Novel Data Routing, Compression and Query Processing Techniques for Energy Conservation in Wireless Sensor Networks

THESIS

Submitted in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Kayiram Kavitha

ID No. 2008PHXF423H

Under the Supervision of

Dr. R.Gururaj

Birla Institute of Technology and Science, Pilani.

2014

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **Novel Data Routing, Compression and Query Processing Techniques for Energy Conservation in Wireless Sensor Networks** is submitted by **Kayiram Kavitha** ID No.2008PHXF423H for award of Ph.D. of the Institute embodies original work done by her under my supervision.

Signature of the Supervisor

Name in capital letters Dr. R.GURURAJ

Designation Assistant Professor

Department of Computer Science and Information Systems

Date:

ACKNOWLEDGEMENT

Completing my Ph.D. degree is probably the most challenging activity of my life. The best and worst moments of my doctoral journey have been shared with many people. It has been a great privilege to spend several years in the Department of Computer Science and Information Systems at BITS-Pilani, Hyderabad Campus and its members will always remain dear to me.

My first debt of gratitude must go to my advisor, Dr.R.Gururaj. He was always there to listen and to provide constructive advice. He taught me how to express my ideas and showed me different ways to approach a research problem. Dr.Gururaj has always given me great freedom to pursue independent work. He is one of the smartest people I know. I hope that I could be as clear in communication as Dr.Gururaj and to someday be able to command an audience as well as he can. During very difficult times of my life, he understood my situation and supported me. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank my Doctoral Advisory Committee members, Prof.Chittaranjan Hota and Prof.Y.Yoganandam, for their very helpful insights, comments, and suggestions. Their guidance has served me well and I owe them my heartfelt appreciation.

In addition, I would like to thank my colleagues who provided invaluable support and suggestions throughout this process. Last, but not least, I thank my parents, for giving me life, for educating me, for unconditionally supporting me throughout.

ABSTRACT

Wireless Sensor Networks (WSNs) play a significant role in applications like-disaster management and human relief, habitat monitoring, studying the weather and eco systems etc. Since the location of deployment of these WSNs is remote in most of the cases, the source of energy is restricted to batteries. Usually, WSNs form either a tree or a graph structure for routing queries and their results. For a tree-based routing structure, Base-Station (a node which has abundant computational, memory, and energy resources) becomes the root node. The applications and end users submit their queries at base-stations. In pull-based data acquisition model, the base-stations adopt appropriate strategies to route these queries to concerned nodes in the network for execution. Nodes execute the queries and send the result back to the base-station. The base-stations compile the final results and send it to the users whereas in push-based data acquisition model, the sensor nodes continuously keep sending the sensed data to the BS. As the energy resources available in the network are limited, all routing and data processing techniques adopted need to be energy efficient to make the network functioning effective and long lasting.

A significant amount of work has been done by researchers in the past to achieve energy efficiency in WSNs. Effective routing tree construction and maintenance schemes have been proposed to construct load balanced routing trees, and facilitate recovery in case of node failures. Next, to reduce the volume of data under transmission, a good number of data compression schemes have been proposed. This would reduce the energy consumption during the data transmissions. Similarly, techniques have been proposed to process the queries in more efficient way by exploiting containment relationship. The primary objective of all the available schemes is to conserve the energy which leads to prolong lifetime of the network. But, we observe certain drawbacks in the existing schemes. The drawback in the existing tree construction and maintenance schemes is that the load distribution among the nodes is not

reasonable and effective. Same is the case with re-organization of the routing tree while recovering from a node failure. Next, when we look at the existing lossy data compression schemes applied to data sets with considerable amount of correlation, they are not so effective in terms of achieving higher precision with minimal loss. We also observe that the existing query processing schemes that exploit containment relationship among the queries do not take the conditions set on time, in the *WHERE* clause. These factors motivated us to carry out our research in the direction of achieving better energy efficiency in the network during data routing, transmission, and query processing activities.

In this thesis, we propose *Workload-Aware Tree Construction* (WATC) algorithm for balancing the load among the nodes in the tree. We propose a novel *Workload-Aware Path Repairing* (WAPR) scheme to recover from node failures in the routing tree in a more efficient way. We also propose a novel *Dual Tree Data Routing* (DTDR) scheme to optimize the power utilization in WSNs. According to this, two data routing trees are constructed for a WSN, where each tree is built using our novel WATC algorithm, for balancing the load among the nodes in the tree. The network switches between these two trees at specified intervals for its operations. This facilitates uniform distribution of load across the nodes, leading to longer network life. All the above proposals are meant for push-based data acquisition in WSNs.

We propose a novel lossy data compression scheme, *Induced Redundancy based Lossy Data Compression Algorithm,* (IR-LDCA) which is best suited for WSNs that sense data with higher correlation. Our algorithm induces certain amount of redundancy into the data set to achieve more effective data compression and also gives the user a flexibility to control the compression ratio and loss of data. This can significantly minimize the transmission costs w.r.t., energy in the network.

In this thesis, we also propose a novel query processing scheme that exploits the cached results at the BS and the commonality among the queries that are to be executed in the network. This would significantly reduce the amount of energy consumed in transmitting data and processing queries at nodes. We have conducted a series of experiments to prove the effectiveness of our proposed schemes.

TABLE OF CONTENTS

S. No.	Content	Page no.
	Chapter 1: Introduction	
1.1	Data Routing, Data Compression and Query Processing in WSNs	2
1.2	Major Issues and Challenges in Data Routing, Data Compression, and Query Processing in WSNs	4
1.3	Our Contributions	7
1.4	Thesis Outline	10
	Chapter 2: Review of Existing Data Routing, Compression and Query Processing Techniques for Energy Conservation in WSNs	
2.1	Routing Strategies	11
2.1.1	Tree-based Routing Schemes	13
2.1.2	Node Failure and Recovery	18
2.2	Data Compression Schemes to achieve Energy Efficiency in Transmission	19
2.3	Query Processing	22
2.3.1	Ad-hoc Query Processing Schemes	23
2.4	Limitations of Existing Approaches	24
2.4.1	Limitations of Existing Tree Construction and Maintenance Schemes	24
2.4.2	Shortcomings in Existing Data Compression Schemes	25
2.4.3	Drawbacks in Existing Query Processing Schemes	25
2.5	Overview of Proposals made in the Thesis	26
2.6	Summary	27
	Chapter 3: Novel Approaches to Construction and Maintenance of Routing Trees in WSNs	

3.1	Introduction	29
3.2	A Novel Energy Efficient Workload Aware Tree Construction Scheme	30
3.2.1	Tree Construction	30
3.2.2	Performance Evaluation	39
3.3	A Novel approach to recover from node failure	44
3.3.1	Performance Evaluation	51
3.4	A Novel Dual Tree Data Routing scheme	54
3.4.1	Data Routing Tree Construction	55
3.4.2	Performance Evaluation	59
3.5	Summary	63
	Chapter 4: Efficient Data Compression Scheme for WSNs	
4.1	Introduction	65
4.2	Induced Redundancy based Lossy Data Compression Algorithm. (IR-LDCA)	66
4.3	Illustration of the IR-LDCA	69
4.4	Performance Evaluation	71
4.5	Summary	75
	Chapter 5: Efficient Query Processing Scheme for WSNs	
5.1	Introduction	77
5.2	Ad-hoc Query Processing on Cached Data at Base-Station	78
5.3	Exploiting Query Containment (Bit-map approach)	82
5.3.1	Identifying Query Containment Relationship	82
5.3.2	Framing super-query-set	84
5.4	Performance Evaluation	85
5.5	Summary	88
	Chapter 6: Conclusion	89
	References	91

Appendix-1 102

List of Publications 109

Biography Statement of the Candidate 111

Biography Statement of the Supervisor 111

LIST OF TABLES

Table No.	Title	Page no.
Table 3.1	Details of the tree structure for the network given in Figure 3.3	40
Table 3.2	Details of example WSN in Figure 3.11	52

LIST OF FIGURES

Figure No.	Title	Page no.
Figure 1.1	Tree topology for WSN	5
Figure 1.2	Disconnection of a portion of the tree due to a node failure.	6
Figure 2.1	Example node deployment	14
Figure 2.2	Example tree structure of WSN	15
Figure 2.3	The balanced tree structure for the initial figure 2.2 (Result of ETC algorithm)	15
Figure 2.4	The redrawn tree from figure 2.3	16
Figure 3.1	Tree topology of WSN	31
Figure 3.2	Example node deployment	32
Figure 3.3	Example FHF tree structure of WSN	33
Figure 3.4	Final workload-aware tree	39
Figure 3.5	Graph depicting the total number of packets transferred till the network dies for a network of 20, 50 and 70 nodes.	41
Figure 3.6	Graph depicting the network lifetime till the network dies for a network of 20, 50 and 70 nodes.	42
Figure 3.7	Graph depicting the total residual power of all nodes in the network for a fixed 40 packets delivered in the network of 20, 50 and 70 nodes.	43
Figure 3.8	Graph depicting the total number of nodes alive for 40 packets delivered in the network of 20, 50 and 70 nodes.	44
Figure 3.9	Disconnection of a portion of the tree due to a node failure.	45
Figure 3.10	Example node deployment	46
Figure 3.11	Example tree structure of WSN	46
Figure 3.12	Graph depicting the total no. of packets transferred till the network dies in the network of 20,50 and 70 nodes	52

Figure 3.13	Graph depicting the network lifetime till the network dies for a network of 20, 50 and 70 nodes.	53
Figure 3.14	Graph depicting the total residual power of all nodes in the network for a fixed 40 packets delivered in the network of 20, 50 and 70 nodes.	53
Figure 3.15	First Routing tree of the network	58
Figure 3.16	Second Routing tree for the example WSN in figure 3.3	58
Figure 3.17	Graph showing the number of packets transmitted in the network in WTC against DTDR scheme	60
Figure 3.18	Graph showing simulation time of the network in WTC against DTDR scheme	61
Figure 3.19	Graph showing the residual power of the network in WTC against DTDR scheme	62
Figure 3.20	Graph showing no. of nodes alive in the network in WTC against DTDR scheme	62
Figure 4.1	Compression ratios achieved for varying values of c for temperature and pressure	72
Figure 4.2	RMS error percentage for varying values of c for temperature	73
Figure 4.3	RMS error percentage achieved for varying values of c for temperature and pressure	73
Figure 4.4	Compression rations achieved in K-RLE and IR-LDCA for temperature	74
Figure 4.5	RMS error percentages in K-RLE and IR-LDCA for temperature and pressure.	75
Figure 5.1	Flowchart showing the workflow for the proposed query processing scheme	81
Figure 5.2	Simple Bit-map showing initial value ‘0’ in all its cells	82
Figure 5.3	Bit-map after plotting Q2	83
Figure 5.4	Bit-map after plotting Q2,Q3,Q4 and Q5	83
Figure 5.5	Graph depicting the no. of packets transmitted in the network for executing 1000 queries	86
Figure 5.6	Graph showing the network lifetime	87

Figure 5.7	Graph showing the no. of queries executed during the lifetime of the network	87
Figure A.1	Screenshot showing the initial tree set-up in a WSN.	104
Figure A.2	Screenshot showing the packet transmission in a WSN.	105
Figure A.3	Screenshot showing the final workload-aware tree.	105
Figure A.4	Screenshot of the node failure and recovery scheme.	106
Figure A.5	Screenshot of the path repairing scheme in the simulator.	106
Figure A.6	Screenshot of the simulator during data compression scheme.	107
Figure A.7	Screenshot showing sample queries during the simulation	108
Figure A.8	Screenshot showing the final results after query processing	108

LIST OF ABBREVIATIONS

WSNs	Wireless Sensor Networks
BS	Base-Station
QRT	Query Routing Tree
FHF	First-Heard-From
IR-LDCA	Induced Redundancy based Lossy Data Compression Algorithm
ETC	Energy-driven Tree Construction
AP	Alternate Parent
APL	Alternate Parent List
CNL	Child Node List
ß	Branching Factor
RLE	Run Length Encoding
MHS	Minimum-Hot-Spot
DTDA	Dual-Tree-based Data Aggregation
LTC	Light-weight Temporal Compression
DTDR	Dual Tree Data Routing
WATC	Workload-Aware Tree Construction

WAPR	Workload-Aware Path Repairing
RMS	Root Mean Square
GPS	Global Positioning System
LZW	Lempel-Ziv-Welch
S-LZW	LZW for Sensor nodes
AWL	Average Workload
Λ	Threshold
SF	Suitability Factor
FRT	First Routing Tree
SRT	Second Routing Tree
FBS	First Base-Station
SBS	Second Base-Station

Chapter 1

Introduction

With the advent of automation in monitoring systems, the sensor devices are widely used. A sensor is a tiny electronic device with a sensing component, small microcontroller, energy source (usually battery), and a radio transceiver for wireless communication. In general, each sensor device is small, lightweight, and portable. In real-time, sensors are used to monitor physical parameters such as temperature, pressure, humidity, illumination intensity, sound intensity, vibration intensity, wind direction, speed, pollutant levels etc. They measure different parameters from the environment and transform them into electrical signals. The microcontroller is used for data processing activities while the radio transceiver performs communication operations. A wireless sensor device can communicate with other sensor devices within its transmission range. Therefore, many such wireless sensor devices collaborate to form a Wireless Sensor Network (WSN) [1-4], and each sensor device in such network is called as a *sensor node*.

In the early days, the WSNs were designed for military applications [5-7] such as battlefield surveillance. Nowadays, the WSNs are widely used in many industrial, and scientific applications like- Waste Water Monitoring, Green House Monitoring, Air Pollution Monitoring, Machine Health Monitoring, Landslide Detection, and Forest Fire Detection etc. One of the primary advantages of using a WSN is its ability to operate unmanned in remote geographic locations, extreme weather conditions, hazardous work environment etc. The source of power for each sensor node to operate is driven from a self-equipped battery. Each sensor node expends energy in sensing, computing, and data transmission activities. Further, observations reveal the fact that the energy consumed in transmitting one bit of data is approximately equivalent to that of energy consumed for processing 1000 instructions [8] in a sensor node. Hence, data transmission is the most energy demanding operation in a WSN. The battery power of each sensor node is limited and cannot support WSN operations for longer duration. Hence, battery needs to be replaced when its power is exhausted. As applications of WSN can be found in hard-to-reach geographical locations, it is extremely difficult or impossible to replace battery. To address this issue, it is highly desirable to utilize the power at each sensor node in a conservative manner.

In this thesis, we propose novel approaches for energy efficient data routing, data compression, and query processing in WSNs.

The objectives of this thesis are as follows.

1. Review the state-of-the-art techniques for data routing, data compression, and query processing in Wireless Sensor Networks.

2. Propose novel energy efficient data routing tree construction and maintenance schemes with emphasis on effective load balancing among the nodes, and recovery from node failures.

3. Propose a novel data compression scheme to reduce the data transmission costs w.r.t., energy consumption.

4. Introducing a novel query processing scheme to conserve the power in the network by exploiting result caching, and query containment.

5. Validating the proposed schemes for their effectiveness.

1.1 Data Routing, Data Compression and Query Processing in WSNs

In a WSN, one of the nodes is designated as the *Base-Station* (BS). This BS node connects the WSN with the outside world. Further, a base-station is assumed to possess unlimited energy, memory, and processing power. Each sensor node is also equipped with limited computation, storage, and communication capabilities. The sensor nodes sense data continuously and store in their built-in memory and/or transmit to the BS.

In query-response model (pull-based data collection) [9], the queries submitted to the BS by end-users/applications may be either *ad-hoc* [10] or *continuous* [11]. In case of ad-hoc queries, it requires one time execution only. For example, '*Give the maximum temperature today'*, is an ad-hoc query. On the other hand continuous queries refer to such requests which require processing at some specified intervals of time or on some event. For instance, the query '*Give the average temperature for the last 10 hours on daily basis'* is a continuous query. Hence, all the queries submitted by the external world first arrive at the BS. If these queries can be answered with the data available at the BS, the BS will send the results to the requester after processing the query. In case the data is not available at the BS, then they are routed to the concerned nodes in the network. After processing, nodes will send the results back to the BS from where the same is sent to the requester.

If it is a push-based data acquisition model [9], the sensor nodes keep sending (pushing) the sensed readings for the specified phenomenon, to the BS continuously or at some specified intervals. This is not in response to any query.

Usually, WSNs form either a tree [12, 13] or a graph [14-16] structure to formulate data routing paths. Here data could be of the following forms. If it is a query-response model, the content of the query (sent from BS to node) or the results (sent from node to BS), could be the data encapsulated into network packets. If it is a push-based continuous data acquisition model, the series of sensed data readings sent by nodes to the BS form the data packets. In addition to the above, data could be in the form of control messages that are transmitted in the network for the purpose of maintenance or management of the sensor network.

If a tree structure is formed for data transmission, the BS acts as the root node of the tree, and the data routing takes place along the routing paths. According to this, each node has a single routing path to the BS, which includes all its ancestors. If a graph structure is formed, multiple paths can exist between any two nodes (including the BS) in the network.

In general, sensor nodes are deployed densely in the area of interest for satisfactory coverage. In many applications this spatially dense deployment of sensor nodes capture highly correlated data [17, 18]. Some of the WSN applications such as environment monitoring systems [19-22] may require sensor nodes to continuously sense and transmit the sensed data in fixed time interval. Hence, the data in such applications mostly exhibit temporal correlation [23-25]. Temporal correlation refers to the relationship (in terms of proximity of the values) that exists between two consecutive data readings. If the above mentioned temporal correlation between the sensed data readings is exploited, the quantum of data to be transmitted can be reduced considerably. To achieve this, data compression schemes [26, 27] are applied at node level.

We present the major issues and challenges in data routing, data compression, and query processing in WSNs in the next section.

1.2 Major Issues and Challenges in Data Routing, Data Compression, and Query Processing in WSNs

Many WSN applications [28-30] form tree topology for communication and as a result, each communication from the BS to a sensor node and vice-versa will follow a specified path. In Figure 1.1, a sample communication tree structure in a WSN is shown. The BS is always considered as the root node. Each node has a routing path to the BS, which includes all its ancestors. In the context of above tree topology for data routing, and push-based data acquisition scenario, where nodes continuously transmit the sensed readings to the BS, each node has to send its own sensed data, and forward the data of its children. Thus, the intermediate nodes are always taxed with extra burden. The extra burden taken by an intermediate node is directly proportional to the size of its sub-tree. Such intermediate nodes tend to expend more energy and die early. Even in case of query-response model, the intermediate nodes are always taxed with extra burden of query dissemination and result transmission.

In ad-hoc query processing, for a given query that involves certain nodes in the network, communication between BS and the nodes happens by following specific routing paths. All such routing paths, between a BS and nodes collectively form a *Query Routing Tree* (QRT) [31-33] for the respective ad-hoc query. Majority of the data acquisition approaches [34-39] proposed so far, adopt QRTs that provide routing paths for data transmission in the network.

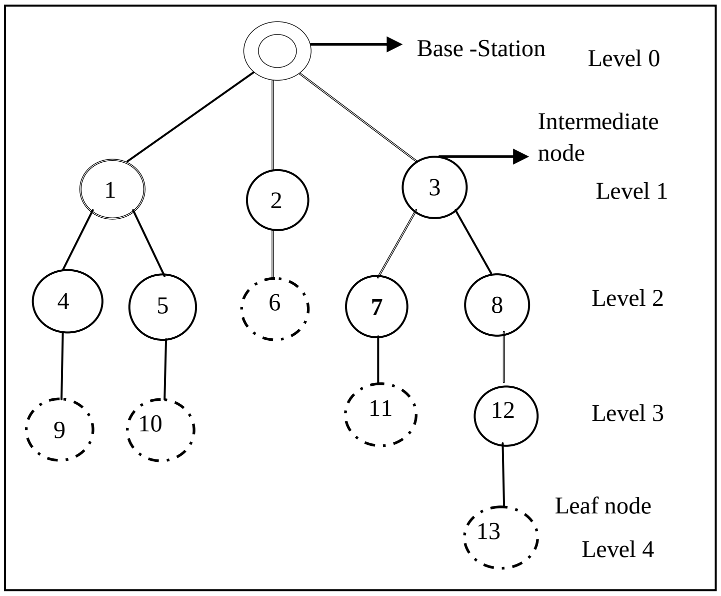


Figure 1.1 Tree topology for WSN

The First-Heard-From (FHF) [40] is one of the basic and popular approaches for ad-hoc tree construction. In this scheme, the BS sends a '*hello*' message to the nodes within its transmission range. The sensor node which ever first hears the '*hello*' message becomes the *child* (dependent). This child node confirms the relationship with the sender with an '*ack*' message. The child nodes further send the '*hello*' message to the nodes within its transmission range to form a network. This process is continued recursively until all the nodes are connected to form a tree. This ad-hoc tree construction scheme does not consider the shortest path or minimum hop-distance from a sensor node to reach the BS. In our experimentation with trees of varying sizes, we found that this approach leads to energy wastage due to inefficient data routing in WSN.

Next, when a node fails due to zero battery power or any other reason, its sub-tree loses connectivity with the BS. Hence, the sub-tree is totally cut-off from the network. As the time of failure of a node is not known in advance, no alternative arrangements could be made to handle this situation. Many applications come to a halt due to such node failures. The failure of an intermediate node results in disconnection of its entire sub-tree from the WSN. We call such portion as *disconnected sub-tree* and a sample is shown in Figure 1.2. As a result of this, the residual battery power of the entire sub-tree remains unutilized. This reduces the effectiveness of the WSN.

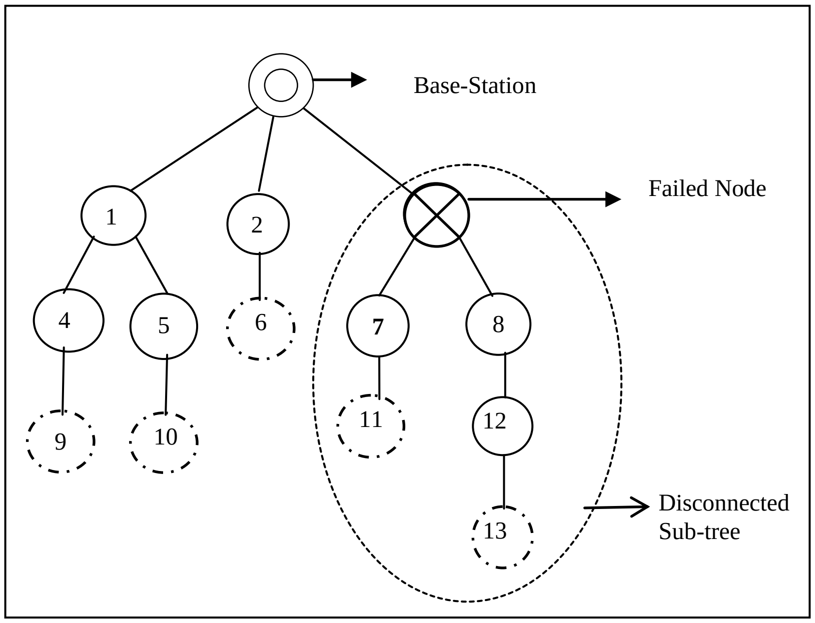


Figure. 1.2 Disconnection of a portion of the tree due to node failure.

Further it is observed that there is a huge wastage of power in the transmission of correlated data. As the data transmission is a highly energy consuming task in WSN, we understand that there is a need for exploiting data correlation to achieve energy efficient transmission.

