processing method based on real-time sensing and a multiple-query model. To the best of our knowledge, this is a novel approach in WSNs with query-based
topologies, and we extend the work in (Al-Hoqani, Yang and Fiadzeawu, 2017a) by solving more complex query scenarios. Furthermore, we use data sharing for multiple queries in multiple base stations and a genetic algorithm to analyse the performance of the proposed approach.

7.2 WSN

7.2.1 WSN and query model

The proposed WSN model consists of query flow and query response. Likewise, the WSN contains sensor and sink nodes, where the former can also act as routers. This way, any sensor node can be involved in the execution of a query. Figure (7-1) illustrates the graph-based query-flow model used to retrieve data from different wireless sensor network’s (WSN) databases (DBs). The model generates a query plan after analysing the WSN topology in order to select the best query path obtained from the routing and neighbour tables.

![WSN Diagram](image)

**Figure 7-1** Multiple on-demand queries for in-network wireless sensor networks (WSN)

The on-demand query sensing framework has been extensively applied in ambiguous dynamic systems to optimise the use of network resources (Al-Hoqani, Yang and Fiadzeawu, 2017). In this work, we focus on the optimisation of user queries for accurate sensor readings and maximisation of sensor battery life by reducing the energy consumption per query. In our WSN model, the readings are selected from a history DB using the previous values with the lowest number of bytes per query. The proposed query optimiser illustrated in Figure (7-2), employs the involvement of three databases for different query injected in the sensing system.

Let us consider three query statements, Q1, Q2, and Q3, from four different sensors, S1 to S4, at different network levels, where $a$, $b$, $c$, and $d$ are temperature readings from sensors S1 to S4, simultaneously sensed from all the sensors all at once. They are to integrate different
operation from simpler to the most complex, which has more than three operations in one query as in it clearly seen in Q3 Eq. (3). The different operation are mentioned in the below queries such as merge, min, and max operations.

Statement Q1 = query (Max (a + b)) 

Statement Q2 = query (Merge (min (a + b) + c)) 

Statement Q3 = query (Min (max (a + b) + merge (c + d))) 

The selection of the best query execution plan in a network is known as query optimisation. In our model, we consider the cost consumed per each query executed for query optimisation. For every query Qy requested by the end user, the sink node evaluates its contents Qs. The collection of all query data in the network at time Ty is defined by hops parameter Dh when queries are executed from history DB, Dc when they are executed from sensor cache memory, and Dw when they are executed from WSN DB. In addition, we define cost function fco as the cost of transporting data through multiple hops until they are retrieved at the user interface. This function depends on the transmission time and number of bytes per hop, which is considered as constant, according to the whole query packet size. For instance, we can set the cost considering 10 bytes per hop different than the specified query cost in previous chapters, which was 30 bytes, disregarding changes occurring in the user queries because the query is too specific and also consuming some energy for computation. Then, from all the possible query
execution plans, we select the one that minimises the cost function and improves energy efficiency.

For energy efficiency, the information from energy consumption and data retrieval obtained from the last query at each network level is considered. Precisely, we determine the energy consumption without considering any aggregation process and the packet transmission among sensor nodes. The energy cost is calculated using the energy consumption per query byte, similar to the above discussion. Moreover, we ensure data transmission by using the probability model proposed in (Al-Hoqani, Yang and Fiadzeawu, 2017) to update the DB. Then, the best query execution plan is obtained in real time from the developed on-demand query engine, incorporating graph-based solutions or a genetic algorithm, depending on the type of query.

In the context of SQL statement execution using probabilistic models and distribution techniques, the selection of the optimisation method depends on several parameters from the probabilistic model and the type of query. According to Oracle (Oracle, 2013), there are three main stages in query processing with distribution techniques, namely, the initial execution, execution, and subsequent execution stages as shown in Figure (7-3). At each stage, different operations are performed depending on a threshold set for the query cost according to historical records. Each query passes through three stages, constituting the adaptive model for on-demand in-network queries illustrated in Figure (7-3).

The sensor DB searches an answer in its cache according to the query, compares it with the history DB output, and the WSN DB decides the action corresponding to the query. The proposed optimiser employs a genetic algorithm at the sink node level (L0) and a probability density function at the interface level as in the work (Al-Hoqani, Yang and Fiadzeawu, 2017b). Hence, as shown in Figure (7-3), the stages of the proposed adaptive query model follow the process:

1. On-demand query
2. Sensor aggregation
3. Sensor non-aggregation
To process multiple queries and retrieve data, we adopt a dynamic strategy that updates the best query execution plan with the lowest cost and highest energy saving. Figure (7-4) shows a multi-query model, where a query is first introduced in the query engine for analysis. Second, the query is broadcast to the network sensors. Third, the query is evaluated, and if it is a non-aggregation query, a genetic algorithm plan is executed; otherwise, a graph-based plan is executed. Finally, the query is checked for syntax, to match with an existing query in the cache memory, or to be executed.

Hence, the dynamic strategy is one of the strategies assure that the most efficient query execution plan is executed in the WSN. The dynamic strategy consists of both a query builder, which is coded into the query execution plan and performs direct runtime updates and reactive optimization, in which the best plan is selected and measure the effects of runtime changes. The proposed model illustrated in Figure (7-4) shows the different stages of query processing, which are also related to the query optimiser. First, an SQL statement is submitted to the WSN. Next, the requested data is soft-parsed for syntax, data availability, and query generation. Then, hard-
parsing corresponding to the optimisation stage generates multiple queries and plans after cost evaluation. Another process determines the row source generation, which retrieves the query states either dynamically or randomly according to the SQL statement. Finally, the best execution plan is implemented by the WSN.

### 7.2.2 Network Initialization

The query model relies on the forward routing table and the neighbour routing table of each sensor that are determined during the WSN initialisation. We consider following series of initialisation steps, illustrated in Figure (7-5). First, the sink node is assigned to level 0 (L0). Then, it broadcasts a route request packet (RREQ) that updates the level of the sensor nodes that immediately receive the packet. Hence, nodes A and B shown in Figure 7-5 are updated to level 1 after receiving the RREQ. Likewise, the RREQ is forwarded to all the sensor nodes, the level will be updated until the last level is reached, which we assume to be L6 in this experiment, they then send a propagation of route reply packet (RREP) back to the node from which they received the RREQ. For instance, nodes A and B send an RREP back to the sink node as shown in Figure (7-5).

![Graph-based network queries](image)

The different network level is specified when the sink node broadcast to a route request packet, then it is a new network level so it will increment the current existing network level b
one point. So the level will be recorded as L1, L2, and L3 till it reach the end of the network levels, which it start sending propagation of rout reply packets.

The sensor nodes are to synchronize and coordinate among each other by recording their own address and neighbor address with the total hops to sink for the purpose of any further broadcasting transmission responsibilities.

7.2.3 Query execution plan optimization

Queries in a WSN require a subset of source nodes to measure and retrieve data. We evaluate the energy saving after executing queries at specific intervals and attributes. Figure (7-6) illustrates a typical case study for query processing.

![On-demand tree-based sensing system](image)

Figure 7-6 On-demand tree-based sensing system

In this case study, we consider a packet size of 30 bytes. In addition, the expected cost depends on the packet hops from the corresponding sensor nodes to the user interface. The adjacent list and algorithm cost flow are contracted from graph query according to Figure (7-6). From the adjacent list shown in Table (7-1), the query path will be selected according to the Breath-First-Search (BFS) theory approach to select level wise instead of branch wise. This algorithm generates a spanning tree called the breadth-first search tree. The nodes in a branch indicate the shortest path from the root or the ‘distance’ from the root to a destination node.


<table>
<thead>
<tr>
<th>Node</th>
<th>Binary</th>
<th>Adjacent List</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0001</td>
<td>D, C</td>
</tr>
<tr>
<td>B</td>
<td>0010</td>
<td>C, E</td>
</tr>
<tr>
<td>C</td>
<td>0011</td>
<td>A, B, E, D, F</td>
</tr>
<tr>
<td>D</td>
<td>0100</td>
<td>A, C, F, G</td>
</tr>
<tr>
<td>E</td>
<td>0101</td>
<td>B, C, F, H</td>
</tr>
<tr>
<td>F</td>
<td>0110</td>
<td>C, D, E, G, H, I</td>
</tr>
<tr>
<td>G</td>
<td>0111</td>
<td>D, F, I, Sink</td>
</tr>
<tr>
<td>H</td>
<td>1000</td>
<td>E, F, I, Sink</td>
</tr>
<tr>
<td>I</td>
<td>1001</td>
<td>F, G, H, Sink</td>
</tr>
</tbody>
</table>

**Table 7-1** Adjacent list for a graph query network architecture

7.3 Graph Theory Optimization

7.3.1 Tree-based search using BFS (Breadth-First Search) algorithm

First, we use a breadth-first search algorithm to traverse the different node levels and initialise the WSN by setting the routing and broadcasting information. Specifically, the forward routing table is created at each node, which is part of a tree structure, and the neighbour routing table is created whenever required to determine child nodes. The visited nodes register with their level and neighbour nodes, or for a parent node, the number of hops and destination are registered. The breadth-first search algorithm follows pseudocode 1:

**Pseudocode 1: BFS for searching the best queue in a tree**

**Input:** Graph G=(V, E), directed, vertex s ∈ V  
**Output:** for all vertices u reachable from s, dist(u) is set to the distance from s to u

```plaintext
{Queue-of-node Q:  
Nodes u, v  
Q = {s} // Q is a queue  
Initialise(Q) // initialise Q to be empty  
v = root of T; // T tree of nodes  
visit v;  
enqueue (Q,v);  
While Q is not empty {  
dequeue(Q,v);  
for all edges (u,v) ∈ E {  
visit u;  
enqueue (Q,u)  
If dist(v) > dist(u) + l(u,v) {  
dist(v)=dist(u)+l(u,v)  
end if: }  
}}
```
In the case of a query request to node A, the path with the cost values, fitness values, and ratio for each query, if we assume the query request to node A, is calculated. The path needs to be constructed as explained in Table (7-2). It can be noticed that from source A to the destination sink node, there are six paths, and clearly one of which is less costly, and hence that will be selected. The chromosome in our case study represents the query, where each query carries four of attributes, {light, temperature, battery, and aggregate}. The graph in our example is $9 \times 9$, and to solve the matrix, we formulate the problem in GA, and the fitness cost and the reproduction process for a new generation are formulated. The same steps are followed by GA to mutate the massive and complicated network for as many requests according to user needs. Table (7-2) illustrates the different query plans and their cost for the user query from A node.