Next, in query-response model, a set of ad-hoc queries submitted at BS are often found to be requesting the same results. Sometimes they may not be equal, but they exhibit certain overlap in their query results. Such relationship among queries is termed as *query containment* [41, 42]. The containment relationship [43] is said to be satisfied, iff two queries yield same or partially same query results. We strongly believe that this containment relationship among the queries can be exploited to minimize the volume of data transmitted, and to some extent the number of computations in the network.

Our work addresses the above issues and challenges related to data routing, data compression and query processing in WSNs. In the following section we present a brief account of our contributions made in this Thesis.

1.3 Our Contributions

Now we present the contributions of this thesis. To achieve energy efficiency in WSN, we propose – (i) novel energy efficient data routing tree construction and maintenance schemes for push-based data acquisition model, (ii) a novel data compression scheme for energy efficient data transmission, and (iii) an efficient ad-hoc query processing scheme that exploits cached data at BS, and query containment relationship among queries to minimize the number of query/result transmissions in the network.

i). Novel Schemes for Energy Efficient Tree Construction and Maintenance

When a tree topology is adopted for data transmission in push-based data acquisition scenario, the intermediate nodes always take additional load due to the responsibility of forwarding the data from its children, to the BS. Due to this reason, the intermediate nodes closer to the root (BS), and/or having considerably large number of descendants die sooner than the nodes which are relatively closer to the leaf level, and/or with lesser number of descendants. The failure of an intermediate node results in disconnection of its entire sub-tree from the WSN, and the network loses services of the portion (sub-tree) of the network for which the failed node is the root. As a result of this, the residual battery power of the entire sub-tree remains unutilized. This reduces the effectiveness of the WSN. To overcome these problems workload balanced tree construction approaches [44] have been proposed in the past. But, we observe that the solutions provided were not that comprehensive. Further reconnection of this disconnected sub-trees to the network is another challenge. Though few schemes [45, 46] have been proposed in the past to address this, none of them are effective with respect to load balancing.

Now, in this thesis we propose the following schemes for effective tree construction and maintenance in WSNs.

a) **A novel *Workload-Aware Tree Construction* (WATC) scheme**, where workload balancing is done by our novel technique that computes the workload of every node based on the number of child nodes. As the nodes are arranged level-wise, the average workload associated with each level is computed to compare it with workload of the individual nodes at that level. If a node is found to possess workload more than that of the average workload, then some of its child nodes can be attached to other nodes with the least workload at the same level. This way, a tree is balanced with respect to workload, having the load distributed uniformly among the nodes at a given level.

b) **A novel *Workload-Aware Path Repairing* (WAPR) scheme,** to address the issue of recovery from node failures in WSNs. The purpose of this scheme is to reconnect the disconnected sub-tree (which is a result of an intermediate node failure due to power outage) to the main network. In this scheme, we try to optimize the communication cost by applying tree reorganization technique that minimizes the number of hops from the disconnected sub-tree to the BS and also balance the workload among the intermediate nodes while choosing an alternative path for the disconnected sub-tree. When a node fails, the tree is re-organized by distributing the workload uniformly.

c) **A novel *Dual Tree Data Routing* (DTDR) scheme,** for push-based data acquisition model to achieve improved network lifetime. According to this scheme the network will have two base-stations (located opposite ends of the network) and two routing trees with one base-station per tree. Each routing tree is built by distributing the workload uniformly among the nodes in the tree, using our novel WATC scheme mentioned above. The network switches between these two trees at specified intervals for its operations. According to this the role played by a node in one spell is reversed in the succeeding spell with respect to the level/depth of the node in the tree. This DTDR scheme facilitates uniform distribution of load across the nodes, leading to longer network life.

All the above mentioned proposals intended for effective data routing tree construction and maintenance are elaborated in Chapter 3 of the thesis.

ii). A Novel Technique for Data Compression to achieve Energy Efficiency in Transmission

To reduce the data transmission activity in the network, the data compression schemes are applied at node level. The data compression schemes can be broadly classified into two categories: *lossless compression* [47, 48] and *lossy compression* [49, 50]. With lossless compression, original sampling data can be perfectly restored at the receiving end i.e., without any loss in the precision of the data. But this hinders achieving higher compression ratios. With lossy compression, some degree of information loss in terms of *Root Mean Square* (RMS) error is present. For the WSN applications, which doesn't require high precision, lossy compression techniques are more preferable. We have proposed a new lossy compression scheme called as *Induced Redundancy based Lossy Data Compression Algorithm* (IR-LDCA) to compress the data under transmission in the network. Our novel compression algorithm is presented in Chapter 4 of the thesis.

iii). A Novel Scheme for Energy Efficient Query Processing

In this thesis, we present a novel query processing technique for ad-hoc queries that exploits cached results at BS and query containment among the queries to be executed in the network. If the query results can be obtained in its entirety from the cached results then we execute that query against the cached results at the BS and send the results to the user. Otherwise if the query results need to be extracted partially or completely from the network nodes, then we figure out the commonalities w.r.t., the data requirements of all such ad-hoc queries and then formulate a set of queries which we call as a set of *super queries*, and transmit the same to the required nodes for execution. The result set of each super query is a super-set of the results of all the queries that the super query contains. We present a bit-map approach for identifying the commonalities (containment) among the queries and formulating a set of super queries. This helps in minimizing the number of queries executed and the volume of results transmitted in the network which eventually leads to optimal utilization of the power and longer network life. This approach is explained in Chapter 5.

1.4 Thesis Outline

The outline of the rest of the thesis is organized as follows. The Chapter 2 reviews the existing schemes for data routing tree construction and maintenance, data compression, and query processing in WSN. Chapter 3 provides complete details of our novel data routing tree construction and maintenance schemes along with the analysis about the performance. The Chapter 4 details our proposed lossy data compression scheme IR-LDCA, and its performance analysis. The Chapter 5 explains our novel query processing scheme with necessary discussion about the effectiveness. Finally, the Chapter 6 concludes this thesis after summarizing our major contributions, and providing future directions for research in this area.

Chapter 2
Review of Existing Data Routing, Compression and Query Processing Techniques for Energy Conservation in WSNs

This chapter reviews the literature which is relevant to our work presented in this thesis. In the recent past, several schemes have been proposed for constructing data routing structures, data compression, and query processing in WSNs, which are meant for achieving energy efficiency. The focus of this chapter is to review existing schemes for data routing tree construction and maintenance, data compression, and query processing in WSNs. Finally, the chapter concludes with our proposals for novel approaches to – (i) energy efficient data routing tree construction and maintenance in push-based data acquisition model, (ii) effective data compression for energy efficient data transmission, and (iii) effective ad-hoc query processing that exploits cached data at BS, and query containment relationship among queries to minimize the number of query/result transmissions in the network.

2.1 Routing Strategies

Data routing is an important issue in energy constrained WSNs. In this section, we discuss various routing strategies and related issues for WSNs. In a WSN, the transmission range for each sensor node is fixed. A sensor node will be able to communicate with another node in the network only when its transmission range is intersecting with that node. So, this situation gives rise to a graph like communication structure in the network. Sensor nodes become vertices and the communication links become edges in the graph. In early days, the communication between any two nodes used to happen by flooding [51, 52] the data packets in the network. According to this, each node forwards its incoming packets to all its connected neighbors except to the one which sent this packet. Flooding guarantees packet delivery to every node. But, each node may receive multiple copies of the same packet. Due to this redundancy in packet transmission, there is a huge wastage of power. To address this issue, topology-based routing [53] was introduced. In this scheme, each node has a unique identification, and between any two nodes the logical routing path is predetermined and fixed. These routing paths are maintained in a routing table at every node. Any change in the topology is communicated to all the nodes for keeping the routing information up-to-date. Each node forwards data to the next node en-route destination by consulting the routing information stored locally. As every node is proactive and stores the information about the changes done to the routing paths of the network, such scheme is called as *proactive routing* [54]. This fixed path avoids redundant transmissions. Each change in the topology is propagated to the routing tables stored at all the nodes in the network. This is an energy consuming activity. To overcome this, *reactive routing* [55] was introduced. In this, each node generates a routing path to the destination when needed. Hence, a node needs to maintain only the current path information which is in use, and discard the previous ones. Since, the topology changes are rapid and unpredictable, a large part of network bandwidth and energy are required in maintaining the routing information at nodes. To overcome the above mentioned limitations of topology-based methods, *location-aware routing* [56] was introduced, where the nodes are aware of their physical location. By means of global positioning techniques [57] like Global Positioning System (GPS), the physical location of a node may be determined. This makes sensor nodes addressable and aids in data routing. Each node computes and broadcasts its location information. During packet transmission, the source node first identifies the geographic location of the destination node and sends the packet to some node in the same direction to that of the destination node. In this routing scheme, route discovery is done on demand and hence conserves network power to some extent. In the routing schemes discussed above one major drawback is that the routing path between two nodes is not fixed and is computed on demand. This process always consumes energy and computational resources. To avoid the above problems, tree-based routing structure was investigated. Now, in the following sections we discuss the existing data routing schemes for WSNs that adopt tree topology.

2.1.1 Tree-based Routing Schemes

A tree structure can be seen as a special graph where each node has only one parent and entire structure has one root node (usually the base-station). In general, the deployment of sensor nodes in the geographical plane forms an arbitrary graph in terms of communication links. To organize sensor nodes of this arbitrary graph in hierarchical order, the routing trees are constructed by identifying the children for each node starting from the root (BS). In this tree, every node has exactly one path to the BS, with certain intermediate nodes, which are ancestors of the node. Now, we brief on the existing tree-based routing schemes in WSNs.

To maximize network lifetime, an energy efficient *spanning-tree based multi-hop routing technique* [58] was proposed. In tree-based routing structures every node has a defined path to the base-station. According to this spanning tree-based routing tree, each routing path spans through all the nodes of the network without forming a cycle. In this scheme, routing tree is constructed using Kruskal algorithm [59]. Although this gives an efficient routing tree, frequent use of one single path may lead to failure of nodes on the path much earlier than other nodes. Hence, there is a need for a collection of trees and use each of them for a fixed number of rounds so that energy consumption is balanced among all the nodes in the network. However, nodes nearer to BS may still die sooner than other nodes in the network.

The Energy-driven Tree Construction (ETC) [40] algorithm, which is an ad-hoc tree construction scheme, uses First-Heard-From (FHF) [40] approach in its first phase, to construct a query/result routing tree for a WSN, where each node after hearing from other nodes within its transmission range, selects one among them as its parent. Each of the remaining nodes will become an *Alternate Parent* (AP) for that node. The list of such alternate parents for a given node is maintained as the *Alternate Parent List* (APL). Each node stores its APL and *Child Node List* (CNL) locally. In ETC algorithm, the maximum number of children for a node is indicated by *branching factor (ß),* which is considered to be the threshold value to indicate the maximum number of children a node can have. The branching factor (*ß)* is calculated at the BS using the formula *d*√*n* where *d* is depth of the tree and *n* is number of nodes in the tree. Later, in the second phase, this *ß* value is disseminated to all the nodes in the tree for load balancing. The candidate nodes for balancing are those which exceed the threshold *(ß)*. The excessive workload of the candidate nodes is distributed amongst the other nodes of the tree. The candidate node instructs some of its children to look for a new parent. Then such child nodes will select the first node of their respective APLs as their new parent. This process of balancing the workload continues for all nodes of the tree.

To illustrate the ETC algorithm, we consider a set of 11 nodes deployed as shown in Figure 2.1. First, we explain the initial tree construction using FHF approach as given in ETC. The Base-Station (node 0), sends a '*hello*' message to all its neighbors (nodes within transmission range) i.e., nodes 1, 2, 3. Each of these child nodes reply with '*ack*' message back to the parent node (node 0). This is to confirm the parent-child relationship.

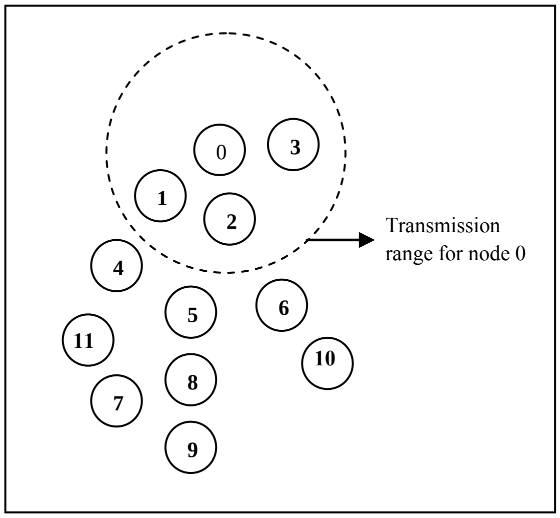


Figure 2.1 Example node deployments

Further, nodes 1, 2 and 3 send '*hello*' message to other nodes within their transmission range (except to their parents). This process continues till it reaches the leaf nodes of the network. The initial tree structure formed is shown in Figure 2.2. When an external entity wants to query the WSN, submits its query to the BS. The BS then transmits the received query to the concerned nodes. Each node processes the query on its local data and sends the results back to the BS. The network lifetime is the time elapsed between starting of the network and the moment it halts. A network is considered to have reached a halt state when not even a single node is connected to the BS. Otherwise, we say that the network is alive if at least one node in the network is active and connected to the BS. Hence, the network lifetime is highly dependent on the battery life of its nodes.

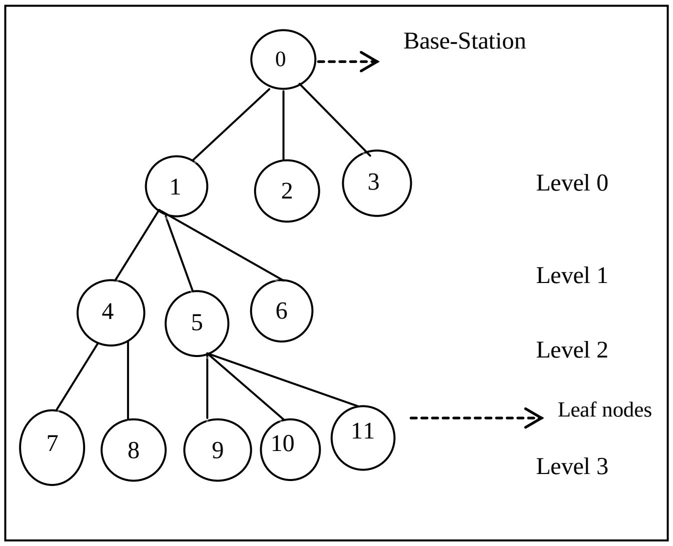


Figure 2.2 Example tree structure of WSN.

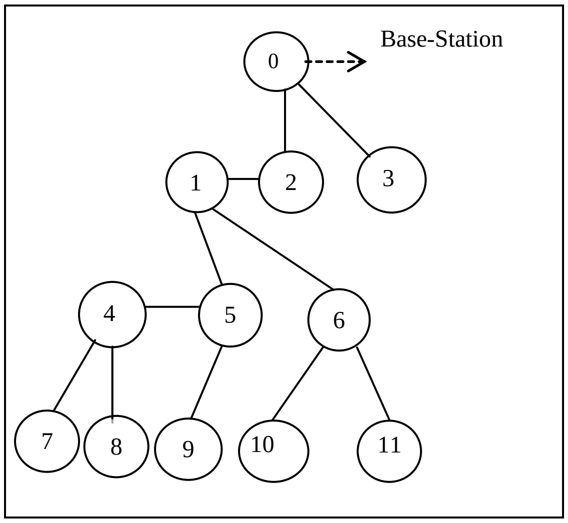


Figure 2.3 The balanced tree for the initial tree in Figure 2.2 (Result of ETC algorithm)

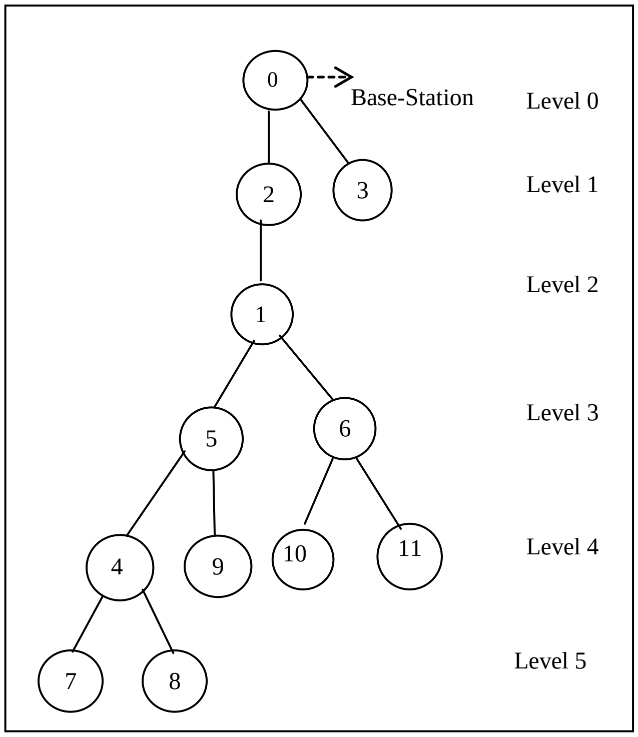


Figure 2.4 The redrawn tree from Figure 2.3

If we analyze the energy consumption rate of every node in the network, where the query load on each node in the network is uniform, we find that the intermediate nodes are being taxed more when compared to leaf nodes of the tree. In our example, we may note that the intermediate nodes of the tree at level 1 (node 1) and at level 2 (nodes 4, 5) not only transmit self-data, but also take the responsibility of forwarding the query results produced by their descendants. The parent-child relationship among the nodes is also depicted in Figure 2.2. This is how the initial tree is formed as per the FHF approach.

In the ETC process, it is evident that the reorganization of the tree is done only to distribute the child nodes of an intermediate node whose branching factor exceeds the *threshold* (ß), among the other suitable nodes of the tree irrespective of the level.

The ETC algorithm when applied on the initial tree of the WSN shown in Figure 2.2 yields the resulting tree shown in Figure 2.3. The Figure 2.4 is the redrawn tree structure for the tree shown in Figure 2.3.
Following the same idea of controlling the number of children per parent, the work in [60] proposes an algorithm for load-balanced tree construction. The input to the algorithm is an ad-hoc tree constructed using FHF approach. In ad-hoc query processing, for a given query that involves certain nodes in the network, communication between BS and the nodes happens by following specific routing paths. All such routing paths, between a BS and nodes collectively form a *Query Routing Tree* (QRT) [31-33] for the respective ad-hoc query. In this scenario, if multiple QRTs are formed for executing different queries, in some cases some nodes may be part of more than one query tree. In such cases some nodes perform redundant transmission of packets due to many incoming streams. Such nodes are termed as *hot-spots*. The retransmission at these hot-spots leads to energy wastage in the network. Therefore, the *Minimum-Hot-Spot* (MHS) algorithm proposed in [60] aims at minimizing hot-spots during tree construction. This leads to reduced packet collision during query execution and improves energy efficiency in the network. MHS employs a distributed tree balancing process, where the degree of each node is balanced for each tree depth. Each node chooses a parent with least number of children. It is obvious that the parent with least number of children is the best choice. First the node with fewest choices is asked to choose its parent. While the node with many options, however can get the best parent (node with less children). At each node, the number of children (degree) needs to be minimal, to make sure that the node will not become a hot-spot.

A Dual-Tree-based Data Aggregation (DTDA) scheme for grid-based WSNs is proposed in [61]. The DTDA scheme uses dual tree structure for data transmission. The scheme addresses the load balancing problem in WSNs. The network is partitioned into logical grids. Whenever a sensor node detects an event, it alerts all its one-hop neighbors. If the receiving node detects an event they make an entry in their event table and further alert their one-hop neighbors else it discards. This way a group of nodes detecting an event form a region of interest. The nodes in a given region of interest select two appropriate nodes with higher residual energy in the region of interest as base-stations. The node with higher residual energy will live longer than others. So, two such root nodes in the region of interest will form their individual trees. This tree construction is again based on residual power. Roots prefer nodes with higher residual energy as their children. First the root broadcasts request to its one-hop neighbors along with its residual energy and number of nodes in this tree. Each node receiving such request will check if it is a leaf or non-leaf. If it happens to be a non-leaf node, it forwards the request. In case if a node receives two requests, it will choose the tree with minimum number of nodes. Now, every parent node confirms a child node with higher residual energy. Now, two trees are established. The two roots send their aggregated data to the mobile sinks by two separate routing paths. Thus, the two trees operate simultaneously throughout the network lifetime.

2.1.2 Node Failure and Recovery

In the routing tree constructed using the above mentioned ETC algorithm, when a node fails, the dependent nodes lose their parent, the dependent sub-tree is totally cut-off from the network. Many applications come to a halt due to such node failures. In this section we present some of the important path repairing schemes meant for recovery from failure in a WSN.

The work in [45], proposed an algorithm to handle arbitrary node failure in a WSN with tree topology. Each node pre-computes its only alternate parent based on the neighborhood information received through piggybacking the query-response messages. This approach helps in reducing the communication overheads as it reduces the number of extra message transmissions, which eventually helps in saving the battery power. When a parent node fails, all its children can directly contact their respective alternate parent to establish the alternate path to the root (BS) of the tree. Here path repairing is done correctly within a constant round of message transmissions. In this approach, there exists only one alternate parent for every node. Sometimes this alternate parent might have already been overloaded, and the inclusion of the new child may further worsen the situation. As a result, the battery lifetime of this new parent gets exhausted faster due to this added workload. This algorithm does not consider the case when the alternate parent as well fails. Further, a node failure due to power outage is not predicted.

The work described in [46], proposes a dynamic route discovery technique that establishes a new path to the BS from a node when its parent fails. In this, the node looking for a new parent considers only those nodes among its neighbors, which have a link to the BS through some path. Once the set of alternate parents is found, the node will select one of them as its parent such that the number of hops from the selected parent to the BS is minimal. This path repairing is done dynamically to establish a new path to the BS. This mechanism consists of 4 steps- (i) failure detection, (ii) failure information propagation, (iii) identifying alternate parents, and (iv) new parent selection. First, a node detects if its parent is alive and can connect to the base station. This is done by exchanging a set of messages between a node and its parent. If it finds that the parent is not alive, then it initiates the path repairing process. It sends a request to all its neighboring nodes. If a neighboring node has a path to the BS, it will send back a message (that also includes its hop-count to the BS) to the sender node. These set of nodes from which the positive response has been received become alternate parents for the requesting node which is looking for a new parent. Finally, a new parent is selected such that the number of hops to the BS is smallest among all the candidates. The above algorithm also addresses the issue of eliminating loops that are possible in the new routing path generated. Hence the above technique makes the WSN fault-tolerant.

2.2 Data Compression Schemes to achieve Energy Efficiency in Transmission

The sensor nodes are deployed densely in the area of interest for satisfactory coverage. This dense deployment results in multiple sensor nodes employed for sensing single event. Temporal correlation is the closeness/similarity that exists between two data values pertaining to two consecutive time points. The spatially dense sensor nodes capture highly correlated data. The degree of correlation increases with decrease in inter-node separation distance. Some of the WSN applications such as environment monitoring systems may require sensor nodes to continuously sense and transmit the sensed data in fixed time interval. Hence, the data in such applications mostly exhibit temporal correlation [20-23]. To reduce the data transmission activity, data compression is performed at node level, by taking temporal correlation into consideration. There has been significant amount of work on data compression for sensor networks. Many algorithms exploit the natural correlation existing in the sensor data. This data compression will result in reduction in volume of data during the transmission and hence, can reduce power consumption in the network. The data compression schemes employed in WSN can be classified into two. One is *lossless data compression* and the other is the *lossy data compression* scheme.

Lossless Data Compression Schemes

The lossless algorithms, as the name indicates compress the data without any loss. Hence, applications requiring higher precision adopt these schemes. It is obvious that to achieve higher compression ratios, we may need to compromise on precision.

The Modified Adaptive Huffman Coding algorithm [47] is a lossless data compression algorithm. This algorithm uses a tree approach where the leaves of the tree represent sets of symbols with the same frequency. In this approach, the number of levels in the tree is kept as minimum as possible. This algorithm uses the above mentioned tree to assign a smaller bit-sequence representation to the symbols with higher frequencies and similarly a larger bit-sequence for the symbols with lower frequencies. It was observed that the compression ratio of this algorithm is significantly less, and is beneficial only when handling highly correlated data.

The Lempel-Ziv-Welch (LZW) [62] is a dictionary based lossless data compression algorithm that builds its dictionary as the data is read in from the input bit stream. S-LZW (LZW for Sensor nodes) [48], which is an extension to LZW, splits the uncompressed input bit stream into fixed size blocks and then compresses each block independently. For each new block the dictionary used in the compression is re-initialized by using the 256 codes which represent the standard character set.

The Run-Length Encoding (RLE) [64] is a basic lossless compression algorithm. The simple idea behind this algorithm is to replace the repeated consecutive occurrences of the same data with a single value pair. For example, if any data item is represented by *m*-bit pattern, and a data item *d* occurs *i* (represented by *n*-bit pattern) consecutive times in the input stream, that series can be replaced by '*i d*' which needs *n*+*m* bits only instead of *m*i* bits. The main problem with this algorithm is its low compression ratios since two values having decimal representation may be close but not exactly equal.

Lossy Data Compression Schemes

Here, we give a briefing on the existing lossy data compression schemes for WSNs. In contrast to lossless compression, lossy compression schemes exhibit higher compression ratios with certain amount of loss in precision.

The work in [48] describes a lossy compression algorithm which is known as Light-weight Temporal Compression (LTC) scheme for WSN. The algorithm exploits the fact that the captured readings for microclimate data, in a small window of time, are linear in nature. It identifies such windows and generates a series of line segments that accurately represent the data. This scheme performs compression by introducing error bounded by a *control knob*, which is in the order of the error specified on the hardware. This algorithm attempts to represent a long sequence of similar data with a single symbol. It is effective on data sets which are largely continuous and changes in readings are infrequent. Thus, the results of LTC show that it performs better on the data related to temperature than on humidity or wind speed. This shows that the compression ratio in LTC is highly dependent on the nature of the data. The LTC algorithm is designed for mica motes (sensor hardware) with 8-bit processor, which has no capability to handle floating point values. This limits the applications of LTC to compression of only integer data.

The work in [49] describes K-RLE, a lossy compression algorithm. K-RLE is a variant of RLE algorithm. It shows increased compression ratios compared to RLE, but with certain amount of error (data loss). The performance of this algorithm depends on the choice of the value of the parameter *K* which represents the precision. This algorithm emphasizes on processing the data locally at node level. In this, if a data item *d* or a data item between *d+K* and *d-K* occurs for *n* consecutive times then the occurrences are replaced by a single pair '*n' 'd'* (as illustrated in case of RLE algorithm in the previous discussion). If *K=0*, then K-RLE is RLE. This *K* value makes the K-RLE lossy compression algorithm, leaving RLE a lossless algorithm. The choice of *K* also influences the percentage of data and the extent to which it is modified by this algorithm.

2.3 Query Processing

As the *Base-Station* (BS) links the WSN with the outside world, in query-response model (pull-based), all queries are submitted at the BS. In general, the queries submitted by end-users/applications may be either ad-hoc or continuous. In case of ad-hoc queries, it requires one time execution only. For example, '*Give the maximum temperature today*' is an ad-hoc query. On the other hand continuous queries refer to such requests which require processing at some specified intervals of time or on some event. For example, the query '*Give the average temperature for the last 10 hours on daily basis'* is a continuous query. In general, both BS and the sensor nodes are capable of processing queries. The only difference between a sensor node and BS is in terms of the resources available. Hence, all the queries submitted by the external world first arrive at the BS. If these queries can be answered with the data available at the BS, the BS will send the results to the requester after processing the query. In case the data is not available at the BS, then they are routed to the concerned nodes in the network. After processing, nodes will send the results back to the BS from where the same is sent to the requester.

To achieve energy efficiency, there have been numerous schemes [34-39] for query processing in WSNs. Since, our focus in this thesis is on ad-hoc queries, now we explain some popular existing query processing schemes for WSNs.

2.3.1 Ad-hoc Query Processing Schemes

The query processing requires message transmission in WSNs both for query dissemination and result collection. For energy constrained WSNs, the goal is to reduce the number of message transmissions. The scheme proposed in [65] is for processing aggregate queries with the help of domain knowledge and sensing constraints. An aggregate query may involve one or more of the operations like- *average, sum, min, max, and count*. For instance, consider the query '*Is the average of the temperatures of the nodes n_1 and n_2, greater than 80'*. To process this query, the BS collects temperature from n_1 and n_2 and computes the average and checks if it is greater than 80. The result of this query is Boolean. If the domain semantics specifies that the temperature at any node is between 0 and 100, this knowledge can be used by BS in simplifying the execution. For example, to process this, BS can retrieve one of the temperatures first and check if it is less than 60. If so, it tells that independent of the temperature reading at the other node the result is false. In that case, it need not send and process the query for n_2. This way unnecessary query/result transmission and processing are avoided. But, this approach can be applied only for certain queries, which use universally fixed knowledge and constraints.

The Query model in [66] allows queries to specify geographical area rather than explicitly stating single data sources. Each query region is specified as a bounding rectangle using co-ordinates. Two queries q_1 and q_2 are said to exhibit containment relationship, when the query q_1 is found to request data from the sensor nodes within the geographical boundaries specified by q_2 or vice-versa. This is identified by constructing a R-Tree, which stores the bounding rectangle co-ordinates for each query. This R-Tree is used to identify the largest rectangle that contains all other rectangles. Now, the BS will disseminate queries to the respective nodes contained in the rectangle. The individual nodes process the queries and transmit the results back to the BS where it is recompiled to answer individual queries. This scheme eliminates redundancy in query and result transmission, and significantly increases the query efficiency.

We observe certain limitations in the existing tree construction, data compression and query processing schemes for WSNs discussed in this chapter. We present the details in Section 2.4.

2.4 Limitations of Existing Approaches

2.4.1 Limitations of Existing Tree Construction and Maintenance Schemes

The limitations of the existing data routing tree construction and maintenance schemes viz., ETC algorithm [40], dynamic route discovery [46] etc., are discussed in this section.

The ETC algorithm computes the *branching factor* (ß), only once during the initial tree construction. During the workload balancing, each node is asked to maintain the size of its sub-tree as recommended by branching factor. After this balancing, some of the children get accommodated with new parents. Sometimes a child node looking for an alternate parent can choose some AP, which is at a higher depth than the self, which results in increase in the number of levels (height) of the tree. This increases the hop-count from the node (which has chosen a new parent) to the BS. Thus, change in depth causes branching factor to change dynamically in the balancing phase which is not taken care of by the ETC algorithm.

Another drawback in ETC approach is that, while reorganizing the tree, if a node's branching factor exceeds the *threshold* (ß) , some of the children will be attached to a new parent (picked from the APL) whose branching factor is less than the *threshold*. Unfortunately, this algorithm will not choose the parent with minimal branching factor; instead it just selects the first possible node that satisfies the required conditions in terms of branching factor, from the APL as its parent. As a result, in the reorganized tree, some nodes are overburdened while some are under loaded. This situation leads to early termination of some nodes due to overload, resulting in reduction of the effectiveness of WSN in terms of its lifetime and coverage.

The drawbacks of the existing dynamic route discovery scheme are- (i) while selecting an alternate parent, the existing workload of the alternate parent is not considered which may again lead to overloading the new parent, and (ii) this approach is not capable of predicting the failure of a node (due to power outage) in advance.

2.4.2 Shortcomings in Existing Data Compression Schemes

The main drawback of K-RLE [49] is that the compression ratios depend on the data sources. The user chooses the *K* value depending on the compression ratio desired. K-RLE can achieve higher compression ratios at the cost of data precision when *K* increases. Thus, the value of *K* provides an indication about the data loss resulting from the process. The experimental results for K-RLE show that for *K* equal to 2, the compression ratio achieved is 40% more than that of the RLE algorithm, and the data loss seen is 50%. Hence, in K-RLE we can see that compression ratios fall down as the precision requirements are high. This limits K-RLE from adoption by applications which require high precision. Hence, there is an obvious requirement for a better lossy compression algorithm in WSN, which can result in higher compression ratio with minimal loss.

2.4.3 Drawbacks in the Existing Query Processing Schemes

The drawbacks in R-Tree [66] approach are as follows. While constructing R-Tree, the containment among the queries is identified by looking at the geographical co-ordinates specified in individual queries. Hence this containment relationship does not take into account the conditions specified in the *WHERE* clause of the query i.e., filters applied on other parameters like time etc. This is the major drawback of the R-

Tree algorithm. As we observe that in most of the wireless sensor applications data retrieval mainly based on the conditions specified in *WHERE* clause with respect to time for temporal data readings for various physical properties, we believe that if the containment relationship is identified based on the sensing time of the readings, it is more likely to increase the effectiveness.

2.5 Overview of Proposals made in the Thesis

To address the issues detailed in Section 2.4, in this thesis we make the following proposals.

1. We propose *Workload-Aware Tree Construction* (WATC) scheme, for efficient workload balancing among the nodes in the tree. Our WATC scheme computes the workload of every node based on the number of child nodes. As the nodes are arranged level-wise, the average workload associated with each level is computed to compare it with workload of the individual nodes at that level. If a node is found to possess workload more than that of the average workload, then some of its child nodes can be attached to other nodes with the least workload at the same level. This way, a tree is balanced with respect to workload, having the load distributed uniformly among the nodes at a given level.

2. To address the issue of recovery from node failures, we propose *Workload-Aware Path Repairing* (WAPR) scheme, to reconnect the disconnected sub-tree (which is a result of an intermediate node failure due to power outage) to the main network. In this scheme, we try to optimize the communication cost by applying tree reorganization technique that minimizes the number of hops from the disconnected sub-tree to the BS and also balance the workload among the intermediate nodes while choosing an alternative path for the disconnected sub-tree.

3. We also propose *Dual Tree Data Routing* (DTDR) scheme for increasing the lifetime of the network. According to this scheme the network will have two base-stations (located opposite ends of the network) and two routing trees with one base-station per tree. Each routing tree is built by distributing the workload uniformly

among the nodes in the tree, using our novel WATC scheme mentioned above. The network switches between these two trees at specified intervals for its operations. According to this the role played by a node in one spell is reversed in the succeeding spell with respect to the level/depth of the node in the tree. This DTDR scheme facilitates uniform distribution of load across the nodes, leading to longer network life.

All the proposals mentioned above are specifically meant for WSNs that adopt push-based data acquisition model. The details of these proposals are elaborated in Chapter 3 of the thesis.

4. Keeping the drawbacks of the existing lossy data compression schemes in view, we propose a novel lossy data compression algorithm called *Induced Redundancy based Lossy Data Compression Algorithm* (IR-LDCA) to achieve better compression ratios at minimal precision loss. The detailed description of this scheme is presented in Chapter 4.

5. We propose a novel query processing scheme for WSNs to improve the energy efficiency, by exploiting the cached results at BS and the containment relationship among the queries submitted at the BS. According to this, we formulate super queries that contain the results of all the queries related by containment. We explain this scheme in-detail in Chapter 5.

2.6 Summary

In this chapter, we reported the highlights of our literature survey done on the existing tree construction and maintenance, data compression, and query processing schemes for WSNs. The major drawback in the existing tree construction and maintenance schemes is that the load distribution among the nodes is not reasonable and effective. Same is the case with re-organization of the routing tree while recovering from a node failure. Next, when we look at the existing lossy data compression schemes applied to data sets with considerable amount of correlation, they are not so effective in terms of achieving higher precision with minimal loss.

We also observe that the existing query processing schemes that exploit containment relationship among the queries do not take the conditions set on time, in the *WHERE* clause.

The above mentioned shortcomings motivated us to investigate novel approaches for above mentioned activities. We propose-

(i) a novel *Workload-Aware Tree Construction* (WATC) scheme, for efficient workload balancing among nodes in a tree.

(ii) a novel *Workload-Aware Path Repairing* (WAPR) scheme for effective path repairing to recover from node failure situations.

(iii) a novel *Dual Tree Data Routing* (DTDR) scheme to facilitate energy efficient operation of the network by uniform distribution of load across the nodes leading to longer network life.

(iv) a novel lossy data compression scheme called *Induced Redundancy based Lossy Data Compression Algorithm* (IR-LDCA), to reduce the data transmission costs and energy consumption.

(v) a novel query processing scheme which exploits the cached results at BS and the containment relationship among the queries submitted at the BS.

All the proposed approaches result in energy efficiency in the network and longer network lifetime. The above mentioned proposals are elaborated in the following chapters in the same order.

Chapter-3
Novel Approaches to Construction and Maintenance of Routing Trees in WSNs

In this chapter we present our novel approaches to construction and maintenance of routing trees to optimize the power utilization in WSNs that adopt push-based data acquisition model. Our schemes address the drawbacks of the existing routing schemes that adopt tree structures for data routing.

3.1 Introduction

In WSNs, routing trees play a major role in forming routing paths for data transmission. Since the nodes are organized in hierarchical pattern, nodes that are part of the higher levels tend to expend more energy due to increased loads when compared to the nodes at relatively lower levels of the tree. This leads to lot of imbalance in the workload handled by nodes. This is a major concern w.r.t., achieving increased lifetime of the network. There exist a reasonable number of schemes to construct routing trees that are load-balanced [34-39]. Further, even after construction of a tree which is reasonably balanced w.r.t., load, it is very common to experience node failures due to various reasons and recovering from such node failure to bring the disconnected parts of the tree linked to the network is another major challenge. A good number of schemes [60, 61] have been proposed to achieve this. Still we observe that the above schemes used for tree construction and recovery suffer certain drawbacks as detailed in Section 2.4.1. We propose efficient tree construction and maintenance schemes for push-based data acquisition in WSNs. Our proposed schemes are intended to overcome the drawbacks of the existing schemes.

First we present our proposed *Workload-Aware Tree Construction* (WATC) scheme [68], in which the tree construction takes place with even distribution of workload among the nodes at each level. We compare the performance of this approach with the existing ETC algorithm [40] to validate our ideas. Next, we discuss our proposed *Workload-Aware Path Repairing* (WAPR) scheme [69], to recovery from node failure in data routing trees. We also present our experimentation results that prove the efficacy of our technique over the existing ones. Next, we elaborate on our proposed *Dual Tree Data Routing* (DTDR) scheme [67], in which the network will have two base-stations (usually located at opposite ends of the network) and two routing trees. The network operates by switching between these two BSs, where the switching is triggered by events that occur on predefined relative power levels in the network. In our proposed DTDR scheme, load taken by each node in the network is balanced, as the nodes change their roles alternatively w.r.t., the sub-trees they support. This load balancing would prolong the network lifetime. We also present the overall functioning of the proposed dual tree scheme with necessary experimental proofs regarding its improved performance. All required experiments are conducted on our custom-built simulator.

Our proposed tree construction and maintenance schemes are based on the assumptions that the network adopts push-based data acquisition model where each node continuously senses and sends its data to the BS through the ancestors lying on the path to the BS.

3.2 A Novel Energy Efficient Workload-Aware Tree Construction Scheme

In this section, we present the working theory behind our proposed *Workload-Aware Tree Construction* (WATC) scheme [68]. At the end of this section we also present the experimental results for the proof of validation of our ideas.

3.2.1 Workload-Aware Tree Construction (WATC) Scheme

Now, we give a detailed description of the proposed WATC scheme. Our tree construction process is explained in following steps.

i). Construction of initial tree using FHF approach as described in ETC algorithm.

ii). Computing the workload of nodes and load balancing.

FHF Tree Construction

Now, we explain the process of constructing FHF tree with suitable illustration. Initially, the BS transmits a '*hello*' message to the nodes within its transmission range. Each of the receiving nodes acknowledges the receipt of '*hello*' message from the BS, with an '*ack*' message and becomes a child node of BS. We assume that the BS is at level 0, and these set of child nodes, identified as above, will be at level 1. Now, each node at level 1 sends '*hello*' message to the neighboring nodes. In this process, a node may receive '*hello*' messages from more than one node. Now each node identifies the node from where it has first heard the '*hello*' message from and confirms it to be its parent. All other nodes from where it has received '*hello*' messages will become *Alternate Parents* (APs).

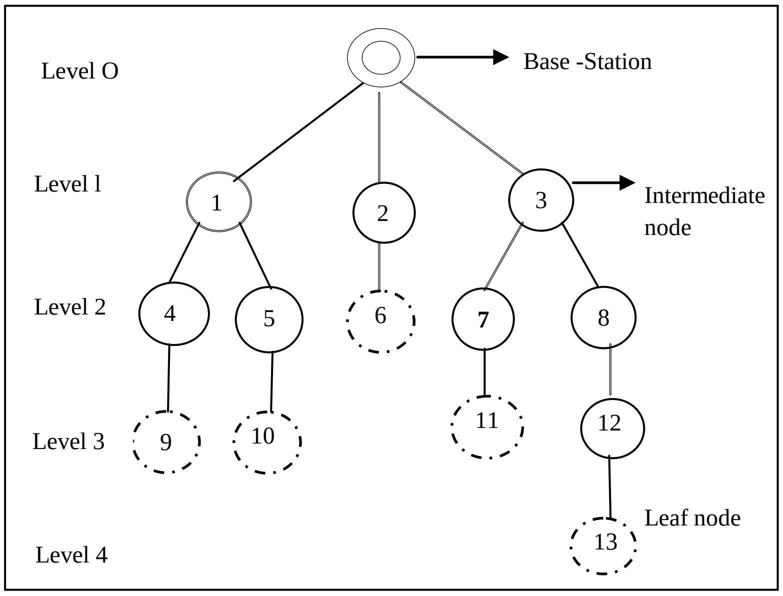


Figure 3.1 Tree topology for WSN

Every node maintains the list of its APs in *Alternate Parent List* (APL). Also, each parent maintains the list of its child nodes in *Child Node List* (CNL). Thus, the level 2 is established for the tree. This process is repeated to establish all the levels till the complete tree is formed as shown in Figure 3.1. In this process, every time a child parent relationship is established between two nodes, the same is informed to the BS where the global information about the tree is maintained. We also assume that the APL and CNL of each node are also available at the BS. Each node can listen to nodes within its transmission range. When a node becomes a child of its first requested parent node, the rest of the nodes which are able to transmit to this node become its APs. A node maintains all the incoming request messages in APL, which is used during node failure and alternate parent assignment during the tree balancing phase.

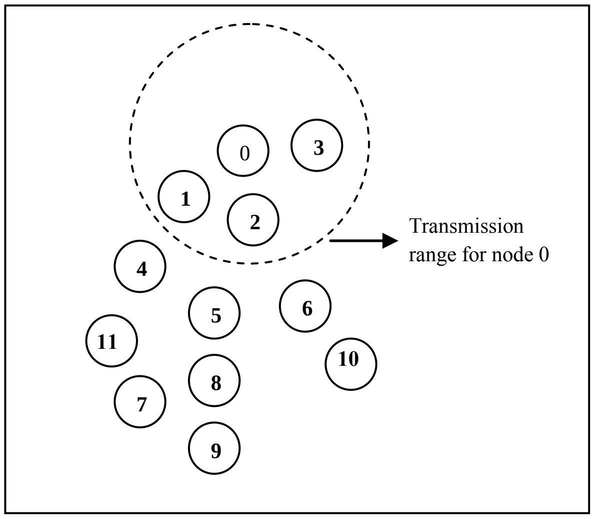


Figure 3.2 Example node deployment

To facilitate our description, consider a set of 11 nodes deployed as shown in Figure 3.2. First, we explain how an initial tree is constructed by FHF approach as given in ETC with the following illustration in Figure 3.2. The BS (node 0), sends a '*hello*' message to all its neighbors (nodes within transmission range) i.e., nodes 1, 2, 3. Each of these child nodes reply with '*ack*' message back to the parent node (node 0). This is to confirm the parent-child relationship. Further, nodes 1, 2 and 3 send '*hello*' message to other nodes within their transmission range (except to their parents). This process continues till it reaches the leaf nodes of the network. The initial tree structure formed is shown in Figure 3.3. Since a push-based continuous data acquisition model is assumed, each node in the network performs sensing and attempts to transmit the sensed data (in the form of packets) to the BS in a continuous manner. The network normally works as long as every intermediate node is alive. Hence, the network lifetime is highly dependent on the battery life of its nodes. If we analyze the energy consumption rate of every node in the network, we find that the intermediate nodes are being taxed more when compared to leaf nodes of the tree. In our example, we may note that the intermediate nodes of the tree at level 1 (node 1) and at level 2 (nodes 4, 5) not only transmit self-data, but also take the responsibility of forwarding the data produced by their descendants. The parent-child relationship among the nodes is also depicted in Figure 3.3.

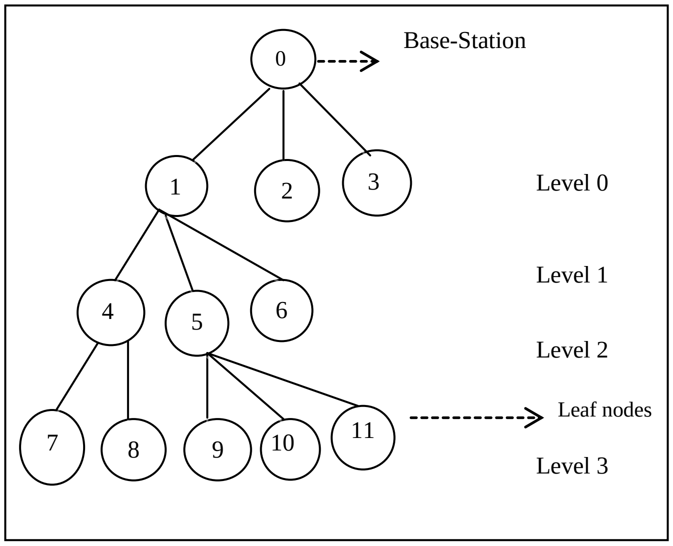


Figure 3.3 Example FHF tree structure of WSN

With this FHF approach, it is clear that the tree construction is ad-hoc in nature, leading to uneven workload distribution among the sensor nodes at a given level. Hence, the need arises for an optimal tree construction scheme, which can distribute the workload at each level uniformly**.**

Computing the Workload and Load Balancing

In the next step, we compute the workload of each node in the tree to calculate the average workload at each level to perform load balancing as described in our work [68]. Now, we elaborate our load balancing approach. The workload of a node is the sum of its self-load and the workload due to its children (sub-tree). The BS computes the workload of each node in the network and stores in the data structure that holds the other information about the nodes of the tree. We assume that the self-load of each node is 1 unit of work. The workload of a node is computed as given in Procedure 3.1. The Table 3.1 gives the computed workload for each node of the WSN given in Figure 3.3.

Next, the BS computes the Average Workload (AWL) for each level, as given in Algorithm 3.1, and stores in the data structure (table of data) that holds the complete information about the routing tree, which is available at the BS. The table contains the information about parent, APL, CNL, workload for each node, average workload for each level, etc.