<table>
<thead>
<tr>
<th>S</th>
<th>Query route</th>
<th>Cost</th>
<th>Fitness</th>
<th>Fitness rate</th>
<th>Chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{A, D, G, I, sink}</td>
<td>120</td>
<td>0.008333</td>
<td>9%</td>
<td>0001-0101-0101-0111</td>
</tr>
<tr>
<td>2</td>
<td>{A, D, F, I, sink}</td>
<td>150</td>
<td>0.006667</td>
<td>8%</td>
<td>0101-1011-0111-0101</td>
</tr>
<tr>
<td>3</td>
<td>{A, D, G, sink}</td>
<td>120</td>
<td>0.008333</td>
<td>9%</td>
<td>0101-0111-0101-0111</td>
</tr>
<tr>
<td>4</td>
<td>{A, D, F, I, sink}</td>
<td>110</td>
<td>0.009091</td>
<td>10%</td>
<td>1001-1011-1101-0111</td>
</tr>
<tr>
<td>5</td>
<td>{A, D, F, H, sink}</td>
<td>130</td>
<td>0.007692</td>
<td>9%</td>
<td>0101-1011-0101-0111</td>
</tr>
<tr>
<td>6</td>
<td>{A, D, F, H, I, sink}</td>
<td>180</td>
<td>0.005556</td>
<td>6%</td>
<td>0011-0011-1101-0111</td>
</tr>
</tbody>
</table>

For query optimisation, the speed of the query depends on the data transfer and the order of joining multiple queries. Since measuring the query elapsed time is very important, the GA is adopted because of its dynamic nature, and compared with the Dijkstra, a graph theory method. The dynamic feature of the GA will be discussed in Section 7-4.

In summary, the BFS is used to generate the different tree paths as a response to the query asked, which worked very significantly and clearly answered the query asked, although it is not one of the best graph methodology

### 7.3.2 Graph-based search using Dijkstra algorithm

We assume that the number of bytes of a packet is denoted as PB (i.e. Packet Bytes) and the hop count is denoted as HC (i.e. Hop Count). Therefore, the total number of transmission
bytes denoted as TB (i.e. Total Bytes) can be calculated by using the following formula: (Yan & Al-Hoqani, N, 2018)

\[ TB = PB \times HC \]  

(7-1)

The total energy consumption \( E \), which is the proportional function of TB, is given as follows:

\[ E = (TB) \]  

(7-2)

For query MERGE (L, P, Q), we can use the Dijkstra algorithm to obtain the shortest path from the sensor node to the aggregation node, which is B, and then add the shortest path from the aggregation node to the sink node. Therefore, we should find the best aggregation node by choosing the shortest path from the sensors to the sink node, which could be calculated by the Dijkstra algorithm, and by choosing the first common ancestor node as the aggregation node for all the participated nodes. QP is the query plan according to the Dijkstra algorithm, if

\[ QP(i, j) = \{V_i \ldots V_k \ldots V_s \ldots V_j\} \]  

(7-3)

if the shortest path from vertex ‘i’ to vertex ‘j’, ‘k’ and ‘s’ are the middle vertices in this path, then QP (k, s) must be the shortest path from k to ‘s’. In Figure (7-5), the shortest path from node ‘L’ to ‘SN’ is

\[ QP(V_L, V_G, V_C, V_B, V_{SN}) \]  

(7-4)

while one of the shortest paths from ‘Q’ to ‘SN’ is

\[ QP(V_Q, V_P, V_M, V_B, V_{SN}) \]  

(7-5)

Moreover, the shortest path from ‘P’ to ‘SN’ is

\[ QP(V_P, V_M, V_D, V_B, V_{SN}) \]  

(7-6)

Consequently, ‘B’ is the best aggregation node, it is much better than ‘SN’. Moreover, ‘P’ is another aggregation node for the packet from ‘Q’ and packet ‘P’. There are four hops from ‘P’ to ‘B’ and five hops from ‘Q’ to ‘B’ if they send the packet to ‘B’ separately; the cost, based on equation (7-2) and equation (7-3), would be:

\[ E = (TB_P + TB_Q) = (TB_P \times 4 + TB_Q \times 5) \]  

(7-7)

where \( TB_P = TB_Q \), the total energy consumption is expressed as
\[
E = (PB \times 9)
\]

If the packets from both ‘P’ and ‘Q’ are aggregated at node ‘P’, the cost would be the following:

\[
E' = (PB_P \times 1 + PB_{merge}(P+Q) \times 4) = (PB_Q \times 1 + PB_{merge}(P+Q) \times 4)
\]

Figure (7-7) shows the structure of the aggregation packet (P+Q). The length of the aggregation packet is much shorter than the length of the sum of two separate packets because there is only one public ‘Head’, which means

\[
PB_{merge} (P+Q) < PB_P + PB_Q
\]  

when \( PB_P = PB_Q = PB \). Therefore,

\[
PB_{merge} (P+Q) < PB \times 2
\]

Then,

\[
PB_{merge} (P+Q) \times 4 < PB \times 8.
\]

Accordingly,

\[
E' = (PB \times 1 + PB_{merge}(P+Q) \times 4) < (PB \times 1 + PB \times 8).
\]

Therefore, ‘P’ is the best aggregation node for packets from both ‘P’ and ‘Q’.

Figure (7-7) illustrates the process to execute query \( \text{max}(L, P, Q) \) and shows the best execution plan. The primary process can be described as follows:

- The sink node analyses the query and determines that the three sensor nodes in the query share the first hop (i.e., node B).
- Once the query packet arrives at node B, it analyses the query and divides it into two sub-queries, one going through node C (i.e., read from node L), and the other through node D (i.e., \( \text{max}[P, Q] \)). Therefore, node B is an aggregator for the query function (i.e.,
it aggregates data from nodes L, P, and Q), and hence it must wait for the results from nodes C and D.

Figure 7-7  On-demand graph-based execution plans

- The query analysis and node selection are repeated until reaching target nodes L, P, and Q.
- When node P receives its subquery, it is analysed, and node P generates another subquery to node Q (i.e., read from node Q). Therefore, node P is also an aggregator (i.e., it determines \( \text{max} \{P, Q\} \)) that must wait for the result from node Q.