Procedure: 3.1: ***Workload (node)***

This procedure computes the workload of a node and returns the same. Here, *getChildNodeList(node)* is to extract the list of child nodes of a given node.

```
1.  Set $WL := 1$
2.  Set $CL := getChildNodeList(node)$
3.  Set $size := |CL|$                        //size of the child list
4.  If ($size$ !=0)
      $WL := \sum_{i=1}^{size} \textbf{Workload}(CL[i-1])+1$        //recursive call
5.  End if
6.  Return $WL$
```

The workload balancing for each level is performed at the BS, as per the Algorithm 3.2. All the data that is necessary to perform this load balancing is available at the BS. In our scheme, we define the *threshold* value (λ) for the maximum amount of workload for any node at a given level, as 1.5 times the average workload of that level. This *threshold* (λ) is different from the *threshold* (ß) used in the ETC algorithm. If the workload of a node is greater than the *threshold* value given for that level, then it is considered as the *candidate node* for load balancing. The candidate node will then refer to its CNL and select one child node with highest workload for relocation.

Algorithm: 3.1: *AverageWorkload(level)*

This algorithm computes the average workload of a given level. We use NODES as array of nodes at that level. The method *getNodes(level)* gives the list of nodes at that level. AWL is the average workload, and *getWorkload(node)* returns the computed and stored value of workload of a given node.

```
1. Set NODES:=getNodes(level)
2. Set k:=| NODES |
3. AWL:= $\left(\sum_{i=1}^{k} getWorkload\ (NODES[i])\right) / k$
4. Return AWL.
```

Now this selected node need to be connected to a new parent. In order to select a new parent for this child node (identified for relocation), its APL needs to be examined. One of the nodes in the APL, will be selected to become its new parent, if it satisfies the below mentioned criteria.

- The node should have least workload among all APs in the APL.
- The sum of the existing workload of the new parent and the workload of the relocated child should be less than the *threshold* (λ) of the level to which the new parent belongs.

Once the new parent is found, the newly established parent-child relationship is captured into the table at BS by appropriate updates to the data structure. In case, if a node doesn't find a new parent satisfying both the above criteria, it may be due to selection of dependent node (of candidate node) with highest workload. As an alternative, the dependent node with second highest workload will become a candidate for relocation, and looks for the new parent as per the above mentioned criteria. This process of identifying a dependent of the candidate node, for relocation will repeat until the workload of the candidate node goes below threshold.

Algorithm 3.2: Workload Balancing(*node*)

This algorithm balances the workload of the *node.*

Here we use *getChildNodeList()* to extract the list of child nodes of a given node, *getAverageWorkload(level)* to extract the average workload at a given level, *getWorkload(node)* to extract the stored value of workload of a given node, *getAlternateParentList(node)* extracts the alternate parents of the given node , *getNode(workload)* extracts the node id of the node with the given workload, *addChild(m,n)* adds a node *n* to the Child Node List of node *m*, *getLevel(node)* extracts the level of a given node.

```
1.  if(getWorkload(node) > 1.5* getAverageWorkload(getLevel(node))
2.  CNL[]:=getChildNodeList(node);
3.  For each j:=1 to |CNL|
4.  WL[j]:= getWorkload(CNL[j])
5.  End for
6.  APL[]:= getAlternateParentList(bestchild)
7.  For each k:=1 to |APL|
8.  W[k]:=getWorkload(APL[k])
9.  End for
10. bestnewparent:= getNode(getLeast(W[]))
11. if( getWorkload(bestchild)+getWorkload(best parent)< 1.5*
    getAverageWorkload(getLevel(bestnewparent)))
12. addChild(bestnewparent,bestchild)
13. End if
14. End if
15. End Procedure
```

In some rare cases, we may not find a suitable new parent. In such case the tree remains unchanged. This way, we balance the tree by distributing the workload uniformly among the nodes at same level. Thus, our algorithm ensures uniformity in workload distribution in the network, thereby increasing efficiency and lifetime. The complete algorithm is presented in Algorithm 3.2. The BS will apply this algorithm to all the nodes of the tree starting from the top of the tree.

Now, we illustrate our WATC load balancing approach with the help of the initial tree given in Figure 3.3, which is constructed using FHF approach. We know that every node has a parent node and a list of alternate parents. All nodes maintain their CNL as well as APL. As mentioned earlier, all this information about each node of the tree is available at the BS. For instance, at node 4, the child nodes are 7 and 8. Therefore, the workload associated with node 4 is the sum of the workloads of node 7, and 8 and the workload of node 4, itself. This is computed using the method given in Algorithm 3.1. The computed workloads and other relevant details for all the nodes of the tree in Figure 3.3 are presented in Table 3.1.

Next, the Average Workload (AWL) of each level of the tree is computed from top to bottom by considering the workload of all the nodes at respective levels. The average workload is computed as- the sum of workloads of all the nodes divided by the no. of nodes at that level. The procedure for computing the average workload is given in Algorithm 3.1. Now, for illustration, we describe the computations involved in deriving the average workload, and the *threshold* (λ) for each level.

For level 1, the no. of nodes is 3 (node 2, 3, and 4), and the sum of the workloads of all its dependents is 5 (1+1+3). Now, the average workload for level 1 is 1.6. Similarly, for level 2 it is 2.6. Since the level 3 contains only leaf nodes we need not compute the average workload. For a given level, we define the *threshold (λ)* as 1.5 times the average workload of that level (which is the result of observations made during the simulations).

Nodes 1, 2, and 3 at level 1 need to maintain 2.4 as the maximum workload. Similarly, nodes at level 2 are to maintain 3.9 as their maximum workload. If the workload of a particular node is more than *threshold (λ),* then, we identify such node as the candidate node for applying load balancing.

Table 3.1. Details of the tree structure for the network given in Figure 3.3.

Node No.	**Level No.**	**Parent Node**	**No. Child**	**Child Node List (CNL)**	**Workload**	**Alternate Parent List (APL)**	**Average Workload (AWL)**
0	0	-	3	1,2,3	12	-	-
1	1	0	3	4,5,6	9	2,3,4,5,6	1.6
2	1	0	0	None	1	1,3,5,6	1.6
3	1	0	0	None	1	1,2,6	1.6
4	2	1	2	7,8	3	5,6,7,8,11	2.6
5	2	1	3	9,10,11	4	2,4,6,7,8,9,10,11	2.6
6	2	1	0	None	1	2,3,4,5,7,8,10,11	2.6
7	3	4	0	None	1	5,6,8,9,10,11	1
8	3	4	0	None	1	5,6,7,9,10,11	1
9	3	5	0	None	1	7,8,10,11	1
10	3	5	0	None	1	6,7,8,9	1
11	3	5	0	None	1	4,6,7,8,9	1

Now, some dependent node of the candidate node needs to be attached to a new parent. The candidate node selects the dependent with the highest workload to be the suitable child node for reorganization. This child node is to be detached from the candidate node. To reduce the workload of the candidate node, we attach this child to a new parent. In our example, we observe that the workload of node 1 at level 1 exceeds the threshold; hence, node 1 becomes the candidate node for applying load balancing. For node 1, we find that the node 5 is the suitable node for relocation. This node 5 has an APL with nodes-{2, 4, 6, 7, 8, 9, 10 and 11}. Now, the node 5 looks for a node from the APL, which satisfies the criteria mentioned earlier in the algorithm. Among the nodes in APL, the node with least workload becomes the suitable node to become the new parent of node 5. Further, this addition of node 5 should not increase the workload of its new parent beyond its *threshold*. In case if the workload of the new parent exceeds the *threshold*, then select the next suitable node from the APL, and check the above conditions for this new choice as well. Repeat this until the parent is found. In our example, for node 5 it is evident that node 2 becomes the new parent. We also observe that the new workload of node 2 is less than the *threshold*. The above technique is applied repeatedly to all the levels. The Figure 3.4 gives us the complete details about the reorganized tree. According to this, the intermediate nodes {1, 4, and 5} will now have uniform workload. The nodes {2, 3, and 6} which were under-loaded before reorganization are now given some workload to maintain uniformity in workload allocation. We discuss the performance of our WATC approach in the following section.

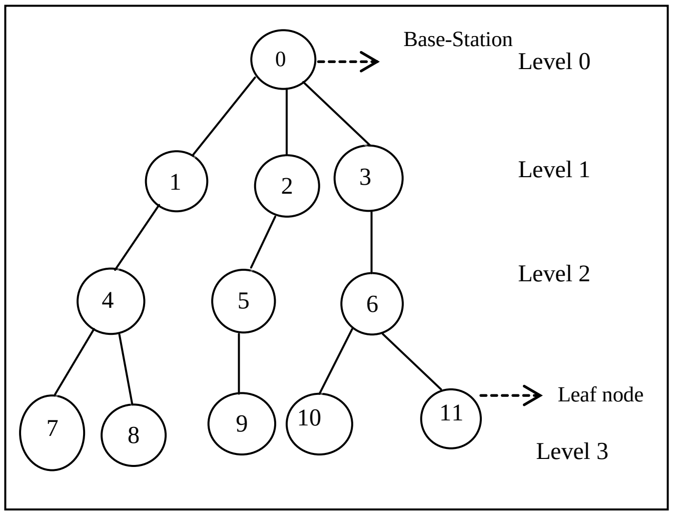


Figure 3.4 Final workload-aware tree

3.2.2 Performance Evaluation

In this section, we present a brief report on the series of experiments we have conducted on our custom-built simulator to assess the effectiveness of our WATC approach discussed in the previous section, and compare the same against the existing ETC algorithm.

Simulation and Experimental Set-up

We have developed a custom-built simulator, which is implemented using Java Technology. Our simulator is meant for Windows platform and is console-based. This simulator allows us to define the geographical range of the network along with the number of nodes. We can also define the transmission range of each node. Since we plan to conduct experiments to assess the power consumption of our technique, we have also designed our simulator wherein the power allotted to each node can also be defined. The deployment of nodes using our simulator can be either random or predefined. For the simulation purpose we assume the connectivity to be constant.

The simulator was run for networks of sizes 20, 50, and 70 nodes. The input to the simulator is the location of nodes specified by their *x* & *y* co-ordinates. The radio communication range of each node is set as 3m. Each sensor node is initialized with 1J of energy. To simulate a push-based data acquisition model, the sensor nodes send the sensed data to the base-station continuously so that the energy of the nodes gets depleted. We just focus on simulating the communication load due to transmission. Further detailing about the simulator is given in Appendix-1.

Simulation Results

Now, we present the performance of our scheme, along with their result analysis. A set of experiments were conducted to compare the performance of our WATC scheme against that of the technique described in ETC. The comparison is made based on the following metrics.

1. The number of packets transmitted during the lifetime of the network.
2. The lifetime of the network.
3. The residual power available in the network after the network comes to halt.

We have done our experimentation with trees of 20, 50, and 70 nodes. In each case, we simulated five different topologies. The presented reading for each performance metric is the average of the five simulation runs conducted on above trees. We have computed- (i) number of packets delivered, (ii) network lifetime, and (iii) residual power in the network for each simulation. The average values of our results for each experiment are depicted in the graphs presented in this section. According to Figure 3.5, we observe that our WATC algorithm shows a good increase in the number of packets transmitted, when compared to the ETC algorithm. From the results it is evident that as the number of nodes in a network increases, the number of packets delivered in the network also increases.

Figure 3.5 Graph depicting the total number of packets transferred till the network dies for a network of 20, 50, and 70 nodes.

The graph in Figure 3.6 shows the network lifetime which we consider as a significant parameter to quantify the effectiveness of the network. The network lifetime is the time elapsed between starting of the network, and the moment it halts. A network is considered to have reached a halt state when not even a single node is connected to the BS. Otherwise, we say that the network is alive if at least one node in the network is active and connected to the BS. The graph shows excellent results associated with our approach when applied on network with 20, 50, and 70 nodes. A significant increase in the network lifetime can be seen in network with 70 nodes. This shows that the workload-aware algorithm increases the longevity of WSN as the network size increases. The longer the lifetime, higher is the energy efficiency of the network.

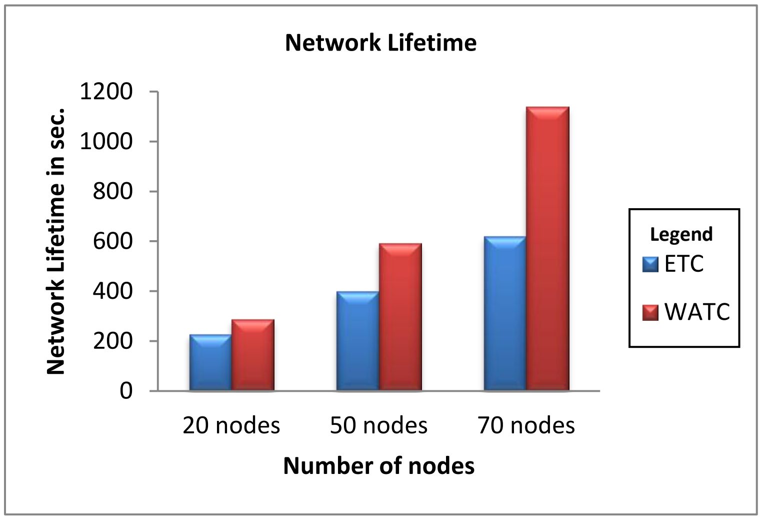


Figure 3.6 Graph depicting the network lifetime till the network dies for a network of 20, 50, and 70 nodes.

As the network spends the maximum amount of energy for packet transmission, battery power is reduced at each of these nodes after each transmission. Optimizing the battery power utilization is one of the major issues in WSN. We measure the power utilization of WSN by considering the residual power (power left) of each node after the network dies. The following graph as shown in Figure 3.7 shows the total residual power in the network. In the case of ETC algorithm, when the intermediate nodes die quickly, the network will come to a halt early. It is found that a huge amount of battery power of nodes in WSN is left unutilized. Our algorithm has less residual power when compared to the ETC algorithm. More residual power in case of ETC is due to early termination of intermediate nodes.

Next, the Figure 3.8 shows the number of nodes alive in the network after transmitting 40 packets in three networks with 20, 50, and 70 nodes. The graph clearly shows the difference between ETC and our WATC algorithm. The number of nodes alive after transmitting 40 packets is less in case of ETC because of the quick termination of intermediate nodes and its children. In case of our WATC algorithm, we see that more number of nodes are alive than the ETC algorithm which implies increase in lifetime of the WSN, proving that our WATC algorithm is more effective.

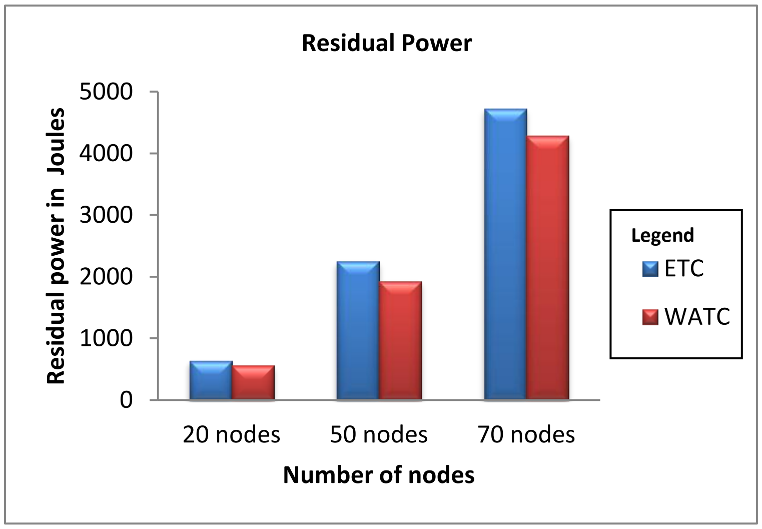


Figure 3.7 Graph depicting the total residual power of all nodes in the network of 20, 50, and 70 nodes.

From the above experimental results, it is evident that we have successfully achieved the goal of optimal power utilization at intermediate nodes, using our WATC algorithm. We have showed that intermediate nodes can live for a longer time when reorganization of the initial tree is done by considering the workload at different levels. Hence, we conclude that WATC algorithm is successful in optimizing the energy consumption and increasing effectiveness of the network through its load balancing strategy. Various comparisons have been provided to show how our scheme works better than ETC algorithm in terms of lifetime, throughput, residual power, and power utilization.

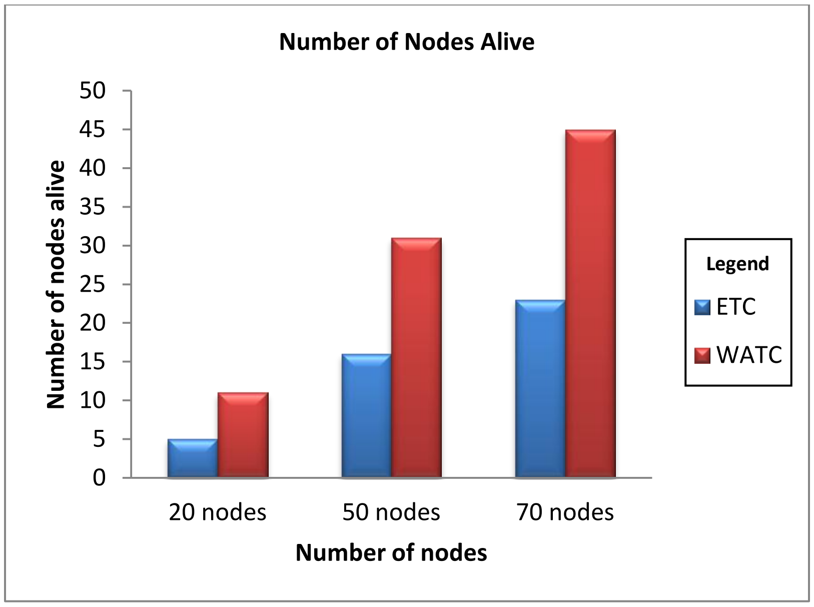


Figure 3.8 Graph depicting the total number of nodes alive after 40 packets transferred in the network of 20, 50, and 70 nodes.

3.3 A Novel Approach to Recover from Node Failures

In this section, we explain our proposed *Workload-Aware Path Repairing* (WAPR) scheme [69] to recover from node failures in routing trees constructed using our WATC approach described in Section 3.2. Though, the node failures can occur due to reasons like hardware/software failure, natural calamities (like flood, earthquake, fire etc.), power outage etc., we focus on failures due to power outage (zero battery power) at sensor nodes. The failure of an intermediate node, results in disconnection of its entire sub-tree from the WSN. Therefore, as a result of a node failure in a WSN, it loses services of the portion (sub-tree) of the network for which the failed node is the root. We call such portion as *disconnected sub-tree*, as shown in Figure 3.9. As a result of this, the residual battery power of the entire sub-tree remains unutilized. This reduces the effectiveness of the WSN. When a node fails (due to zero battery power), the dependent nodes lose their parent. The routing tree gets disturbed. Hence, the dependent sub-tree is totally cut-off from the network. Many applications come to a halt due to such node failures. This issue is addressed with our WAPR scheme explained in this section.

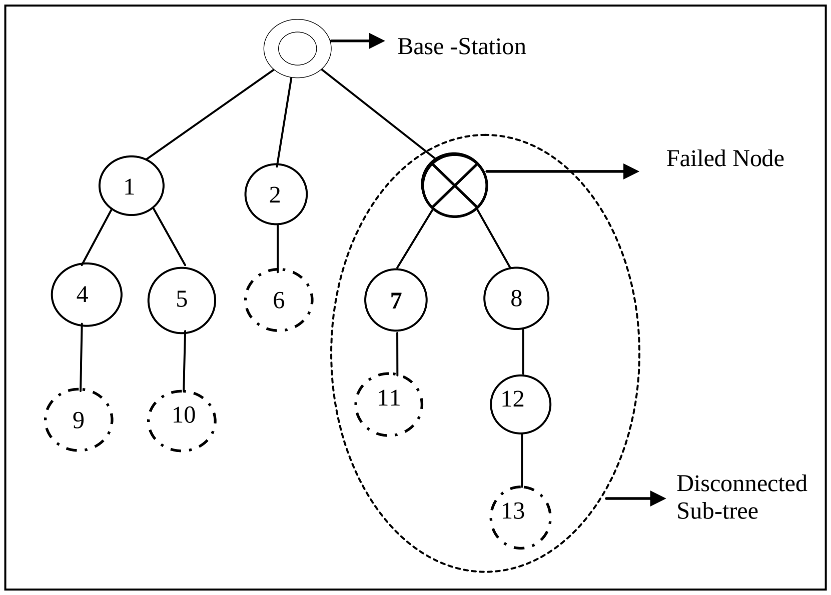


Figure 3.9. Disconnection of a portion of the tree due to node failure.

Workload-Aware Path Repairing Scheme

The main objective of our *Workload-Aware Path Repairing* (WAPR) scheme is to reconnect the disconnected sub-trees (which are result of intermediate node failures due to power outage) to the main network. In this scheme, we try to optimize the communication cost by applying tree reorganization technique that minimizes the number of hops from the disconnected sub-tree to the BS, and also balance the workload among the intermediate nodes while choosing an alternative path for the disconnected sub-tree. This reduces collisions and further early node failures in the reorganized network. Hence, our proposed path repairing scheme is workload-aware.

Now, we illustrate the working of the WAPR scheme, for a WSN with 14 nodes deployed as shown in Figure 3.10. First we construct the initial tree using FHF approach and apply our WATC scheme to obtain the workload balanced tree. The tree structure formed is shown in Figure 3.11, and adopts our novel WAPR scheme for handling node failure situations.

The network functions effectively, when majority of the nodes are alive and connected. Hence, the network lifetime is highly dependent on the battery life. If we analyze the energy consumption rate of every node in the network, we find that the intermediate nodes at level ‘x’ are taxed more compared to intermediate nodes at level ‘x+1’. This leads to early shutdown of intermediate nodes. This node failure will disconnect its sub-tree from the network.

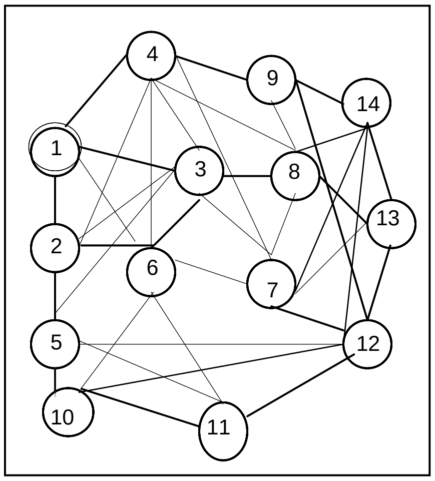


Figure 3.10 Example node deployments in a WSN

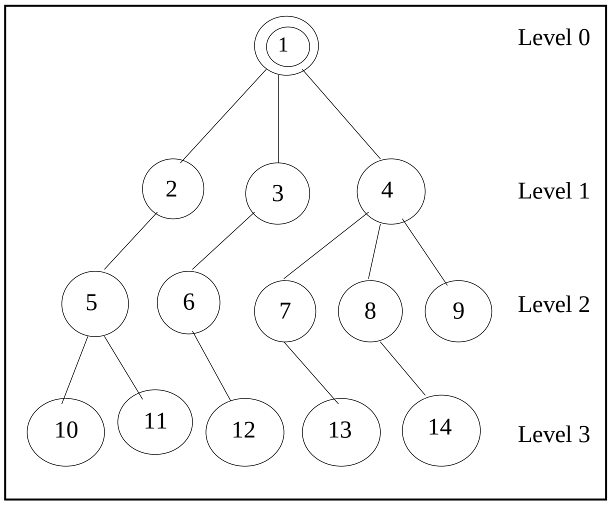


Figure 3.11 Example tree structure of WSN.