### 7.3.3 Query elapsed time

Query elapsed time \( T_y \) is computed as soon as the query engine receives the query answer, and the ODQS system generates a timestamp for the query start time, and the query finished time (Habib and Marimuthu, 2012). The difference between both times is the elapsed time. Equation (7-14) is used to make sure the complex query transmission time requirement is satisfied, where \( T_{y(i)} \) is the current query time, and \( T_{y(i-1)} \) is the previous query.

\[
T_y = T_{y(i)} - T_{y(i-1)} \tag{7-14}
\]

In the proposed query processing method, all sensor nodes route to their neighbors and the sink node contains the best paths to each sensor node, thus providing an efficient solution to queries generated by users. Furthermore, the adaptive query model follows the user specifications in the query statement by the initial selection of sensing requirements. By default, the model produces a probabilistic model as in (Al-hoqani et al., 2017).
Furthermore, complex queries are divided into subqueries according to the source of the query data, which can be the history DB, cache DB, or WSN DB. Hence, queries can be divided into three types of subqueries, which after retrieving the corresponding data from their DBs, are merged as a single response. To sum up, the core of the graph-based Dijkstra algorithm is to find the best aggregation node for every multi-sensor query. The fewer the hops are, the shorter the path is.

7.4 Experimental setup

7.4.1 WSN structure and query processing

For this experiment, we encoded the graph-based algorithm in Visual Studio C# to obtain the best query execution plan (i.e., that which saves the most energy and time). Moreover, we used the ad-hoc on-demand distance vector (AODV) protocol for the on-demand query tree structure according to the signal strength of the sensor nodes. This way, the node with the highest signal value at each level forwards the packets to the next level. If some nodes share the same signal strength at a level, the shortest path is selected for packet forwarding, which also allows overcoming possible packet collision or missing sensor readings.

We verified the proposed query processing approach using a test-bed Sensinode cc2430 platform over a developed on-demand query-based interface on a graph-based WSN architecture. The network consisted of a user input PC, one sink node, several sensor nodes to measure temperature and light from a room, and the sensor battery level. The data were collected on a storage JSON file for statistical use in the probabilistic model.

The optimizer receives the query, executes its ID, and retrieves the data requested by the user, provided that the data are available in the history DB. Otherwise, the optimiser requests data from the sensor DB and collects the statistics from the probabilistic model using the on-demand sensing system. Then, the optimizer maintains a copy of the answer and presents it to the user interface.

7.4.2 Multiple query sharing

We executed the three in-network statements to request sensor nodes readings, as requested by end-user. After analysing the query statements, the dynamic query processor retrieved the corresponding data according to its location. However, the queries show a random representation as defined in query statements Q1, Q2, and Q3, providing enhancements to the
proposed model. These enhancements are perfect for complex queries, as our model manages
to control the tradeoff between the different stages involved. The response is retrieved either
from the model stages after evaluating its cache location, or as updated data from the sensor
network DB.

To discuss multiple queries, we consider an example of the three query statements as
stated in section 7.2.1: statements Q1, Q2, and Q3 as the resulted output is clearly illustrated in
Table 7-3. If sensor $a$ is cached and sensor $b$ is from the history DB, then the energy saving
would be 100%. On the other hand, in Q3, if sensor $c$ is also available in the cache memory of
the sensor, then only the $d$ sensor will be required to sense resulting in 25% energy saving,
according to our easies energy saving multiple query sharing data model. Multiple queries are
first analysed and categories, they are collected from existing databases; if data do not exist,
then they are requested from the sensor’s network database.

For this experiment, we used the graph-based network structure; multiple users use our
ODQS (Al-hoqani et al., 2017) interface which is adapted to multiple pc modes. Sharing
resources is applied in the sink after it replies with existing historic result, also other cache-
memory sensor node data are shared. Moreover, the broadcast will not be sent to the nodes until
the sink confirms the query path for each query. Additionally, multiple queries are merged
together into one query to be forwarded to the nodes query attributes requested. Then they are
analysed, and the out-of-date sensors required will be merged into a single query for execution.

Table 7-3 Sharing data for Complex query approach

<table>
<thead>
<tr>
<th>Query input</th>
<th>Historic DB</th>
<th>Cache DB</th>
<th>WSN DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select sensor a, b, c</td>
<td>-</td>
<td>-</td>
<td>a, b, c</td>
</tr>
<tr>
<td>Select sensor a, b, c, d, e</td>
<td>-</td>
<td>a, b, c</td>
<td>d, e</td>
</tr>
<tr>
<td>Select sensor a, b, c, d, e, f, g</td>
<td>a, b, c</td>
<td>d, e</td>
<td>f, g</td>
</tr>
<tr>
<td>Select sensor a, b, c, d, e, f, g, h</td>
<td>a, b, c, d, e</td>
<td>f, g</td>
<td>h</td>
</tr>
<tr>
<td>Select sensor a, b, c, d, e, f, g, h, i</td>
<td>a, b, c, d, e, f, g</td>
<td>h</td>
<td>i</td>
</tr>
<tr>
<td>Select sensor a, b, c, d, e, f, g, h, i, j, k</td>
<td>a, b, c, d, e, f, g, h</td>
<td>i</td>
<td>j, k</td>
</tr>
</tbody>
</table>

According to our analysis, data transmission is reduced after aggregation; the graph-
based aggregation algorithm generates the optimal aggregation node. For multiple queries,
using the existing cache database in the sensors or using the historic warehouse database to select the best execution plan saves more energy.

Table 7-4 presents the comparison of the output resulting from our algorithm with normal query processing and another aggregation algorithm. The queries in the experiment are used from the network structure shown in Figure (7-7), for multiple queries, we combined Q1 and Q2 from Table 7-4 into one complex query, and after calculating the query hops, we compared the results.

Table 7-4 Performances of different query algorithms

<table>
<thead>
<tr>
<th>Query no</th>
<th>Query (MAX)</th>
<th>Normal Query</th>
<th>Aggregate query</th>
<th>Multiple query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>hops</td>
<td>Bytes</td>
<td>hops</td>
</tr>
<tr>
<td>Q1</td>
<td>A, B</td>
<td>2</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>Q2</td>
<td>C, H</td>
<td>5</td>
<td>150</td>
<td>4</td>
</tr>
<tr>
<td>Q3</td>
<td>L, N</td>
<td>8</td>
<td>240</td>
<td>7</td>
</tr>
<tr>
<td>Q4</td>
<td>L, O</td>
<td>9</td>
<td>270</td>
<td>6</td>
</tr>
<tr>
<td>Q5</td>
<td>N, R</td>
<td>9</td>
<td>270</td>
<td>6</td>
</tr>
<tr>
<td>Q6</td>
<td>M, Q</td>
<td>10</td>
<td>300</td>
<td>6</td>
</tr>
<tr>
<td>Q7</td>
<td>P, Q</td>
<td>11</td>
<td>330</td>
<td>6</td>
</tr>
</tbody>
</table>

In Figure (7-8), the resulting energy consumption is clearly reduced to half as can be seen, from 390 bytes to 270 bytes (Q. No. 3 & 4), and from 500 bytes compared to 240 bytes for queries number 5, 6, and 7.

![Figure 7-8 Energy consumed for different algorithms](image_url)
By contrast, it is observed that the proposed solution for complex queries performed better than the other aggregation-only and normal-sensing methods. To achieve high energy efficiency, the genetic algorithm, which is considered as one of the best optimisation algorithms for search heuristics in any database, is adopted. The result shows an improvement in the elapsed time and cost when performing each query.

7.4.3 Cost-based model

We adopt the energy model from sky mote board (Alonso, 2012), this was chosen due to the similarities between the Tmote Sky platform module and Sensinode cc2430, in the fact that both devices, that they are ad-hoc network nature. According to Alonso, the energy consumption of a sensor node for a single tuple is given by the following equations.

$$E_c = \sum_{sk=1}^{\infty} = t_{st} P_{st} + t_{rm} P_{rr} + t_{sm} P_{rt} + t_{cpu} P_{cpu} + t_{i} P_{i} \quad (7-15)$$

Table 7-5  Measured energy model parameters for the Tmote Sky platform

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature sensor</td>
<td>$P_t$</td>
<td>4.32</td>
<td>mW</td>
</tr>
<tr>
<td></td>
<td>$t_t$</td>
<td>220.4</td>
<td>ms</td>
</tr>
<tr>
<td>Radio transmitter</td>
<td>$P_{rt}$</td>
<td>61.1</td>
<td>mW</td>
</tr>
<tr>
<td>Sending message (30 bytes)</td>
<td>$t_{sm}$</td>
<td>9.63</td>
<td>ms</td>
</tr>
<tr>
<td>Radio receiver</td>
<td>$P_{rr}$</td>
<td>63.4</td>
<td>mW</td>
</tr>
<tr>
<td>Receiving message (min)</td>
<td>$t_{rm}$</td>
<td>2.02</td>
<td>ms</td>
</tr>
<tr>
<td>CPU active</td>
<td>$P_{cpu}$</td>
<td>8.76</td>
<td>mW</td>
</tr>
<tr>
<td>Time used for sampling a sensor</td>
<td>$t_{st}$</td>
<td>1.20</td>
<td>Ms</td>
</tr>
<tr>
<td>Power consumed by sensor</td>
<td>$P_{st}$</td>
<td>0.027</td>
<td>joules</td>
</tr>
<tr>
<td>Idle power</td>
<td>$P_i$</td>
<td>2.64</td>
<td>mW</td>
</tr>
</tbody>
</table>
7.4.4 Genetic algorithm presentation

Genetic algorithms (Jain, Chande and Tiwari, 2014) perform directed searches based on the mechanics of biological evolution. In this section, we use this approach to gain insights into the adaptive processes of sensing systems. The GA is used for selected the best query plan when no aggregation is required so depends normal query function, whereas BFS method is used to select the best aggregation node which results on the shortest query path. The GA is used to select the best in term of less costly query.

A chromosome in a genetic algorithm can be represented as a string containing any type of data, (e.g., binary, integer, or real number data). We consider the chromosome as a query plan for execution. Hence, the complex query-sensing problem can be formalised as a search problem in a genetic algorithm, where the query optimisation aims to a subset of the inner query to minimise the execution cost. We execute the genetic algorithm at the base station (sink node) level. A chromosome can be represented as below, where $K$ and $L$ represent the rows and columns between the query relations or nodes. The relations between different sites are presented by $[ (i_1,j_1), (i_2,j_2), \ldots, (i_N,j_N) ]$, where $(i_a,j_a)$ is the coordinates of the $a$th query plans and $P_{i_aj_a} = 1 (a = 1,2,3 \ldots N)$

\[ i_a \in \{x_1, x_2, \ldots, x_K \} \]
\[ j_a \in \{y_1, y_2, \ldots, y_L \} \]

Chromosome equation can be written as

\[ C (i_a, j_a) \quad a = 1,2 \ldots N \]
\[ C_{KL} = (X_K, Y_L) = (x_1y_1)(x_2y_2)(x_Ky_L) = c_{11}, c_{22}, c_{KL} \]

As shown in Figure (7-4), query optimisation involves three steps, namely, query tree generation, plan generation, and code generation. The query tree is generated based on graph theory, whereas the plan and code generation rely on the genetic algorithm, which transforms the different aggregation queries into subqueries after categorising them into three groups. After that, each group searches for the query answer to save more energy. The fitness function $f_{co}$ is used to minimise the query cost for the selected query plan and ensure the normality of the distributed data, thus maximising the sensor battery life.
7.5 Query Database evaluation model