We now give details about our workload-aware path repairing process.

i. **Failure Detection:** Each node identifies itself before it fails due to power outage, and instructs its children to find a new parent.

ii. **Path Repairing:** Identifying a new parent for each of the child nodes of the failing node.

Algorithm 3.3: *FailureDetection(node)*

This algorithm is used by a node to identify the self before it fails and instruct its children to choose a new path to BS. Here, *getPower(node)* extracts the current power level of the node. Threshold is the constant 5 % power level of a node, *getChildNodeList(node)* is to extract the list of child nodes of a given node.

```
1. If(getPower(node) < threshold)
2. Set CNL:=getChildNodeList(node)
3. For each node in CNL
4. FindNewParent(node)
5. End for
6. End if
```

A. Failure Detection

Every node continuously monitors its power consumption and availability. The Algorithm 3.3 is run by every node to achieve this. As soon as the power level of a node reaches the threshold level (when the node is left with 5% of the initial power), it senses the impending failure and informs its child nodes to look for new parent to reach the BS.

B. Path Repairing

This process is initiated at all the nodes, which have received an instruction from the parent which is about to fail, to look for a new parent. Each such child chooses a new parent from its APL. Our proposed *workload-aware path repairing algorithm* is given in Algorithm 3.3. According to this, each node looking for a new parent sends an appropriate request message to all its APs, to send their respective workload and hop-count back. Then each alternate parent after receiving the above message will respond to the same by sending the requested data.

Now, the child node looking for a new parent selects the most eligible alternate parent from its APL, as per the following criteria and makes it as its new parent.

a) The workload of the new parent should be least among the nodes in the APL.

b) The hop-count to reach the BS from the new parent should be minimal.

The above criteria will enable the child to select the best alternate parent (with least workload and hop-count) as its new parent.

To illustrate the working of our WAPR algorithm, we consider the example tree in Figure 3.11, which is a workload-aware tree. Let us assume that, the node 6 has reached its threshold, (i.e., the node has only 5% of the initial battery power and is going to shut down within an hour (approximately)). Now, the node 6 sends a '*good-bye*' message to all its children (node 12). This message implies that the child should look for a new parent which can be chosen from the APL of the child.

At that point, each child node invokes the functionality as per the Algorithm 3.4. According to this, node 12 sends messages to all its APs (5, 7, 9, 10, 11, 13, and 14) to send back the workload and hop-count. Now all the APs of node 12 will respond to this.

The S*uitability Factor* (SF) indicates the degree of suitability of the given APL to become the new parent. Higher the SF more will be the suitability. Now, for each node in the APL, final suitability factor will be computed taking into account both workloads (40% weightage) and hop-count (60% weightage). According to this, if in the APL, the minimum workload of a node is 4, and minimum hop-count is 3; then an AP with workload 6, and hop-count of 4 will get credits for workload as 4/6, and credits for hop-count as 3/4.

The suitability factor is computed as follows:

Suitability Factor = (credits for WL*0.8) + (credits for level* 1.2)

= (4/6 *0.8) + (3/4 *1.2)

= 1.4

After computing the SF for all nodes in APL, the node looking for a new parent will choose the node in APL with highest SF as the best new parent. The SF for all the 14 nodes of Figure 3.11 is computed and presented in Table 3.2. This facilitates choosing a new path with minimal workload and hop-count. Hence, such path will be an optimal one.

Algorithm 3.4: ***FindNewParent(node)***

This algorithm will return an alternate path for the child nodes of CNL to reach the BS. Here, *isAlive(node)* returns true if the node is alive, *getNodeId(list)* extracts the first node ID from the list, *getWorkload(node)* to extract the stored value of workload of a given node, *getLevel(node)* extracts the level number of the node, *addChild(m,n)* adds a node *n* to the CNL of node *m*. *getAlternateParentList(node)* extracts the alternate parents list of the given node, *getMinimum(array)* returns the minimum value in the array, *getMaximum(array)* returns the maximum value in the array.

```
1.  If (isAlive(node))
2.  APL[ ]:= getAlternateParentList(node))
3.  For each node in APL[ ]
4.  WL[ ]:=getWorkload(node))
5.  End for
6.  min_wl:=getMinimum(WL[])
7.  min_level:=getMinimum(getNodeId(WL[]))
8.  mwl:=min_wl/getWorkload(node)
9.  mln:= min_Level/getLevel(node)
10. new:=getMaximum((0.8*mwl)+(1.2 *mln))
11. bestnewparent:= getNodeId((new))
12. else
13. bestnewparent:= getNodeId((new))
14. addchild(bestnewparent, node)
15. End if
```

Table 3.2 Details of example WSN in Figure 3.11

Node no.	**Level no.**	**Work load**	**Suitability Factor**	**Best new parent**	**Alternate Parent List (APL)**
1	1	4	-	-	-
2	2	2	1.4	3	3,4
3	2	2	2	2	2,4
4	2	4	2	2	2,3
5	3	3	2	3	3,6
6	3	2	2	7	2,4,7
7	3	2	4.4	8	3,6,8
8	3	2	4.4	9	3,7,9
9	3	1	4.4	8	8
10	4	1	4.4	11	6,11
11	4	1	1.7	12	6,10,12
12	4	1	2	9	5,7,9,10,11,13
13	4	1	1.7	14	6,8,12,14
14	4	1	2	9	7,9,12,13

After this path repairing, the sub-tree of a failing node is still connected to the BS with the best possible path. Since, both workload and hop-count play a significant role, in selecting the new parent, we conducted experiments to decide upon the weightages given to each of the above criteria. Our experiments suggest that weightages of 40% and 60% can be associated with workload and hop-count respectively. This, results in a reorganized tree that is power efficient and workload balanced. The issue of intermediate nodes and their higher power consumption than the leaf nodes is addressed in our DTDR scheme. We present its detailed explanation in Section 3.4.

3.3.1 Performance Evaluation

In order to validate our ideas and compare the performance of our proposed *Workload-Aware Path Repairing* (WAPR) scheme [69] with that of the existing path repairing scheme [46], we use the simulator explained in Section 3.2.2 with minor modifications to facilitate the implementation of our WAPR algorithm. We also present the analysis of results of various experiments conducted to test the effectiveness of our technique. The performance metrics used to evaluate our scheme against the existing path repairing algorithm are- (i) number of packets transmitted, (ii) network lifetime, and (iii) residual power in the network after the network comes to a halt.

During the simulations, sensors are deployed randomly in the field, and the FHF approach was applied to construct the initial routing tree. We have experimented with this tree by giving some workload to nodes so that nodes expend energy in packet transfer. Each node is initialized with 1J of energy initially. To simulate a push-based data acquisition model, the sensor nodes send the sensed data to the base-station continuously. We just focus on simulating the communication load due to transmission of result. Further detailing about the simulator is given in Appendix-1.

First we conducted tests by applying path repairing scheme [46], and recorded readings for the performance metrics mentioned above. Then we have run simulations using our proposed WAPR algorithm.

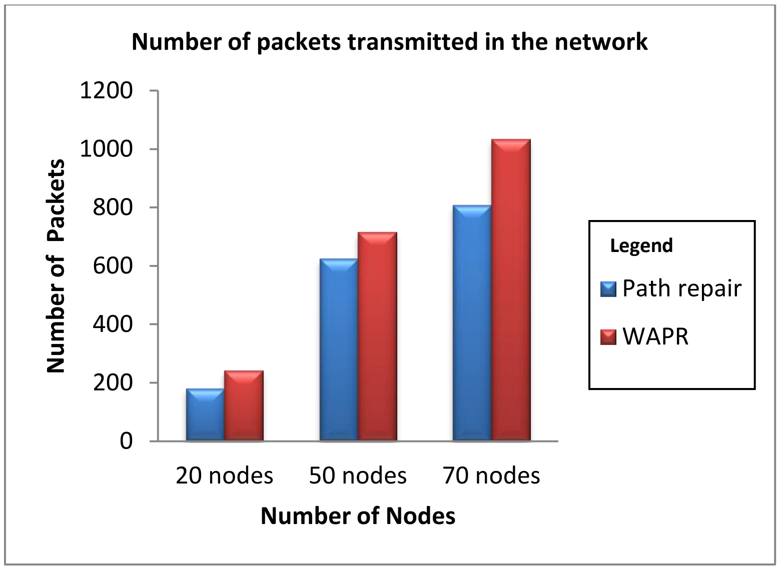


Figure 3.12 Graph depicting the total number of packets transferred till the network dies for a network of 20, 50, and 70 nodes.

Our scheme continuously monitors the power level at each node after every operation performed at that node. As soon as the energy at a node reaches the threshold level of power, the node identifies itself as the one that will shut down soon. On this event, our workload-aware path repairing process will take place at the children of the failing node. This power monitoring and path repairing activities are continuously performed at respective nodes till the network halts. From the results shown in Figure 3.12, it is evident that as the number of nodes in a network increases, the number of packets delivered in the network also increases.

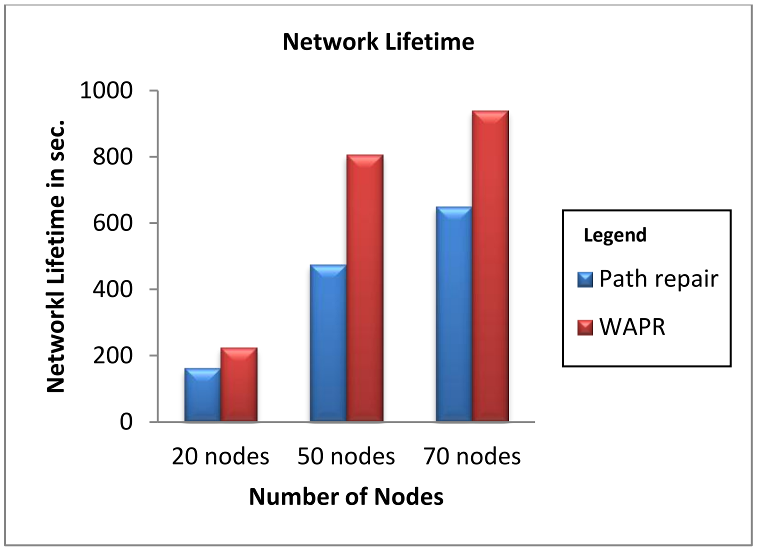


Figure 3.13 Graph depicting the network lifetime for a network of 20, 50, and 70 nodes.

In Figure 3.13, the graph shows excellent results for our WAPR approach w.r.t., the network lifetime. Especially a significant increase in the network lifetime is seen in the network with 70 nodes. This shows that our WAPR scheme increases the longevity of WSN as the network size increases. The longer the lifetime, higher is the energy efficiency of the network.

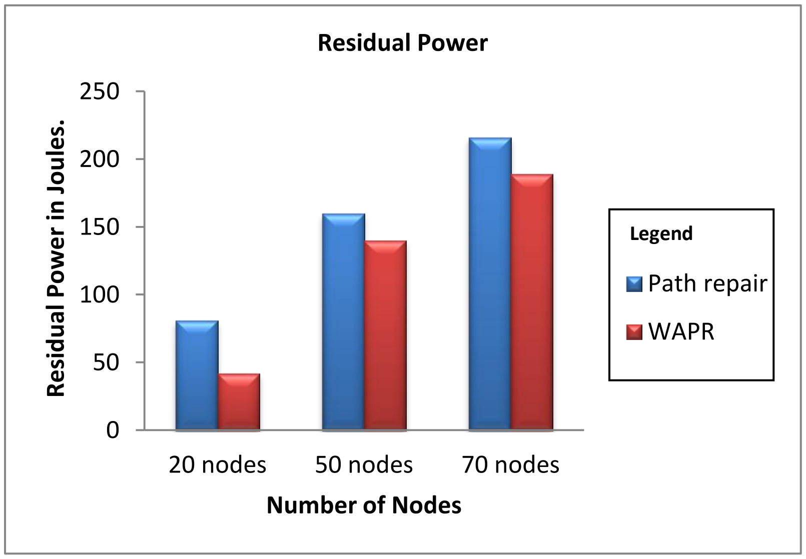


Figure 3.14 Graph depicting the total residual power of all nodes in the network of 20, 50, and 70 nodes.

When a node fails due to zero battery power state, the entire dependent sub-tree also gets disconnected from the network, even though it (sub-tree) has some battery potential. Using our WAPR scheme, we propose an alternate routing path for the dependent sub-tree to stay connected to the network. This shows that the dependent sub-tree expends energy in sending packets to the BS. Thus the residual power left in the network decreases. This means optimum power utilization in the network with the decrease in the residual power (wastage). Optimizing the battery power utilization is one of the major issues in WSN. We measure the power utilization of WSN by considering the residual power (power left) at each node after the halt of the network. The graph in Figure 3.14 shows the total residual power in the network. In case of path repair algorithm, it is found that a good amount of battery power available in the disconnected sub-tree of the WSN is left unutilized. But, our WAPR algorithm has less residual power when compared to the path repair algorithm.

3.4 A Novel Dual Tree Data Routing Scheme

In this section, we give full length description of the construction and functioning of our proposed *Dual Tree Data Routing* (DTDR) scheme [67]. This DTDR scheme is a novel data routing scheme for WSNs where two trees will be used during the lifetime of the network. But at a given point of time, only one tree is operational. Since we have two routing tree structures, we call this scheme as *Dual Tree Data Routing* scheme. We assume that the network has two BSs located at the opposite ends of the topology. Each of the above mentioned routing trees will have its own BS. The network keeps on switching between these two routing trees for communication and data routing. Our DTDR scheme works effectively for WSNs that adopt push-based data acquisition.

Though the name Dual Tree Data Routing (DTDR) sounds similar to that of Dual-Tree-based Data Aggregation (DTDA) [61], the concept of DTDR is entirely different. Our proposed DTDR is meant for data routing and not for data aggregation. Further, in DTDA both trees work simultaneously, whereas in DTDR one tree is operational at a given point of time. Hence there is no overlap in the concept.

3.4.1 Data Routing Tree Construction

In general, the nodes that play the role of a *Base-Station* (BS) are predefined due to the reason that these BSs will have very high computational, battery and storage resources at their disposal. According to the scheme, we assume that there are two such BSs at opposite ends of the topology. First, we construct the *First Routing Tree* (FRT) for the network, with one of the BS as the root. Then we construct the *Second Routing Tree* (SRT) with the other BS as its root and which is one among the leaves of FRT. While constructing the FRT and SRT, we adopt our novel WATC and WAPR techniques for load balancing and path repairing as detailed in sections 3.2, 3.3 respectively. The BS of FRT is called as *First Base-Station* (FBS) and similarly we have *Second Base-Station* (SBS) for the SRT. As per our assumptions we make sure that the SBS is at the far end of the FBS.

Dual Tree operations

The network starts functioning with FBS as its BS and the FRT as the data routing scheme. The FBS collects data continuously from all its child nodes, which forward their own data and the data received from the downstream (children). This network with FRT continues to work until the total residual power of the network becomes equal or close to 50% of the total power in the network that was available when FRT started its operations. The residual power in the network at any given point of time is the remaining power (sum of the power available at all the nodes) at that time.

As soon as the total residual power of the network becomes equal to or close to 50% of the total power in the network that was available when FRT started its operations, the network switches to the SRT. Now, the SBS will start playing the role of BS for the network. This network with SRT continues to work until the total residual power of the network becomes equal or close to 50% of the total power in the network that was available when SRT started its operations in the recent spell.

This process of switching between two data routing trees continues throughout the network lifetime. The network stops functioning as soon as both of its BS(s) become isolated (that means BSs become orphans). This implies that the network can continue its functioning with only one tree in case the other one becomes orphan well before the second one. A node becomes orphan node when all its child nodes terminate their connections due to power outage. This dual tree switching scheme is depicted in the Algorithm 3.5.

In this research, we assume push-based data acquisition model where the sensor nodes of the network keep on sending (pushing) data to the BS using the routing structures formed as explained above. In the routing tree structure, the data packet sent by each node reaches the BS with multi-hop communication. The nodes at the lowest level of the tree are termed as *leaf nodes*. They sense data and send to their respective parents. Such parents are termed as *intermediate nodes*. Therefore, each intermediate node apart from sensing and sending its own data also has to receive and transmit the data packets of its child nodes. All the data packets in the network are destined to reach the BS. We have observed that these intermediate nodes are taxed more than the leaf nodes in the tree structure. Even though we apply our WATC to distribute the workload among nodes at same level of the tree, we are unable to balance the workload of these intermediate nodes in an effective way. As a result of this, the intermediate nodes, closer to the BS, exhaust their power early and get disconnected from the network early. Hence, the overall network comes to a halt soon.

In order to address the above issue, we propose DTDR scheme in which we use two different tree structures as mentioned earlier, which become operational. Due to this, the role of the each intermediate node is changed after a fixed power interval. According to this, in one spell, node at level *l* will become node at level *n-l* (approximately) in the succeeding spell, where *n* is the maximum number of levels in the tree. This results in uniform power consumption at intermediate nodes in the network. Due to this, the intermediate nodes are retained in the network for a longer time. This helps in longer connectivity of the nodes to the network. Thus, it improves the overall lifetime of the network. The leaf nodes, which were having larger residual power previously, are now found to utilize the power in the tree operations heavily due to the change in the role in the succeeding spell, and vice-versa.

Algorithm 3.5: Let $node_i$ be the i^{th} node in the network of ‘n’ nodes, ‘T’,‘k’ be integers, *residual_power*($node_i$) returns the (current/remaining) power in the i^{th} node of the network *power*($node_i$) returns the in the i^{th} node of the network, o*perate_tree(*Base_station) performs the tree operation with Base_station as its base-station. CNL(*node*) returns the Child Node List of the *node*. FBS is First Base-Station 1, SBS is Second Base-Station.

```
1.  Procedure Dual-tree(FBS,  SBS)
2.  Begin
3.  T= $\sum_{i=1}^{n}$ power($node_i$)
4.  While((CNL(FBS) ==NULL)&& (CNL(SBS) ==NULL))
5.  k=2;
6.  if ( $\sum_{i=1}^{n}$ residual_power($node_i$) > (T/k))
7.  operate_tree(FBS);
8.  else if ($\sum_{i=1}^{n}$ residual_power($node_i$) <= T/k)
9.  operate_tree(SBS);
10. end if;
11. end if;
12. k=k+2;
13. End while;
14. End Procedure.
```

Illustration of DTDR Scheme

Now, we explain the functioning of our DTDR scheme with an illustration. Let us consider the example WSN in Figure 3.3. We balance the FHF tree with the node 0 as FBS by applying the Workload-Aware load balancing scheme as explained in

Section 3.2. The FRT thus obtained would be as specified in Figure 3.15. Now, we choose node 11(SBS) which is one of the nodes in the last level of the FRT as the BS for the SRT which we plan to construct. The same process which was carried out for constructing FRT is used to construct SRT, and the resultant tree is shown in Figure 3.16.

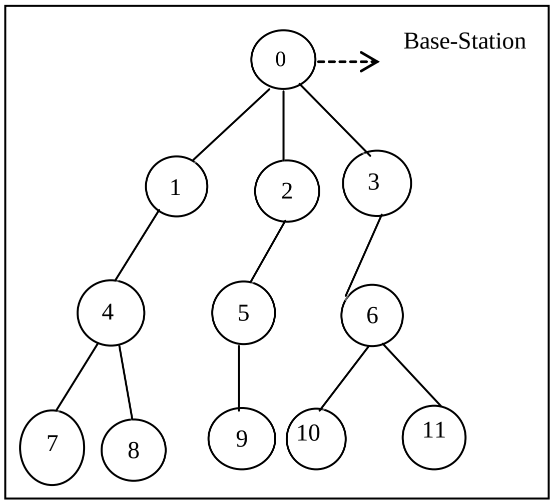


Figure 3.15 First Routing Tree of the network.

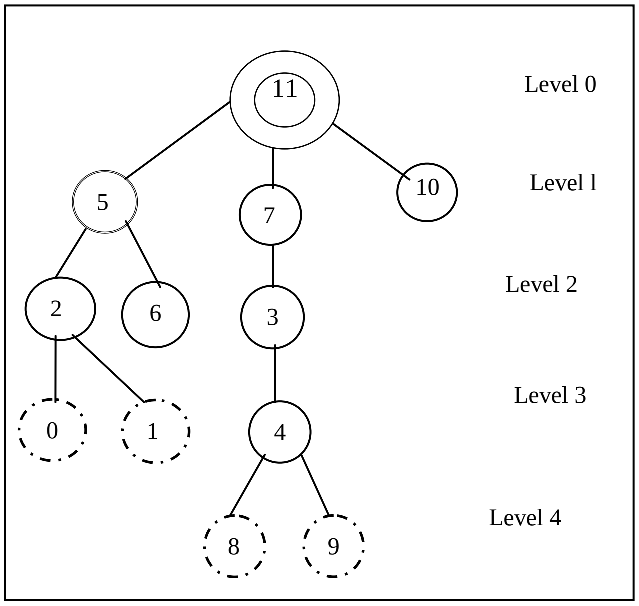


Figure 3.16 Second Routing Tree for the example WSN in Figure 3.3.

According to the example under illustration, FRT shown in Figure 3.15, starts its operations of sensing and sending sensed data to the FBS till the network switches to

SRT due to drop in the residual power levels. In the succeeding spell the network uses SRT for data routing as shown in Figure 3.16. This tree carries on its operations with node 11 as SBS till it switches back to FRT. This way the network switches between FRT and SRT till both BSs become orphans. With this scheme, the energy dissipation in the WSN is uniform.

3.4.2 Performance Evaluation

We present a brief report on the series of experiments we have conducted on our custom-built simulator to assess the effectiveness of the DTDR scheme. We compare the efficacy of DTDR dual tree scheme against the performance of the single tree routing scheme constructed using our WATC approach. Though both DTDR dual tree scheme and WATC single tree scheme are our proposals, we make a comparison between these two to establish the advantage of using dual tree over single tree schemes.

Since we plan to conduct experiments to assess the power consumption of our technique, we have also designed our simulator wherein the power allotted to each node can also be defined. The deployment of nodes using our simulator can either be random or predefined. To simulate a push-based data acquisition model, the sensor nodes send the sensed data to the base-station continuously. Further detailing about the simulator is given in Appendix-1.
A set of experiments were conducted to compare the performance of DTDR dual tree scheme against WATC single tree routing scheme. The comparison is made based on the following metrics.
1. The number of packets transmitted during the lifetime of the network.
2. The lifetime of the network.
3. The residual power available in the network after the network comes to halt.

We have done our experimentation with networks of 20, 50, and 70 nodes. In each case, we simulated five different topologies. In this section, each of the performance metrics is the average of the five simulation runs for the above mentioned networks with tree structures for data routing. We have computed: (i) number of packets transmitted in the network during its lifetime, (ii) network lifetime, (iii) residual power in the network (the amount of power left unutilized at the end of its lifetime), for each simulation. The average values of our results for each experiment are depicted in the graphs presented in this section.