In research by Al-hoqani et al., (2017), the execution cost per query was considered and evaluated in bytes. In this experiment, the execution plan for the sensor nodes is calculated. Similarly, to compute the aggregation query energy consumed, equation (7-16) is used as a fitness function \( f_{ce} \) to measure the query cost in bytes. The sensor data is retrieved only after analysing two additional databases, each database helping the sensor to save half of its energy to extend its lifetime. As a result, if any query matches the data from another source, then is 100% - 75%, representing 25% of the overall energy.

\[
Q(E) = \sum_{i=0}^{n} C \left( n Q_w + n Q_h + n Q_c \right) \tag{7-16}
\]

where \( Q_h \) describes the cost of obtaining the data from a historic database and is the CPU unit cost, \( Q_c \) is a fixed cost of sending a message out of the cache memory of sensor device, \( Q_w \) is the cost of sensing a single attribute from wireless sensor device and depends on the number of hops, and \( n \) is the number of tuples (nodes) retrieved by each database.

In this experiment, the overall query cost, which is the first objective function, involves users’ inputted queries if it is an aggregation query, then the cost is computed using equations 7-17 and 7-19 as explained in section 7.4.1. The second objective function is the sensor lifetime, which is computed using equation 7-15, which covers all sensor states costs. Consequently, the tested multi-objective functions are expected to provide the trade-off between query cost and sensor lifetime.

7.5.1 Execution cost fitness function

For the execution cost of electric power used by the sensor nodes to execute a plan, in our sensing system, we estimate the value for \( C_T \) as the number of \( h \) hops the node is away from the base station to transmit a message, which is obtained from the network initialisation stage.

For non-aggregation queries

\[
E(Q) = C_T \frac{s}{t_p} \left[ C_m + C_a \ n \right] \tag{7-17}
\]

\[
C_T = h \ * N \tag{7-18}
\]

For aggregation queries

\[
E(Q) = (N - 1) \frac{1}{t_p} \left[ C_m + C_a \ n \right] \tag{7-19}
\]
In our experiment, we fixed the message size to 30 bytes, therefore the cost of sending any number of attributes will be the same.

\[ C(T) = C_m \left( \frac{n}{7} \right) + C_a \ n \quad (7-20) \]

\[ C_m \sim 12 \ C_a \quad (7-21) \]

The fraction comes from the maximum number of attributes we can fit into a message.

The parameter describes the topology and will work with tree-based or graph-based network structure. The following data parameters are used for GA test optimization for aggregation queries and none aggregation queries cost function:

- \( C_T \) describes the topology,
- \( C_m \) is a fixed cost for sending a message, 359
- \( C_a \) is the cost for sensing a single attribute, 28.7
- \( t_p \) is the sampling interval (period interval- 30- sec)
- \( s \) is the average selectivity of a predicate (period long- one day)
- \( n \) is the number of tuples or (number of selected nodes)
- \( N \) is the total nodes in the networks.

### 7.5.2 Genetic algorithm

The genetic algorithm is a probabilistic method, which is capable of predicting the result of any inputted query into the sensor tree or graph network structure. The fundamental stages while outlining a GA are:

- Chromosome representation for the feasible solutions to the optimisation problem.
- The initial population of the feasible solutions and a fitness function that evaluates each solution.
- Genetic operators that generate a new population from the existing population.
- Control parameters such as population size, the probability of genetic operators, and a number of generations.

In a graph-based network, for queries that contains \( n \) nodes, the \((i)\) routers are the intermediate in any query requested. According to (Ban et al., 2015) the fitness function is the
inverse of the query cost as in equation (7-16), and the fitness function can be represented by equation (7-22).

\[ f_{co} = \frac{1}{\sum_{i=1}^{n-1} \text{cost} \ E(Q_i)} \]  

(7-22)

7.6 Results and discussion

We propose query processing by incorporating graph theory optimisers into dynamic WSNs. These optimizers rely on collected statistics from the DBs and WSN. Another type of optimiser is the genetic algorithm, which is similar to the Markov decision process because it uses an initial analysis of the WSN structure to select the best execution plan for each query. Hence, the timely collection of the appropriate statistics is critical for maintaining high accuracy in our query processing system. The multi-objective function was performed in MATLAB to input into a genetic algorithm multi-objective optimisation tools. According to the query type, the query cost is calculated, and the two objective functions are computed to find the result for query cost as well as the sensor lifetime cost as presented below.

The output is generated from GA using the multi-objective function in Simulink tools in the MATLAB Simulink tools; as shown in Figure (7-9). After performing the different operations of the genetic algorithm, the histogram rank and Pareto chart showed a robust reduction, meanings that future generation will save on energy when continuously performing sensing. The different GA parameter are tested for best output result, which then the best are chosen for best saving of query cost.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>Uniform</td>
<td>20,000</td>
</tr>
<tr>
<td>Selection</td>
<td>Tournament</td>
<td>2</td>
</tr>
<tr>
<td>Crossover</td>
<td>Heuristic</td>
<td>62%</td>
</tr>
<tr>
<td>Mutation size</td>
<td>Adaptive feasible</td>
<td>-</td>
</tr>
</tbody>
</table>

Table (7-6) lists the parameters of the genetic algorithm used in our experiment. We tested that this genetic algorithm is a suitable solution for solving complex query routing and answering problems in WSNs databases. The algorithm in multiple queries processing primarily schedules the query processing time for huge number of queries, which are requested
on-demand with or without an aggregation function required. The genetic algorithm allows the retrieval of previously available sensor readings from the sensor DB, which is convenient as the number and complexity of queries increase.

The output of the GA is illustrated in Figure (7-9); shows that implementing the genetic algorithm has resulted in saving more energy the chart shows a linear decrease with an increasing number of queries injected into the sensor’s network database. Moreover, it is experimentally tested that both the DA and GA algorithms perform perfectly in choosing the best plan, however GA was expected to outperform DA in terms of time. This theory requires further testing.

Moreover, the two equations used show a strong relationship as they collaborate to minimise the cost function. Regarding query generation time, the results in Figure (7-9) show that the distance between queries attributes decreased steadily over generations. In addition, it explains the importance of sensed data, so the query answers could be retrieved after sensed from the historic database, which specifies the relationship between generated data. A Pareto chart indicates that the increase in the number of queries produces a decrease in sensor battery life. The GA stops when the total population specified reached.

![Average Distance Between Individuals](image1)

![Multi-Objective GA Complex query-base Pareto front](image2)

**Figure 7-9** Query result form for comparing all monitored data
Overall, the proposed query processing method provides a structured and optimal aggregation node method to execute queries in a graph-based network. Moreover, we achieved a 25% improvement in the sensor battery life as compared with conventional sensing methods, by involving two databases, each holding 25% of the overall energy consumed.

We also evaluated a more complex query by requiring data from different parts of the WSN, such as the sensor cache memory, history DB, and WSN DB. The query does not require data from the WSN DB unless it cannot respond with previously retrieved readings. On the other hand, we verified that the sensor nodes do not process RREPs received from nodes at the/or same or lower levels because these RREPs are only used to update the neighbour routing table. Furthermore, when a sensor node received a query, it used its forward routing table to select the next hop; when it received an answer packet, it forwarded the packet in the reverse route of the query reception.

Our query processing method is similar in spirit to the query optimisation presented by Xiang et al. (2007) In fact, this optimisation relies on a similar sensing structure, analyses the user request instead of directly retrieving data from the sensor node, maintains an updated sensor DB, and sets a time interval to gather data using a data aggregation model. Likewise, Markov models are widely used for query processing, and have been integrated into query schedules to determine the optimal sensor node processing.

7.7 Conclusion

The proposed approach combines query processing and real-time sensing with dynamic modelling techniques to optimise query processing in WSNs. Furthermore, dynamic selection and probabilistic confidence intervals are used as query execution principles. The proposed approach is considered a two-tier model according to (Xiang, Lim, K. L. Tan, et al., 2007), with data aggregation and dynamic sensing to acquire high-quality data and guarantee energy efficiency. In fact, we observed that the quality of the data measured from sensors, and consideration of factors such as response time, energy consumption, and communication characteristics in the genetic algorithm could provide high energy efficiency and expect to improve our approach using this type of algorithm. In future work, we will include the a more dynamic environment as the optimality criteria in a genetic-based algorithm approach to test more complex queries using the on-demand approach for acquiring sensors’ essential data readings in a considerable network structure.
Chapter 8 Thesis summary and future work

This chapter presents the main conclusions resulting from the research work of this thesis. Furthermore, it summarises the various research objectives that were achieved and the contributions to knowledge that these make. It concludes by suggesting some open research issues that may be addressed in the future.

8.1 Research Contributions

Due to the complex way in which wireless sensor networks combine a diverse range of technologies, there are many limits to their design and operation. All wireless sensors’ devices built-in technologies such as sensor’s different types are dependents on their initial stages and there is scope for improvement in everything from the design of the battery cells and other hardware in the nodes to the routing algorithms implemented and even the communication protocols used. Within the scope of in-network query processing, some areas and topics have had less development than others.

The current approaches to in-network processing do not support database transaction processing, and most of them do not support ad-hoc queries as an inquiry input tool. Developing an ad-hoc queries approach in such a way as to support the simultaneous processing of queries similar to traditional database transactions would be a worthwhile research effort.

8.2 Achievement of Thesis Objectives

The primary research objective of this thesis was the design, implementation and evaluation of a novel graph-based complex query processing sensing system with energy saving techniques and methodologies. This has been successfully accomplished.

Apart from the current work and algorithm development proposed, the aim to develop a front-end query processor system for the sensor engine to process the data requests for the end user has been achieved. Using the proposed ODQS system it is possible to make any request for data from the sensor network at any time, or when a specific event occurs and at the click of the end user while minimising energy consumption and maximising data quality, without losing any critical events that might occur.
8.3 Thesis Summary

In summary, this thesis has focused on the query processing database in wireless sensor networks; the thesis statement was categorised into different parts and phases. The first part starts with the sampling techniques, which is used to get further knowledge about querying from sensor’s network. Chapter four is focused on adaptive sampling for Wireless Sensor Networks. This chapter studies the adaptive sampling technique in the context of energy efficiency, using household water consumption as a case study. The implementation was performed using a TCP algorithm approach.

The second part of the research work was presented in chapter 5 and proposed a query processing real-time sensing system. For this a real-time query-based sensing system was designed for WSNs. The in-network query processing for distributed query systems is presented. We discussed and reviewed many query processing techniques and various architectures such as tree-based and graph-based and an on-demand real-time sensing models were introduced in detail and assessed for their output advantages. The proposed ODQS system is designed for the purpose of experimental testing and the evaluation of the different aggregation methodologies and merges hypotheses. A sensor database model with a tree-based network structure is introduced.

The third part of the research work, described in chapter 5, included a proposal for an in-network on-demand query-based sensing system for WSNs. The chapter presented a new aggregation and merge approach for query optimisation. The implementation and evaluation of the proposed technique functions and strategies are described, and on-demand queries are tested and compared with un-optimised queries. The tree-based model architecture is also presented in detail. Sensing system comparisons were given for merge and non-merge, aggregate and non-aggregate queries using the ODQS system designed. The sensed data using our approach are available for future reusability in a historic database to save energy of the sensors for fresh data due to injected queries are being always analysed, which means sensor’s database to be retrieved only when available data are out of date.