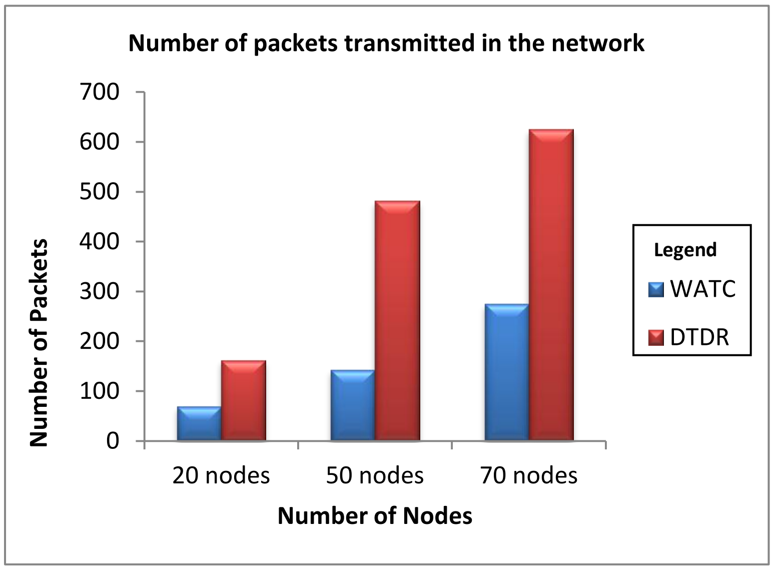


Figure 3.17 Graph showing the number of packets transmitted in the network in WATC against DTDR scheme.

From Figure 3.17, we observe that our DTDR dual tree scheme has shown a good increase in the number of packets transmitted, when compared to WATC single tree scheme. From the results it is evident that as the size of the network increases the total number of packets transmitted also increases.

The graphs in Figure 3.18 show the network lifetime which we consider as a significant parameter to quantify the effectiveness of the network. The network lifetime is the time elapsed between starting of the network and the moment it halts. A network is considered to have reached a halt state when not even a single node is connected to the BS. Otherwise, we say that the network is alive if at least one node in the network is active and connected to the BS. The graph shows excellent results associated with our approach when applied on network with 20, 50, and 70 nodes. A significant increase in the network lifetime can be seen in networks with 50 and 70 nodes. This shows that our proposed approach provides better lifetime for the networks when compared to the earlier technique. We know that longer the lifetime, higher will be the energy efficiency of the network. As the network spends maximum amount of energy for packet transmission, battery power is diminishing at each of these nodes after each transmission. Optimizing the battery power utilization is one of the major issues in WSN. We measure the power utilization of WSN by considering the residual power in the network after the network dies. The residual power is the sum of the residual powers at all the nodes, which are alive and not connected to the BS.

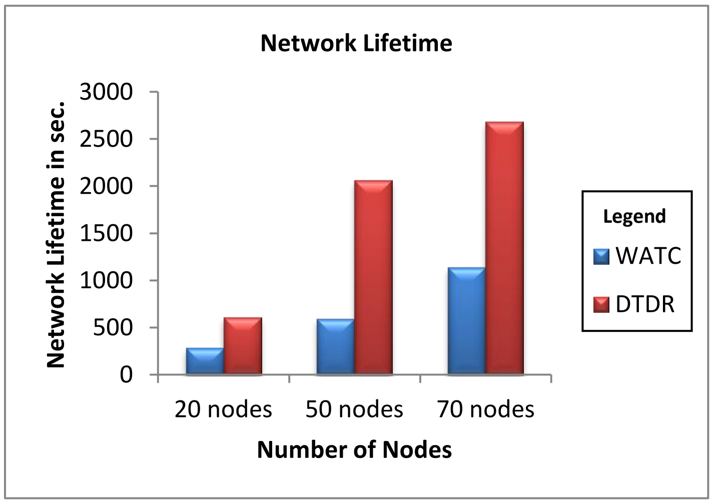


Figure 3.18 Graph showing the network lifetime in WATC against DTDR scheme.

The graph shown in Figure 3.19 depicts the total residual power in the network. In the DTDR dual tree scheme, the nodes undergo role reversal (due to shift in the tree levels) after a power interval, enabling them to live longer when compared to WATC single tree scheme and thus have an efficient way of utilization of power in the network. The graph in Figure 3.19 confirms this speculation.

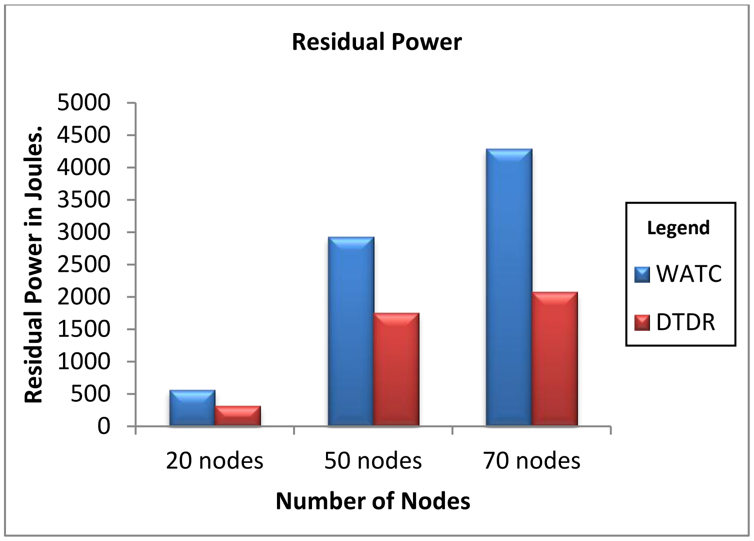


Figure 3.19 Graph showing residual power of the network in WATC against DTDR scheme.

Now in Figure 3.20, the WATC single tree scheme is compared with DTDR dual tree scheme in terms of number of nodes alive after transmitting 50 packets in networks with varying number of nodes. We observe that our approach outperforms the WATC single tree approach as the number of nodes alive is very high. This is due to shift in the roles played by nodes.

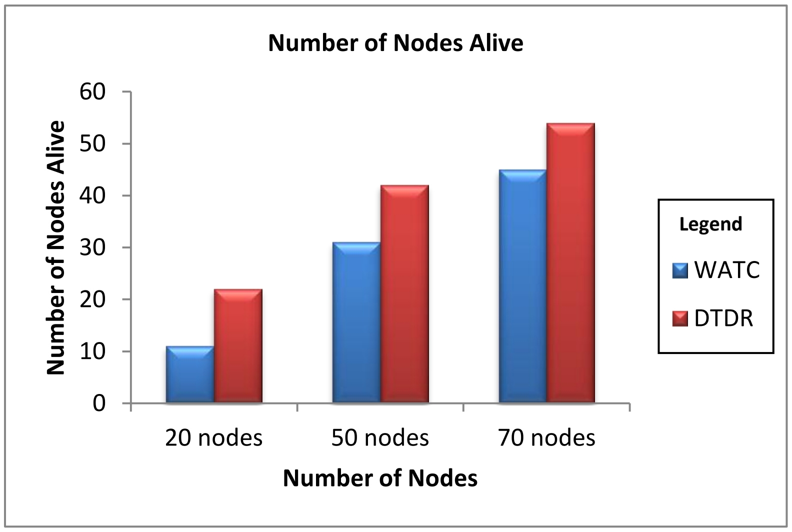


Figure 3.20 Graph showing the number of nodes alive in the network in WATC against DTDR scheme.

From the above experimental results, it is proved that our DTDR dual scheme has optimal power utilization in comparison to WATC single tree scheme. It is also shown that the proposed scheme keeps more number of nodes alive when compared to the WATC. Hence, we conclude that DTDR dual tree scheme is successful in optimizing the energy consumption, and increasing effectiveness of the network through the strategy of changing the roles of intermediate nodes after a power interval. According to this, an intermediate node handling high workload in one spell, will change its role with respect to its location in the structure of the tree and is bound to handle lesser loads in the subsequent spell. It is observed that during the time of switching from one tree to the other the data transmission activity is temporarily stopped for some time and it is obvious that some packets that are under transmission and in the middle of the way may be dropped. This situation can be overcome by devising appropriate mechanism for re-transmitting dropped packets from respective source nodes. This may require some buffering and re-transmission mechanism at node-level. This we would like to address in our future work.

3.5 Summary

In this chapter first we presented our Workload-Aware Tree Construction (WATC) scheme which is effective in balancing the load among the nodes in the network. The performance of this scheme is compared with that of existing ETC approach and found to be more efficient. The second proposal of this chapter is our Workload-Aware Path Repairing (WAPR) scheme intended for effective recovery from node failure situation. Experimental results show that our WAPR scheme works better than the existing path repairing approach. Our last proposition in this chapter is our Dual Tree Data Routing (DTDR) scheme with two base-stations. There exist two routing trees but, only one tree is operational at a given point of time. Each tree has one of the BS as the root, and one BS can become root node of only one tree. This DTDR tree adopts our WATC algorithm and WAPR scheme for construction and maintenance of each tree in the pair. Our experimental results have showed that our proposed dual tree approach works better than the single tree schemes in conserving energy in the network. One important assumption made in our scheme is that the network will have two nodes capable of playing the role of a BS and are located at the opposite ends of the topology.

Though our novel tree construction and maintenance schemes proposed in this chapter give excellent results for sensor networks that adopt push-based data acquisition model, they can be used in all WSNs where tree based routing is adopted and load distribution is almost uniform.

Chapter 4
Efficient Data Compression Scheme for WSNs

In this chapter we present our novel lossy data compression scheme to achieve energy efficiency by reducing the volume of data to be transmitted. We propose a lossy data compression scheme- "*Induced Redundancy based Lossy Data Compression Algorithm*" (IR-LDCA) [70], to achieve better compression ratios than the existing ones, for the data sets which possesses reasonable degree of *temporal correlation* among the data values. Our algorithm induces certain amount of redundancy into the data set to achieve more effective data compression and also gives the user a flexibility to control the compression ratio and precision. Towards the end of the chapter we also present the experimentation results which prove the effectiveness of the proposed scheme.

4.1 Introduction

One general observation is that when the data values in a dataset exhibit a reasonable degree of correlation, we can apply data compression schemes to reduce the size of the set. This is a proven fact for many applications and also very much relevant in case of data sets handled in WSNs, since the sensor nodes are deployed to capture physical properties in a continuous manner. We observe that such data sets are more suitable for compression.

We got motivated towards the work in the area of data compression in WSNs, when we came across the data set obtained from Intel Berkeley research lab [71]. This data set contains temperature, voltage, humidity and pressure data at regular time intervals. These sensor data readings which are floating point values, exhibit certain degree of correlation. Sometimes, they are found to be redundant as well. The temporal correlation is the proximity that exists between two data values pertaining to two consecutive time points. As the sensor nodes are deployed densely in the area of interest for satisfactory coverage, it is possible to have multiple sensor nodes sensing same physical property like temperature. The spatially dense sensor nodes capture highly correlated data. The degree of correlation increases with decrease in inter-node separation distance. To reduce the volume of data to be transmitted by a node, suitable data compression schemes are applied at the node level.

The data compression schemes can be broadly classified into two categories: *lossless compression* and *lossy compression*. With lossless compression, original sampling data can be perfectly restored at the receiving end i.e., without any loss in the precision of the data. But this hinders achieving higher compression ratios. With lossy compression, some degree of information loss, in terms of RMS error (Root Mean Square error) is present. In order to achieve higher compression ratios, a lossy compression scheme exploits the redundancy in the sensor data. Thus, for WSN applications which doesn't require high precision, lossy compression techniques are more preferable. We present a novel lossy compression scheme called- "*Induced Redundancy based Lossy Data Compression Algorithm"* (IR-LDCA) [70] to achieve higher compression ratios with minimal loss of data. We also conducted experiments to compare the performance of our scheme with the existing lossy compression scheme K-RLE [48], described in Chapter 2, to prove the efficacy of our scheme. The results of the experimentation are discussed in the later parts of this chapter.

4.2 Induced Redundancy based Lossy Data Compression Algorithm (IR-LDCA)

Our lossy data compression algorithm IR-LDCA is based on the following important assumptions about the data sensing and transmission in WSNs. First, we assume that the sensed data has a reasonable amount of temporal correlation. Our second assumption is that a series of sensed data in the form of readings will have to be sent to the BS by a node at a specified time frequency in batch mode. Lastly, we also assume that the sensor hardware should support storage and processing of floating point data.

The IR-LDCA scheme effectively exploits the natural correlation that exists in sensor data. The objective is to exploit the commonality existing in the continuous data stream and also to eliminate redundancy. This is exploited by inducing redundancy in the data set by converting data values to their integer representation. This lossy data compression scheme allows the user to control the compression ratio desired and the data loss during the compression facilitating higher compression ratios with minimal data loss. This scheme reduces the volume of data to be transmitted at a sensor node. Thus, resulting in reduced energy consumption and enhanced lifetime of the network.

Now, we present the full-length description of the proposed algorithm. First, the commonality in the set of readings is explored and then redundancy is eliminated. Let A be an array of temperature values collected by a sensor node over a period of time. The first phase of the algorithm is to identify the commonality in fractional part of two consecutive sensor readings. We calculate the maximum value (*Max (A)*) and minimum value (*Min (A)*) in A, and subtract the minimum value from the maximum to obtain the difference factor- d.

$$d = Max\ (A) - Min\ (A) \qquad (4.1)$$

The *common part* can be calculated by taking the value from any element in A till two place values greater than the place having the most significant non-zero value in d. This *common part* is subtracted from every data item of A. This gives us the modified list A. Now, the new representation for the element of A at index position i is computed as-

$$A[i] = A[i] - common\ part \qquad (4.2)$$

We perform shifting of every value in A by n places, where n is calculated as:
If (*Max (A) >1*), then n is the place value of the most significant non-zero number in *Max (A)*. If (*Max (A) < 1*), then n is the number of zeroes before the most significant non-zero place in *Max (A)*.

By doing this, we can convert every value in A to a new representation which is in the form 0.xyz..., wherein the value of x in *Max (A)* is greater than 0. Now we calculate the value of the variable delta (Δ) using the formula in equation (4.3). Here, Δ gives the maximum error that can be encountered in any element of *A*.

$$\Delta = \frac{\text{range of modified } A * \ 10^{n+c}}{\text{Max(modified } A)} \tag{4.3}$$

In the equation (4.3) above, if *Max (A) <1*, then *n* is – *(n+1)* and the '*c'* can be any value between 0 and 2. The variable '*c'* acts as a control knob for controlling the compression ratio and error.

Now we convert the set of values in *A* to their corresponding integer representations, which are calculated using the formula in equation (4.4).

$$A = floor(\frac{A - Min(A)}{\Delta}) \tag{4.4}$$

The rebuilt data set is ready which can be made into packets. The above explained process, in phase one encodes the floating point data value into their equivalent integer representation, which performs initial compression.

In the second phase of the algorithm, we combine the data and eliminate redundancy in data for achieving further compression. Let z be the minimum bits required to represent the *Max (A).* For example, the binary representation of 8 is 1000; here the minimum number of bits required to represent 8 is 4, this implies z=4.

The procedure given above performs the second phase of data compression. In the *Procedure Combine(A)* given below, we place the *z+1* bit binary representations into a file combining every *4m-1* (where m is any integer greater than 0) consecutive occurrences of any number *k* by *m* single bit 1s. The proposed IR-LDCA has shown good compression ratio and optimized error depending on the value of the control knob *c*.

```
Procedure Combine (A)
Begin
int i, count=1;
From i=1 to length (A)-1
        While A[i] = A [i++] & i < length
                Increment Count
        End
        For j= 1 to Count/4
        Put binary 1 (logical 1) into the binary file.
        End
        For j= 1 to Count %4
        Put the binary equivalent of A[i] calculated upto z+1 bit
        into the file (if needed, do sign extension).
        End
End
          If   i != length (A)
        Put the binary equivalent of A[i] calculated upto z+1 bit
        into the file (if needed, do sign extension).
        End
End Procedure
```

4.3 Illustration of IR-LDCA

Here, we explain the working of the IR-LDCA on a hypothetical data set.

Let the data set A= [[46.89010], [46.89019], [46.89030] …… [46.89061], [46.89062], [46.89063], [46.89065]…… [46.89994]] be a data sample obtained from a sensor node. We compute the difference factor *d* for the above data set by applying equation (4.1)

d= Max (A) – Min (A) = 46.89994 – 46.89010 = 0.00984.

Now, we rebuild the data set A by subtracting the *common part* from every data item. The *common part* as per the algorithm explained in Section 3.1, can be calculated by taking the values from any element of A, till two place values greater than the place having the most significant non-zero value in d. Here, the most significant non-zero place value in *d* is third place to the right of the decimal. Thus, we have to consider till the first value behind the decimal (0.0|0984) for calculating the *common part*. In our example the *common part* is 46.8000. By applying equation (4.2) on the elements of *A*, we obtain the modified version of *A*. For instance, the modified value for the first data item 46.89010 is 46.89010-46.8000= 0.09010.

Now, modified A= [… [46.89061-46.8000]…] = [[0.09010]… [0.09061]… [0.09994]]

Now, we shift the elements in *A* by *n* places to bring them to the form 0.xyz... wherein *Max (A)*, x>0. Here n=1, hence we need to multiply *Max (A)* by 10^1 to bring it to the form 0.xyz with x>0. We further multiply every element in *A* by 10^n.

This makes A= [[0.09010*10^1]...] = [[0.9010]… [0.9061]… [0.9994]].

Using the equation (4.3), we calculate the value of Δ as follows, by taking *c* as 1.

Since *Max (A)* < 1, *n* is taken as – (*n*+1). Hence, *n* = -2.

We have, $\Delta = \frac{(0.9994-0.9010)*10^{-2+1}}{0.9994}$; Thus Δ= 0.0012.

Now change the values in *A* to an integer form by using the equation (4.4).

We have, *A*[0]= floor((0.9010-0.9010)*10-1/0.0012) = 0, similarly *A*[1]=1, A[2]= 2. For some elements in *A* whose values are: 0.9061, 0.9062, 0.9063, 0.9065, the corresponding integer values obtained after applying equation (4.4) are 5,5,5,5. This is termed as induced redundancy.

Thus, $A=\{ 0,1,2, \ldots\ldots \underbrace{5,5,5,5}_{\text{Induced redundancy}} \ldots\ldots,82\}$.

Now, we have to combine similar values in *A* to a single value using the *Procedure Combine (A)* as shown in Section 4.2, to compress the data. Let *z* be the minimum bits required to represent the *Max (A).* Here, the new *Max (A)* is 82. The minimum number of bits required to store 82 is 7, thus, *z*=7.

Hence, after applying *Combine(A)*, the data written to the file corresponding to the data set *A*= {0,1,2, ……5,5,5,5…….,82} is { ***00000000*** 00000001 00000010 ……. ***1000001001*** …… 01010010}

8-bit representation of '0'

8-bit representation of '5' preceded by binary 1 showing a redundancy of four consecutive '5's

Similarly, had there been a redundancy of (say) 9 consecutive '5's then the value stored would have been as follows: 1 1 01010010 01010010. The first 2 ones followed by the binary representation of '5' implying that there is a redundancy of 4*2 repetitions of '5' and the succeeding sequence is showing the ungrouped '5', since we are grouping numbers in multiples of 4. We have implemented the above algorithm on our simulator and evaluated the performance on the sample data.

4.4 Performance Evaluation

To validate the efficacy of our IR-LDCA data compression scheme, we implemented both IR-LDCA and K-RLE algorithms in Java. Since the algorithm works on the data collected at individual nodes, the usual parameters of a WSN like- topology, transmission range etc., will not come into picture.

Evaluation of IR-LDCA

To evaluate our IR-LDCA, we have conducted a series of experiments on the sample data set obtained from Intel Berkeley Research lab. The sample data set comprises of data about temperature and pressure. As the performance of data compression algorithms is assessed based on their compression ratio and error percentage (this indicates the degree of loss), various tests were conducted on IR-LDCA and K-RLE, for similar performance metrics. The data set contained a sequence of 1305 values for each physical quantity. The compression ratio is computed as shown in equation (4.5), given below.

$$\text{compression ratio} = 100 * \left(1 - \frac{compressed\ size}{initial\ size}\right) \tag{4.5}$$

The Root Mean Square (RMS) error is computed as shown in equation (4.6), given below.

$$RMS\ Error = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{2a}} \tag{4.6}$$

In IR-LDCA, the value of *c* (control knob), used in equation (4.3), can be altered to get the desired compression ratio for each parameter (temperature/pressure). In the graph shown in Figure 4.1, the compression ratios for various values of *c* have been indicated for both pressure and temperature. The compression ratio depends on the value of *c*. The x-axis of the graph represents the variable values for c, and the y-axis represents the compression ratio obtained.

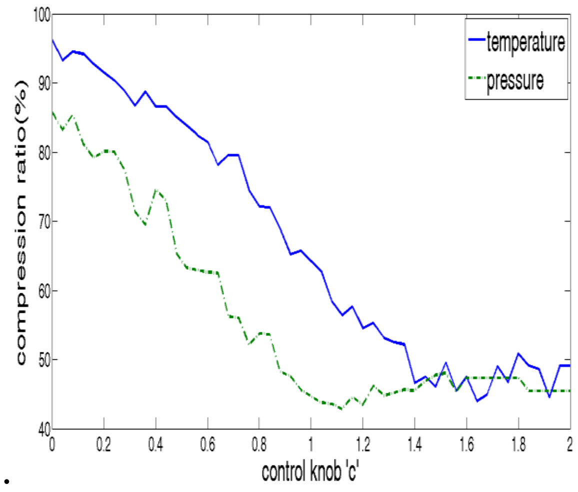


Figure.4.1 Compression ratios achieved for varying values of c for temperature and pressure

The value of the compression ratio for temperature decreases gradually from 95.4% to 45% and the pressure decreases from 85.8% to 45.4%, as the value of *c* changes from 0 to 2. The line graph thus obtained shows a gradual change in compression ratio as the value of *c* changes. This is because, as the value of *c* increases, the value of delta (Δ) obtained in equation 4.3, decreases and this result in lowering of the compression ratio. It is observed that, most of the times, the change in compression ratios for both temperature and pressure show similar behavior (both of them either increase or decrease), between any pair of values for c. This is due to the fact that the data samples are not mutually correlated.

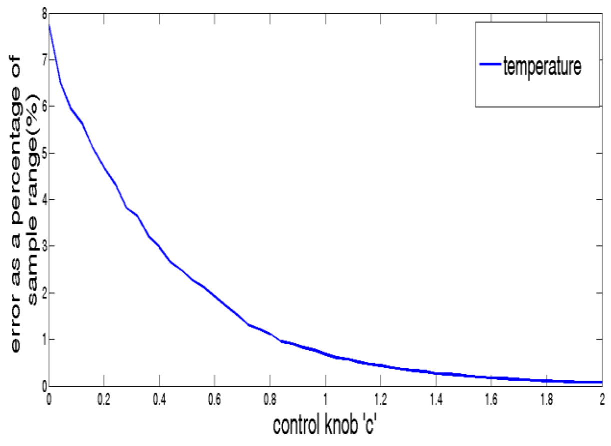


Figure 4.2 RMS error percentages for varying values of c for temperature.

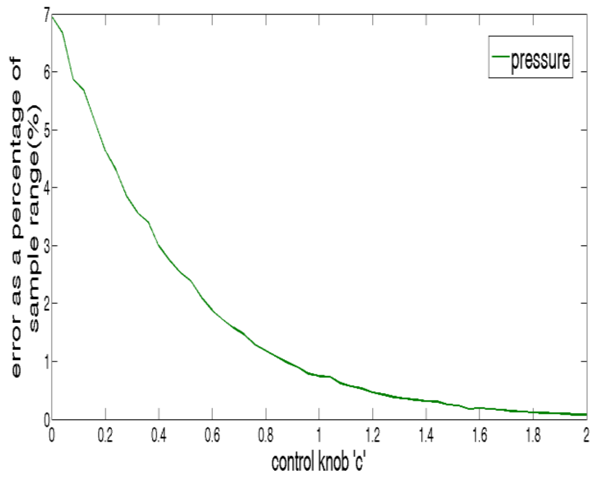


Figure 4.3 RMS error percentage achieved for varying values of c for pressure.