The fourth part of this research work, the probability model-based on-demand query processing, is discussed in chapter 6. The probabilistic model is adapted to the ODQS system to be used with the historical database to sense only when the model is not sufficient to answer the query. This chapter integrates the novel approach to aggregation methodology for the end...
user requirements. The dynamic programming algorithm was also incorporated to find the last sensed queries at minimal cost and retrieve these for the end user.

The fifth contribution was presented in chapter 7. We proposed a graph-based complex query processing and genetic algorithm for a Wireless Sensor Network database. This chapter tackled the problem of multiple complex queries in a graph-based network structure in order to provide the best aggregation node, saving more energy by generating a more accurate query answer while using fewer hops throughout the network. We then implemented the proposed Dijkstra algorithm and evaluated the output using the well-known genetic algorithm for selecting the best sensor node for the best resulting query plan. In this chapter, we also examined aggregated and non-aggregated queries. Furthermore, our model also analyses whether the query should be answered by the historical database or using the sensor cache database, if not then the query will be disseminated to the network to wait for an answer.

The specified research objectives have therefore been met.

8.4 Future work

This section suggests some directions for future research in improving the performance of the in-network communication cost in wireless sensor network databases, for any network structure. The current ODQS system can be enhanced with more network technical components, to be used as an experimental lab platform since the existing query processing environment is not flexible for user testing. Building the query optimiser using more dynamic characteristics may make the platform more appropriate for transmitting queries to the sensor’s database. While the novel idea presented in this thesis advances the current in-network ad-hoc query processing database systems framework in wireless sensor networks, the opportunities for future studies are presented below:

1. Adaptive sampling algorithm: The developed AS-TCP algorithm has not been tested in a real-time environment. Even though a similar sampling performance would be expected for the AS-TCP to the existing sampling techniques as introduced in literature, some implementation issues should be addressed before the real application is taken place. The first issue is to determine the point at which the algorithm should be implemented, which could be at an individual sensor level or a router level or even a coordinator level. The second issue is that a single sensor may be used to sense multiple variables which may not necessarily have a similar dynamic feature. Some of the sensed variables may allow a quick
response but others may not when an intervention is introduced. In this case, different sets of critical parameters used in the AS-TCP might be necessary. Nevertheless, the AS-TCP proposed in this study could serve as the prototype for real dynamic sensing applications.

2. For future improvement to the on-demand ODQS-DP dynamic probability algorithm, it is also recommended to apply optimality rules, taking account of time, energy and delay. Moreover, additional automation for dynamic aggregation and data sensing is essential to acquire high quality data and further energy efficiency. Moreover, the dynamic shapes in our experiment can be also improved using many other techniques using different case studies such as water sensing system for adaptive sensing. Adding another dynamic module will make the system more automated for sensing only when up normal case is recorded or on events.

3. For future enhancement to the on-demand graph-based approach using a genetic algorithm, it is recommended to include a more dynamic environment such as optimality criteria in a genetic-based approach to test more complex queries using the ODQS approach for acquiring sensors data readings in a heterogenic network structure. Moreover, it is suggested that high-end energy efficiency equations are used to test all the algorithms proposed in this thesis, by measuring all different states of the sensor in detail.

4. The on-demand Dijkstra graph-based theory employed in this thesis retrieved the best query execution plan. Taking account of the requirement for WSNs to be able to support more complicated queries like self-join, top-k, and skylines queries, there is a need to analyse all the different query types in terms of their functionality and accuracy. The queries on-demand approach should be applicable to wireless sensor networks ad-hoc protocol, which can be used to monitor the final sensed data under multiple queries with multiple conditions or operations. Arbitrary graph-based theory is another algorithm that needs to be tested to determine whether it will enhance our results or not.

5. Although dynamic programming produces the most optimal query plans, we need to test iterative dynamic programming as future work, because success using dynamic programming only shows a noticeable performance on generating the query plan considering the cost factor using available specified query plans or aggregation points as in chapter 7. Further research is recommended in this novel area of on-demand query-based iterative dynamic programming.
6. Randomised optimisation strategies such as genetic algorithms avoid the higher time and memory costs. Further advanced use of the applied genetic algorithm should therefore be explored for different query scenarios, such as generating predicted query plans for wireless sensor networks database. Improvements could be made to the current single query optimisation by using a genetic algorithm for multiple-object optimisation, especially when using queries on-demand approach and in a graph-based architecture.

Throughout this thesis on in-network query processing we have found that the field of Wireless Sensor Networks is a relatively new endeavour that is currently undergoing intensive research and development as the number of commercial and scientific applications is rapidly increasing. There are many inherent challenges in creating an extended lifelong and reliable sensor network, and energy efficiency management is vital if a sensor network is to be deployed and stay active for months at a time.

In-network aggregation, merge, probabilistic, and dynamic techniques are discussed and compared, representing significant contributions to the most current published research into in-network data processing. Data aggregation techniques can often reduce data consumption by a factor of three or four and data probabilistic techniques can reduce energy consumption by up to an order of magnitude. Often, a dynamic graph-based technique can be combined with aggregation-based techniques, while complex queries and other in-network processing strategies are able further to reduce energy consumption within the WSN.

As Wireless Sensor Networks are deployed in additional innovative and challenging situations, in both commercial and scientific contexts, demand for sophisticated reliability, stability, security and long-lasting networks is likely to create a need for even more research and development in the field of in-network query processing databases.
References


10.1109/IPSN.2006.243752.


Available at:


118. Wright, D. J. (1986) ‘Forecasting Data Published at Irregular Time Intervals