The RMS error percentage (RMS error*100) for varying value of *c*, for temperature and pressure, are plotted in Figure 4.2 and Figure 4.3 respectively. In both the graphs, *c* is plotted on the x-axis and the y-axis represents the error percentage obtained. The value of the error percentage for temperature decreases gradually from

7.75% to 0.072% in Figure 4.2, as value of *c* changes from 0 to 2. In Figure 4.3, the graph shows decrease in the error percentage for pressure from 6.95% to 0.0684%. The line graph thus obtained shows a gradual change in error percentage when the value of *c* changes. This is because, as the value of *c* increases, the value of delta (Δ) decreases resulting in decreased error percentage.

Comparison between IR-LDCA and K-RLE

To establish the effectiveness of the proposed IR-LDCA over the existing K-RLE algorithm, the K-RLE scheme is also implemented and tested for its performance on the same platform. The comparison parameters are compression ratio and error percentage. First, we conducted experiments to compare the compression ratios. In order to make a comparison between K-RLE and our proposed IR-LDCA, we have considered the value of K as the standard deviation of the data sample and c corresponding to their respective highest compression ratios. Next, we conducted experiments to compare the RMS error percentage using similar approach as mentioned above for comparing compression ratios.

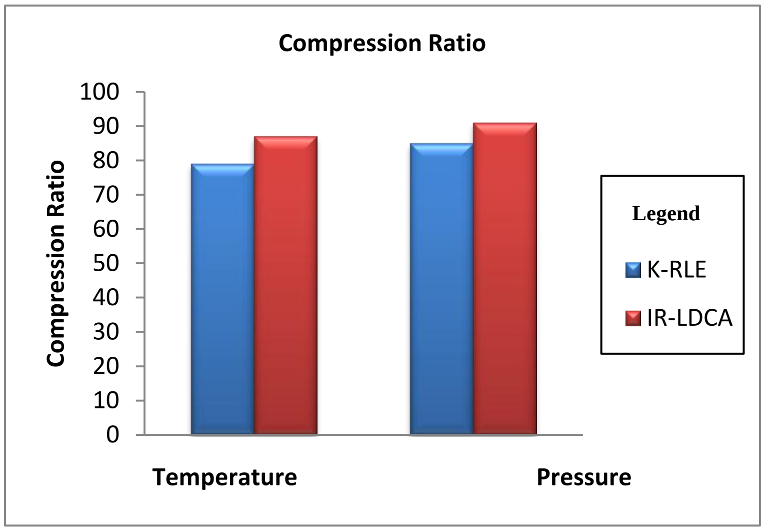


Figure 4.4 Compression ratios achieved in K-RLE and IR-LDCA for temperature and pressure.

The compression ratio obtained for K-RLE is 79.7% and 85.4% for temperature and pressure respectively. In case of IR-LDCA, it is 87.2% and 91.8% for both parameters. The compression ratios given here for IR-LDCA are corresponding to the minimum value of *c*.

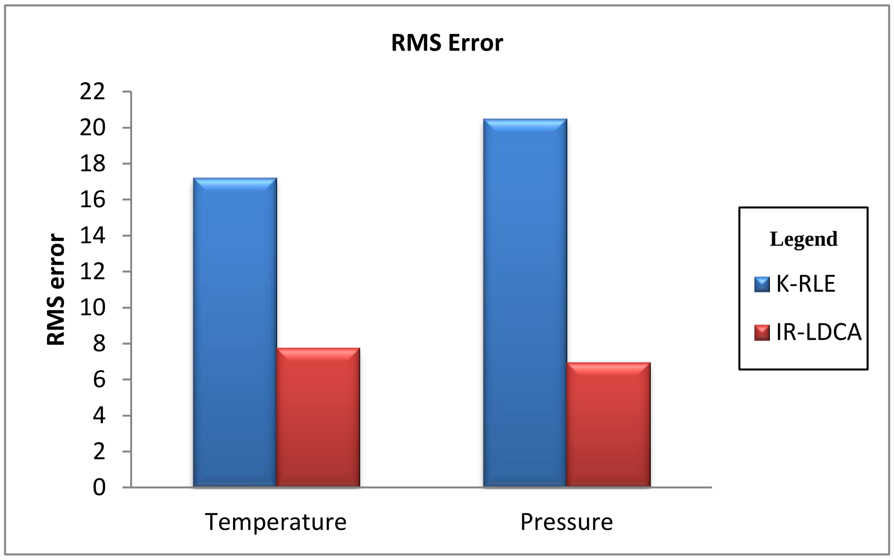


Figure 4.5 RMS error percentages in K-RLE and IR-LDCA for temperature and pressure.

The error percentage obtained for K-RLE is 17.2% and 20.5% for temperature and pressure respectively. In case of IR-LDCA, it is 7.75% and 6.95% for both temperature and pressure respectively.

The results obtained reveal the fact that the improvement in the compression ratio in IR-LDCA against that of the K-RLE is nominal. But our IR-LDCA shows a drastic reduction in RMS error percentage, when compared to that of K-RLE. This proves the advantage of our IR-LDCA over the existing K-RLE algorithm.

4.5 Summary

Our novel lossy data compression algorithm IR-LDCA presented in this chapter, exploits the temporal correlation that exists in the sensor data in a more effective manner. The objective of the work was to exploit the commonality existing in the continuous data stream and also to eliminate redundancy. The IR-LDCA allows the user to control the compression ratio and the data loss as desired during the compression, facilitating higher compression ratios with minimum loss of data. Our algorithm is more flexible and optimized than the existing K-RLE algorithm with respect to the RMS error (data loss). Our experimental results prove that the application of our novel data compression technique reduces energy consumption and increases the lifetime of the network.

Chapter 5

Efficient Query Processing Scheme for WSNs

In this chapter we present our novel query processing scheme [72] for WSNs. The main objective is to achieve energy efficiency in a WSN by reducing the number of messages transmitted during query processing to minimize the power consumption. Our approach is based on making use of the cached data (results) at the BS and exploiting the commonality among the queries (query containment) submitted to the network. This can significantly minimize the transmission and processing costs w.r.t., energy in the network. Finally, we present the simulation results which prove the effectiveness of this scheme over the existing ones.

5.1 Introduction

Majority of the WSN applications submit ad-hoc queries to the BS for processing. As mentioned earlier, an ad-hoc query is a one-time data retrieval request submitted to the network (at the BS). Some of the applications are- Waste Water Monitoring, Green House Monitoring, Air Pollution Monitoring, Machine Health Monitoring, Landslide Detection, Forest Fire Detection etc. These applications monitor certain physical characteristics like temperature, pressure, humidity, light, air pollutant etc. Hence, the external world queries the WSN for these parameters. The role of each sensor node is to sense and transmit the sensed data to the central monitoring station termed as the *Base-Station* (BS). We assume that both BS and the sensor nodes are capable of processing queries. The only difference between a sensor node and BS is in terms of the resources available. The battery attached to the sensor node is the source of energy for performing various operations like sensing, data storage, computations, and transmission. It is observed that 80% of the power is utilized towards radio communication. Hence, to increase the lifetime of the network, it is essential to minimize the volume of data transmission during query processing. Therefore, we have focused on minimizing the cost of executing ad-hoc queries.

Usually, an ad-hoc query submitted at the BS is sent to respective network of nodes for execution. The results of a query are given back to the user through the BS. Hence, all the nodes send their result data to the BS. As the BS has a reasonable amount of storage, the results of a query can be stored at the BS for future use. We call such data stored at the BS as *cached results*. If the query request involves cached results, the BS can process the query and give the results. Otherwise, the query needs to be sent to the concerned nodes for execution.

5.2 Ad-hoc Query Processing on Cached Data at Base-Station

A set of ad-hoc queries submitted at BS are often found to be requesting the same results. Sometimes they may not be equal, but they exhibit certain overlap in their query results. Such relationship among queries is termed as *query containment*. The containment relationship is said to be satisfied, iff two queries yield same or partially same query results. For the sake of convenience w.r.t., terminology, we call a query q that yields the superset of the results of a set of queries $Q\{q_1....q_n\}$, as a *super-query*. This super-query q is formed based on Q. Hence, the challenge is to formulate a super-query-set for a set of queries which can be disseminated and executed in the network leading to reduced transmission cost in terms of energy. Further, a query submitted at BS may need to be processed- (i) completely using the cached data at BS or (ii) completely using the data available in the network, or sometimes (iii) partly using the cached data and partly using the data available in the network.

Now, we explain the first phase of query processing scheme at the BS. The queries submitted at the BS are stored in a query register. Now, each query from the query register may obtain its results completely or partially on the cached results at BS. Sometimes, the cached results may not be useful in any way. In case a query finds its complete results at the BS, then the query need not be disseminated into the network. But, if a query yields partial results at the BS then the remaining results are obtained from the network of nodes. For certain queries which don’t find results at the BS, need to get their query results from the network. Now, if we examine the queries which need data from the network, they may exhibit certain overlap in their query results. For a given set of queries, that are submitted at BS in a given time interval, we capture the data requirements using a *bit-map*. After elapse of the specified time period, a set of super-queries are framed from the bit-map which will be disseminated into the network. The query results obtained from the network of nodes will be stored in cached results at the BS. Now, the BS has complete results for all the queries in the register. Therefore, the queries from the query register are executed upon the cached results. The query results are given to the user. Now, we present an illustration. For simplicity, we have considered 4 sensor nodes in the network and time interval for each sensor node as one hour. We consider 5 queries {q_1, q_2, q_3, q_4, q_5} in the query register as follows.

q_1: SELECT * FROM S1, S2 WHERE ***t*** > 1 AND ***t*** < 2;

q_2: SELECT * FROM S1, S3 WHERE ***t*** > 2 AND ***t*** < 3;

q_3: SELECT * FROM S1, S4 WHERE ***t*** > 3 AND ***t*** < 5;

q_4: SELECT * FROM S3, S2 WHERE ***t*** > 1 AND ***t*** < 4;

q_5: SELECT * FROM S4, S2 WHERE ***t*** > 3 AND ***t*** < 6;

Let us assume that initially the BS has data from Sensor nodes S1, S2, S3, S4 for time interval, t=1 to t=2, which forms the cached results. The query q_1 needs data from Sensor nodes S1, S2 for time interval t=1 to t=2, which is available at the BS. Therefore, q_1 is answered at BS itself.

With the available cached results at the BS, each query in the query register may yield complete, partial or no results. But, the challenge is to identify the data requirements of each query. To solve this problem, we propose a simple technique that uses a bit-map. In general, a WSN is queried for the sensor value(s) during a fixed time interval. The query results are with two columns viz., the time-stamp and its sensor values. Now, we frame an empty two column query result table, where the first column values are generated with the time-stamp for the required time interval. The second column is the sensor values for their respective time-stamps, which can either be obtained from the cached results at the BS or from the network of nodes. Therefore, we fetch sensor values for the required time-stamp from the cached results at the BS if available. For the sensor values which are not available in the cached results they need to be retrieved from the network of nodes. Thus, values for the second column are available either for all rows or only for some rows. The following situations arise based on the values in the second column of the query result table-

1. The second column is completely filled that means query results are available in the cached results at the BS.

2. The data is partly filled, that means there are empty locations for which data is unavailable at the BS.

3. The second column is empty, which means that the query cannot be answered at the BS.

In the first case, the query can be completely answered at the BS itself without transmitting the query into the network. The second case gives partial results at BS and the rest from the network. The third case is to completely retrieve from the network. In the second and third cases, where the data needs to be retrieved from the network, there may be certain redundant data requirement among the queries. Hence, we try to minimize such redundant data transmissions in the network. We present a simple technique to solve this using our bit-map approach. The flowchart for the same is shown in Figure 5.1.

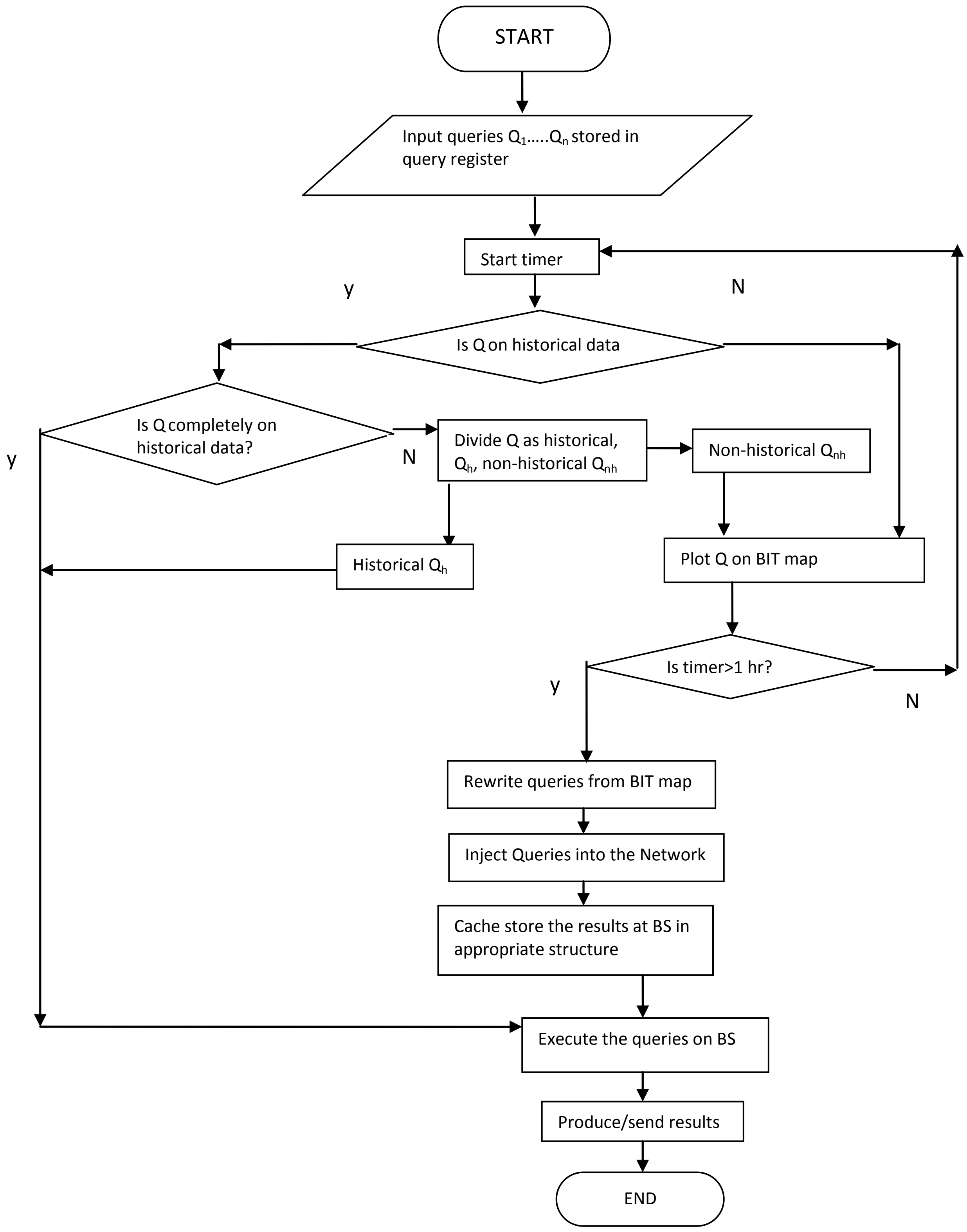

START
Input queries Q1.....Qn stored in query register
Start timer
y
N
Is Q on historical data
Is Q completely on historical data?
y
N
Divide Q as historical, Qh, non-historical Qnh
Non-historical Qnh
Historical Qh
Plot Q on BIT map
Is timer>1 hr?
y
N
Rewrite queries from BIT map
Inject Queries into the Network
Cache store the results at BS in appropriate structure
Execute the queries on BS
Produce/send results
END

Figure 5.1 Flowchart showing the workflow for the proposed query processing scheme

5.3 Exploiting Query Containment (Bit-Map Approach)

5.3.1 Identifying Query Containment Relationship

In general, the queries which do not yield results at the BS are disseminated into the network. In our example, q_2 cannot be answered at the BS. We capture the data requirements of these queries in a 2D bit-map. This 2D bit-map is an array of bit-maps where each bit-map corresponds to a sensor node.

Now we explain the details of the bit-map. Each bit position in the sensor bit-map represents a time-interval. The queries obtained from second and third case in Section 5.2 are the candidates for the bit-map. We exploit commonalities in multiple query results by mapping all the query requirements onto the bit-map. Initially, the bit-map has ‘0’ for all its bits.

A sample bit-map is shown in Figure 5.2 below. A bit-map is meant to capture the data requirements for each sensor. The data required from each sensor for a particular time interval is shown as a bit-map. Initially the bit-map is empty, which is shown with ‘0’ in all its bit-positions.

Sensor nodes/ time	t=1	t=2	t=3	t=4	t=5	t=6
S1	0	0	0	0	0	0
S2	0	0	0	0	0	0
S3	0	0	0	0	0	0
S4	0	0	0	0	0	0

Figure 5.2 Sample bit-map showing initial value ‘0’ in all its cells.

The data required at a particular time interval is marked as bit ‘1’ against their time-interval. Whenever bit ‘1’ is found existing in a cell, ignore that and proceed. It means this particular data has already been in the requirement of some other query. Now, all the incoming queries are plotted on the same bit-map for a certain time period. Depending on the application, the time period can be set. After the time period is elapsed, a set of super-queries are framed from the bit-map.

In our example, q_2 needs data from Sensor nodes S1, S3 for time interval t=2 to t=3 which is not available at the BS. So, the query is plotted onto the bit-map of Figure 5.2. Now, the value 1 is plotted in the bit-map of S1, S3 against the time interval t=2 to t=3 and we get the modified bit-map as shown in Figure 5.3.

Sensor nodes/ time	1	2	3	4	5	6
S1	0	1	1	0	0	0
S2	0	0	0	0	0	0
S3	0	1	1	0	0	0
S4	0	0	0	0	0	0

Figure 5.3 Bit-map after plotting q_2.

Sensor nodes/ time	1	2	3	4	5	6
S1	0	1	1	1	1	0
S2	0	1	1	1	1	1
S3	0	1	1	1	0	0
S4	0	0	1	1	1	1

Figure 5.4 Bit-map after plotting q_2, q_3, q_4 and q_5.

Now consider query q_3, which needs data from S1, S4 for a time-interval t=3 to t=5. This data is not available at the BS. So, plot this on the bit-map obtained in Figure 5.3. Let us take up the query q_4, which needs data from S3, S2 for a time interval t=1 to t=4. First it is executed on cached data at the BS. We have cached data for a time interval of t=1 to t=2. But, we also need data from t=2 to t=4. Hence we plot this requirement (for unavailable data) on the bit-map. In case, if a particular bit position in the bit-map has value '1' before plotting, this can be ignored, as it has already been the requirement of some other query considered earlier in this process. In this way, we use the same bit-map to plot the requirements of all the queries by updating the bit-positions. The Final bit-map after plotting all 5 queries is shown in Figure 5.4 below.

5.3.2 Framing Super-Query-Set

At the BS, we frame the super-query-set from the bit-map on hourly basis or once every six hours depending on the application and user requirement. With an increase in this time period, the number of queries plotted on bit-map also increase. To frame the super-query-set from the 2D bit-map, we scan the individual bit-maps of the sensors. Whenever there is an entry ‘1’, its corresponding time-stamp is considered as the lower bound and continue scanning its adjacent cell for entry ‘0’ to indicate the upper bound for the time series. Now, we get the upper and lower bounds time-stamp for each sensor from the bit-map. This forms the basis for data requirement for each sensor. With this information, we can frame a super-query for each sensor with the lower and upper bound time-stamps.

We consider Figure 5.4 for generating super-queries. Each row specifies the required data from each sensor for their time interval. Like sensor node S1 needs data from the network for a time interval of t=2 to t=5. Hence, the super query that can be generated for S1 is SELECT * from S1 where t>2 and t<5. Similarly, super-queries for the other sensors can be given as follows.

SELECT * from S2 where t >2 and t < 6.

SELECT * from S3 where t > 2 and t < 4.

SELECT * from S4 where t > 3 and t < 6.

Hence, the super-query-set generated will minimize the volume of query/result transmission in the network. The same process is repeated for all bit-maps to obtain a set of super-queries. The super-queries are disseminated into the network. The query results obtained from the network are cached at the BS. Each query in the query register is now executed upon the cached results at BS. Hence, the query results are delivered to the user. After this, bit-maps are set to ‘0’ to make it ready for the next round of plotting to process the subsequent batch of queries.

5.4 Performance Evaluation

In this section, we discuss the details of the performance evaluation of the proposed query processing scheme that exploits cached results and query containment.

We implemented our novel query processing technique on the custom-built simulator developed in Java. We assume few parameters in the network to be constant (like transmission range) and there are no failures in the network. Our simulator is built to work for network with varying sizes. We have considered sensor nodes to form a fixed/static routing tree structure. The transmission range for each sensor is fixed as 10 distance units. Each of the sensors is initialized with power 100Joules.

We assume that the sensors are employed for sensing temperature. We have obtained real-world data sets from Intel Berkeley research lab data [71]. We have formulated user queries assuming certain data needs. We have limited our system to have input query-set that show variation only w.r.t., time interval, to reduce the complexity. Initially there is no cached data at the BS. As queries request data from the network, as and when the result is received, the BS stores a copy of the same, which becomes cached data.

Experimental Results

A set of experiments were conducted to show the performance of our proposed query processing approach. The comparison is made based on the following metrics.
1. The number of packets transmitted in the network for *‘n’* queries.
2. The lifetime of the network.
3. The number of queries executed in the lifetime of the network.

We have conducted our experimentation on networks of varying sizes. The presented reading for each performance metric is the average of the five simulation runs conducted on the simulator. We have computed- (i) number of packets transmitted, (ii) network lifetime and (iii) the number of queries executed in each experiment. The average values of our experiment results are depicted in the graphs presented in this section. In each experiment, we have used the same set of 1000 queries to compute the number of packets transmitted in each of the networks. From Figure 5.5, we observe that for executing 1000 queries in the network, our bit-map approach has shown a good decrease in the number of packets transmitted. From the results it is evident that as the network size increases the number of packets transmitted in the network decreases.

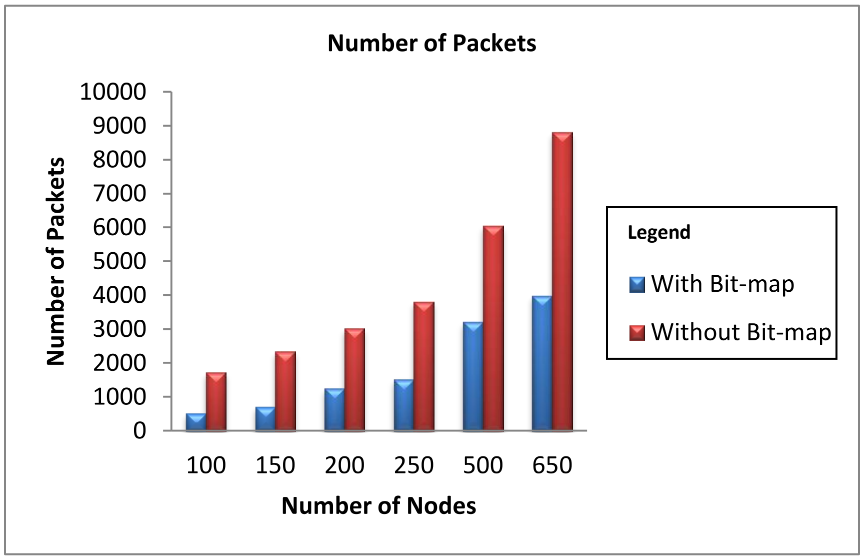


Figure 5.5 Graph showing the number of packets transmitted in the network for executing 1000 queries.

Our next experiment quantifies the number of queries executed during the network lifetime. The graph in Figure 5.7 shows the number of queries executed in the networks varying in size from 100 to 650. We observe that as the network size increases, there is a steep increase in the number of queries executed during the lifetime of the network.

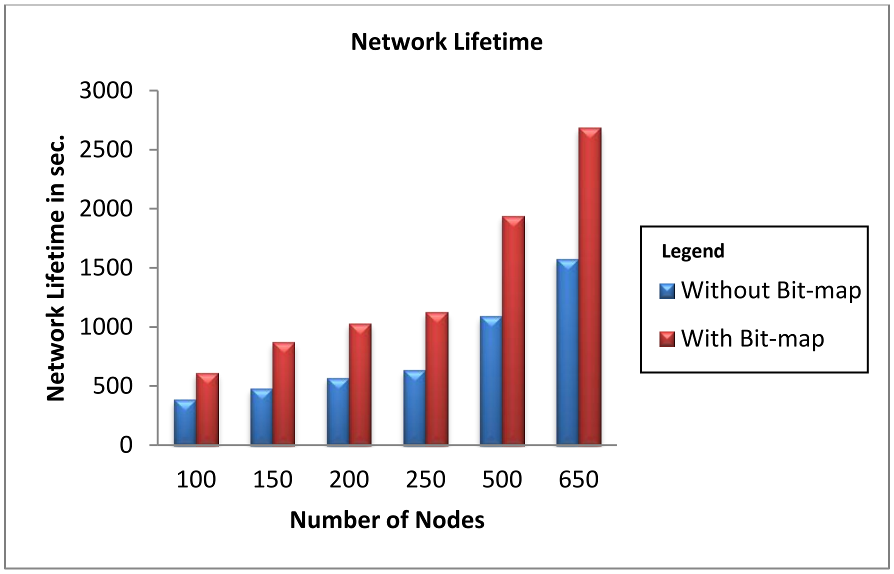


Figure 5.6 Graph showing the network lifetime.

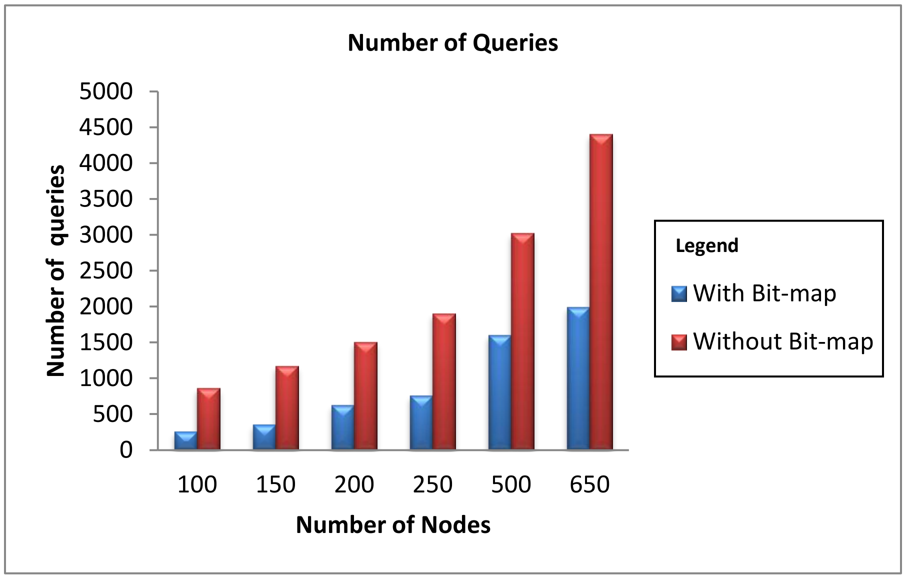


Figure 5.7 Graph showing number of queries executed during the lifetime of the network.

From the above experimental results, it is clear that our proposed approach results in optimal power utilization. In addition, we have observed that effectiveness of our approach increases with the network size. Hence, we conclude that our approach is successful in conserving the network power and increasing its lifetime.

5.5 Summary

Our novel query processing technique presented in this chapter exploits cached results at BS and query containment among the ad-hoc queries to be executed in the network. If the query results can be obtained in its entirety from the cached results then we execute that query against the cached results at the BS and send the results to the user. Otherwise if the query results need to be extracted partially or completely from the network nodes, then we figure out the commonalities w.r.t., the data requirements of all such ad-hoc queries to formulate super-queries, and transmit the same to the required nodes for execution. We introduce a bit-map approach for identifying the commonalities (containment) among the queries and to formulate super-queries. This helps in minimizing the number of queries executed and the volume of data transmitted in the network which eventually leads to optimal utilization of the power and longer network life. Our experimental results prove the effectiveness of our approach.

Chapter 6

CONCLUSION

In this thesis, we presented novel approaches for conserving energy in WSNs, during the data routing, transmission, and query processing operations. Our proposals mainly focus on routing tree construction and maintenance, reduction of volume of data under transmission and increasing the efficiency of query processing. Our proposed techniques tried to address the limitations of the existing schemes in the respective operational areas mentioned above.

In this thesis we proposed-

(i) a novel *Workload-Aware Tree Construction* (WATC) scheme, for efficient workload balancing among nodes in a tree.

(ii) a novel *Workload-Aware Path Repairing* (WAPR) scheme for effective path repairing to recover from node failure situations.

(iii) a novel Dual Tree Data Routing (DTDR) scheme to facilitate energy efficient operation of the network by uniform distribution of load across the nodes leading to longer network life.

(iv) a novel lossy data compression scheme called *Induced Redundancy based Lossy Data Compression Algorithm* (IR-LDCA), to reduce the data transmission costs and energy consumption.

(v) a novel query processing scheme which exploits the cached results at BS and the containment relationship among the queries submitted at the BS.

Our proposed WATC and WAPR schemes work effectively by taking the workload of each node into consideration. Our experimental results show that our workload-aware tree construction scheme is superior to the existing ETC approach in terms of load-balancing and energy efficiency. Similarly, the effectiveness of our workload-aware path repairing scheme is proved to be better than the existing path repairing scheme. Our novel DTDR scheme resulted in much better energy efficiency when compared to that of single tree approach

Our novel lossy data compression scheme IR-LDCA facilitates much better control over the compression ratio and precision than the existing K-RLE scheme. Our experiments proved the same. Our IR-LDCA approach gives attractive results on data sets with higher temporal correlation. The effectiveness of our ad-hoc query processing scheme proved to be better than that of the existing approach in terms of exploiting cached data and containment. Our novel query processing scheme is intended for optimizing execution of ad-hoc queries. All the required experiments are conducted on custom-built simulator with necessary fine tunings.

Future Directions

Now, we brief on scope for future work in this area. Our proposed tree construction and maintenance schemes assume that the sensing and query processing loads assigned to the nodes in the network are almost same (uniform distribution). Investigations are necessary for introducing schemes for WSNs where nodes have varying loads. We also believe that there is enough scope for new path repairing schemes that take into account other reasons for node failure like- physical damage, hardware failure etc. Similarly, failure situations where a collection of nodes in a geographical location become unavailable need to be investigated. In our proposed query processing scheme, while identifying the containment we have considered the conditions set w.r.t., time attribute. We understand that further possibility of exploiting containment w.r.t., conditions set on other attributes may be explored for better effectiveness. We also believe that there is sufficient scope for new query processing techniques that optimize the processing costs of continuous queries.

REFERENCES

[1] Garcia-Hernandez, Carlos F., Pablo H. Ibarguengoytia-Gonzalez, Joaquin Garcia-Hernandez, and Jesus A. Perez-Diaz. "Wireless sensor networks and applications: a survey." In *International Journal of Computer Science and Network Security* 7, no. 3 (2007): 264-273.

[2] Raghavendra, Cauligi S., Krishna M. Sivalingam, and Taieb Znati, eds. “*Wireless sensor networks*” *Springer*, 2004.

[3] Lewis, Franck L. "*Wireless sensor networks: Smart environments: technologies, protocols, and applications.* “ 2004 11-46.

[4] Akyildiz, Ian F., and Mehmet Can Vuran. “*Wireless sensor networks*” Vol.4. John Wiley &Sons, 2010.

[5] Lee, Sang Hyuk, Soobin Lee, Heecheol Song, and Hwang Soo Lee. "Wireless sensor network design for tactical military applications: remote large-scale environments." In *International Conference on Military Communications*, 2009. MILCOM 2009. IEEE, pp. 1-7. IEEE, 2009.

[6] Akylidiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirici, E.: “The survey on sensor etworks” In IEEE *Communications Magazine* 40(8), 114-120, 2002.

[7] Arampatzis, Th, John Lygeros, and S. Manesis. "A survey of applications of wireless sensors and wireless sensor networks." *In Proceedings of the* IEEE *International Symposium on, Mediterranean Conference on Control and Automation Intelligent Control, 2005.*, pp. 719-724. IEEE, 2005.

[8] Madden S.R., Franklin M.J., Hellerstein J.M., HongW., “TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks”, In USENIX OSDI, 2002.

[9] Sathe, Saket, Thanasis G. Papaioannou, Hoyoung Jeung and Karl aberer. “A survey of model-based sensor data acquisition and management.” *In Managing and Mining Sensor Data*, pp 9-50. Springer US, 2013.

[10] P. Xia, P.K. Chrysanthis, A. Labrinidis, “Similarity-Aware Query Processing in Sensor Networks”, In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems,* Island of Rhodes, Greece, April 25-26, pp.8, 2006.

[11] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, “Compressing historical information in sensor networks”, In *Proceedings of the International Conference on Management of data* (SIGMOD’04), Paris, France, June 13-18, pp.527-538, 2004.

[12] Lu, Gang, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks." In *Proceedings of 18th International Parallel and Distributed Processing Symposium, 2004.* pp. 224. IEEE, 2004.

[13] Koubaa, Anis, Mario Alves, and Eduardo Tovar. "Modeling and worst-case dimensioning of cluster-tree wireless sensor networks." In *27th* IEEE *International Real-Time Systems Symposium, 2006. RTSS'06.*, pp. 412-421. IEEE, 2006.

[14] Gengzhong, Zheng, and Liu Qiumei. "A survey on topology control in wireless sensor networks." In *Second International Conference on Future Networks,*

2010. pp. 376-380. IEEE, 2010.

[15] Kawano, Ryouhei, and Toshiaki Miyazaki. "Distributed data aggregation in multi-sink sensor networks using a graph coloring algorithm." In *22nd International Conference on Advanced Information Networking and Applications-Workshops, 2008*. pp. 934-940. IEEE, 2008.

[16] Ke, Wang, Wang Liqiang, Cai Shiyu, and Qu Song. "An energy-saving algorithm of WSN based on Gabriel graph." In *5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09*, pp. 1-4. IEEE, 2009.

[17] Vuran, Mehmet C., Özgür B. Akan, and Ian F. Akyildiz. "Spatio-temporal correlation: theory and applications for wireless sensor networks." *Computer Networks* 45, no. 3 (2004): 245-259.

[18] Vuran, Mehmet C., and Ian F. Akyildiz. "Spatial correlation-based collaborative medium access control in wireless sensor networks." In IEEE/ACM *Transactions on Networking*, 14, no. 2 (2006): 316-329.

[19] Szewczyk, Robert, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. "An analysis of a large scale habitat monitoring application." In *Proceedings of the 2nd International Conference on Embedded networked sensor systems*, pp. 214-226. ACM, 2004.

[20] Cao, Xianghui, Jiming Chen, Yan Zhang, and Youxian Sun. "Development of an integrated wireless sensor network micro-environmental monitoring system."In *ISA transactions* 47, no. 3 (2008): 247-255.

[21] Handcock, Rebecca N., Dave L. Swain, Greg J. Bishop-Hurley, Kym P. Patison, Tim Wark, Philip Valencia, Peter Corke, and Christopher J. O'Neill. "Monitoring animal behaviour and environmental interactions using wireless sensor networks, GPS collars and satellite remote sensing." *Sensors* 9, no. 5 (2009): 3586-3603.

[22] Xu, Ning, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. "A wireless sensor network for structural monitoring." In *Proceedings of the 2nd International Conference on Embedded networked sensor systems*, pp. 13-24. ACM, 2004.

[23] Kusuma, J., Doherty, L., and Ramchandran, K.: Distributed compression for sensor networks, In *Proceedings of the International Conference on Image Processing* (ICIP'01) Vol. 1 IEEE, Thessaloniki, Greece, 82-85, 2001.

[24] Akyildiz, Ian F., Mehmet C., Vuran, and Ozgur B. Akan. "On exploiting spatial and temporal correlation in Wireless Sensor networks." In *Proceedings of WiOpt*. Vol.4. 2004.

[25] R. Maiocchi, B. Pernici, "Temporal Data Management Systems: A Comparative View", In IEEE *Transactions on Knowledge and Data Engineering*, December, Vol.3, No.4, pp.504-524, 1991.

[26] Kimura, Naoto, and Shahram Latifi. " A survey on data compression in wireless sensor networks." *In International Conference on Information Technology: Coding and Computing, 2005.* ITCC 2005. Vol. 2, pp. 8-13. IEEE, 2005.

[27] Pattem, Sundeep, Bhaskar Krishnamachari, and Ramesh Govindan. “The impact of spatial correlation on routing with compression in wireless sensor networks.” *ACM Transactions on Sensor Networks (TOSN)* 4, no. 4(2008):24.

[28] Gavrilovska, Liljana, ed. “*Application and multidisciplinary aspects of wireless sensor networks*”. Springer, 2011.

[29] Kim, Sukun, et al. "Health monitoring of civil infrastructures using wireless sensor networks." In 6th *International Symposium on Information Processing in Sensor Networks,* IPSN 2007. IEEE, 2007.

[30] Hu, Fei, and Xiaojun Cao. “*Wireless sensor networks: principles and practice”*. Auerbach Publications, 2010.

[31] Andreou, Panayiotis, et al. "Optimized query routing trees for wireless sensor networks." In *Information Systems* 36.2 (2011): 267-291.

[32] Madden, Samuel, et al. "Supporting aggregate queries over ad-hoc wireless sensor networks." In *Proceedings of 4th IEEE Workshop on Mobile Computing Systems and Applications*, 2002. IEEE, 2002.

[33] Fasolo, Elena, et al. "In-network aggregation techniques for wireless sensor networks: a survey." In *Wireless Communications*, IEEE 14.2 (2007): 70-87.

[34] Y. Yao, J.E. Gehrke, “The cougar approach to in-network query processing in sensor networks”, In ACM SIGMOD Record (SIGMOD’02), September, Vol.31, No.3, pp.9-18, 2002.

[35] C-C. Shen, C. Srisathapornphat C., C. Jaikaeo, “Sensor information networking architecture and applications”, In IEEE *Personal Communications,* Vol.8,

No.5, pp.52-59, August, 2001.

[36] A. Ollero, M. Bernard, M.L. Civita, L. van Hoesel, P.J. Marron, J. Lepley, E. de Andres, “AWARE: Platform for Autonomous self-deploying and operation of Wireless sensor actuator networks cooperating with unmanned AeRial vehiclEs”, In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics* (2007), Rome, Italy, September 27-29, pp.1-6.

[37] S. Li, S.H. Son, J.A. Stankovic, “Event detection services using data service middleware in distributed sensor networks”, In *Proceedings of the 2nd International conference on Information processing in sensor networks,* Palo Alto, CA, USA, pp.502-517.

[38] T. Liu, M. Martonosi, “Impala: a middleware system for managing autonomic, parallel sensor systems”, SIGPLAN Notices, Vol.38, No.10, pp.107-118, June, 2003.

[39] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, “The Design of an Acquisitional Query Processor for Sensor Networks”, In *Proceedings of the 2003 ACM SIGMOD International conference on Management of data,* San Diego, California, USA, June 9-12, pp.491-502, 2003.

[40] Andreou, Panayiotis, Andreas Pamboris, Demetrios Zeinalipour-Yazti, Panos K. Chrysanthis, and George Samaras. "ETC: energy-driven tree construction in wireless sensor networks." In *10th International Conference on Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09.*, pp. 513-518. IEEE, 2009.

[41] Di Felice, Paolino, Massimo Ianni, and Luigi Pomante. "A spatial extension of TinyDB for wireless sensor networks. "In IEEE *Symposium on Computers and*

Communications, 2008. ISCC 2008, pp. 1076-1082. IEEE, 2008.

[42] Madden, Samuel, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. "The design of an acquisitional query processor for sensor networks." In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of data*, pp. 491-502. ACM, 2003.

[43] Zhang, Chun, Jeffrey Naughton, David DeWitt, Qiong Luo, and Guy Lohman. "On supporting containment queries in relational database management systems." In *ACM SIGMOD Record*, vol. 30, no. 2, pp. 425-436. ACM, 2001.

[44] Dai, Hui, and Richard Han. "A node-centric load balancing algorithm for wireless sensor networks." In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, vol. 1, pp. 548-552. IEEE, 2003.

[45] Chakraborty Suchetana, S. Nandi, and S. Karmakar. "A Tree-Based Local Repairing Approach for Increasing Lifetime of Query Driven WSN." In IEEE *14th International Conference on Computational Science and Engineering (CSE),* 2011, pp. 475-482. IEEE, 2011.

[46] Nanda, Arabinda, Amiya Kumar Rath, and Saroj Kumar Rout. "Node Sensing & Dynamic Discovering Routes for Wireless Sensor Networks." In *International Journal of Computer Science and Information Security,* Vol. 7, No. 3, March 2010.

[47] Tharini, C., and P. Vanaja Ranjan. "Design of modified adaptive Huffman data compression algorithm for wireless sensor network." In *Journal of Computer Science* 5.6 (2009): 466.

[48] Yan-li, Zhou, et al. "Improved LZW algorithm of lossless data compression for WSN." In 3rd IEEE *International Conference on Computer Science and Information Technology*", Vol. 4. IEEE, 2010.

[49] Pamba, E., Chichi, C., Guyennet, H. and Friedt, J.-M.: "K-RLE: A new data compression algorithm for wireless sensor network", In *Proceedings of the 3rd International Conference on Sensor Technologies and Applications*, pp. 502–507, Athens/Glyfada, Greece, 2009.

[50] Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., and Estrin, D.: "Lightweight temporal compression of microclimate datasets", In 1st *IEEE Workshop on Embedded networked Sensors (EmNetS-I)*, Tampa, Florida, USA, November 2004.

[51] García Villalba, Luis Javier, Ana Lucila Sandoval Orozco, Alicia Triviño Cabrera, and Claudia Jacy Barenco Abbas. "Routing protocols in wireless sensor networks." *Sensors* 9, no. 11 (2009): 8399-8421.

[52] Zhang, Ying, and Markus Fromherz. "A robust and efficient flooding-based routing for wireless sensor networks." *Journal of Interconnection Networks* 7, no. 04 (2006): 549-568.

[53] Akyildiz, Ian F., James I. Pelech, and Bülent Yener. "Virtual topology based routing protocol for multihop dynamic wireless networks." *Wireless Networks* 7, no. 4 (2001): 413-424.

[54] Hu, Fei, and Xiaojun Cao. *Wireless sensor networks: principles and practice*. Auerbach Publications, 2010.

[55] Saleem, Muhammad, Gianni A. Di Caro, and Muddassar Farooq. "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions." *Information Sciences* 181.20 (2011): 4597-4624.

[56] Blazevic, Ljubica, J-Y. Le Boudec, and Silvia Giordano. "A location-based routing method for mobile ad hoc networks." In *IEEE Transactions on Mobile Computing, Vol.* 4, no. 2 (2005): 97-110.

[57] Savvides, Andreas, Chih-Chieh Han, and Mani B. Strivastava. "Dynamic fine-grained localization in ad-hoc networks of sensors." In *Proceedings of the 7th Annual International Conference on Mobile computing and networking*, pp. 166-179. ACM, 2001.

[58] Hussain, Sajid, and Obidul Islam. "An energy efficient spanning tree based multi-hop routing in wireless sensor networks." In *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pp. 4383-4388. IEEE, 2007.

[59] Kershenbaum, A., and R. Van Slyke. "Computing minimum spanning trees efficiently." In *Proceedings of the ACM annual conference, Volume 1*, pp. 518-527. ACM, 1972.

[60] Chatzimilioudis, Georgios, Demetrios Zeinalipour-Yazti and Dimitrios Gunopulos, "Minimum-hot-spot query trees for wireless sensor networks", In *Proceedings of the 9th* ACM *International workshop on Data Engineering for wireless and mobile access*, pp. 33-40. ACM, 2010.

[61] Neng-Chung Wang, Shih-Chien Chang, Yung-Fa Huang, Ching-Mu Chen, Young-Long Chen, "An efficient data aggregation scheme for grid-based wireless sensor networks", In *International Conference of Wireless*

Communications and Mobile Computing, IWCMC'10, June 28– July 2, 2010, Caen, France.

[62] Marcelloni, Francesco, and Massimo Vecchio. "A simple algorithm for data compression in wireless sensor networks" *Communications Letters*, IEEE 12.6 (2008): 411-413.

[63] Zhang, Huan, Xiao-ping Fan, Shao-qiang Liu, and Zhi Zhong. "Design and realization of improved LZW algorithm for wireless sensor networks." In *International Conference on Information Science and Technology (ICIST), 2011*, pp. 671-675. IEEE, 2011.

[64] Salomon, David. *Data compression: the complete reference*. Springer, 2004.

[65] Yang, Chi, and Rachel Cardell-Oliver. "An efficient approach using domain knowledge for evaluating aggregate queries in WSN." In *5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009*, pp. 427-432. IEEE, 2009.

[66] Benzing, Andreas, Boris Koldehofe, Marco Volz, and Kurt Rothermel. "Multilevel predictions for the aggregation of data in global sensor networks." In IEEE/ACM *14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2010*, pp. 169-178. IEEE, 2010.

[67] **Kayiram Kavitha**, Polsani Rajashree Rao, Sreeja Tummala, Gajarla Vasavi and Dr.R.Gururaj, "Dual Tree Data Routing Scheme for Wireless Sensor Networks," *In 3rd International Conference on Advances in Information Technology and Mobile Communication,* AIM -2013, 26-27, April 2013, Bangalore, India.

[68] **Kavitha, Kayiram**, Cheemakurthi Ravi Teja, and R. Gururaj. "Workload-aware tree construction algorithm for wireless sensor networks." In *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks*, Vol. 4, no. 1, March 2012.

[69] **Kayiram Kavitha**, Cheemakurthi Ravi Teja, R.Gururaj, "Workload-aware path repairing scheme for Wireless Sensor Networks," In 7th IEEE *International Conference on Industrial and Information Systems* (ICIIS-2012), Indian Institute of Technology Madras (IITM), Chennai, India, 06-09 August 2012.

[70] **Kayiram Kavitha**, Dhruv Sharma, Rahul Surana and R Gururaj. Article: "Induced Redundancy based Lossy Data Compression Algorithm." In *International Journal of Computer Applications* Vol. 62(16) Page no. 16-21, January 2013, Published by Foundation of Computer Science, New York, USA.

[71] Intel Berkeley Research lab, http://db.csail.mit.edu/labdata/labdata.html

[72] **Kayiram Kavitha**, Vinod Pachipulusu, Sreeja Thummala, Dr.R.Gururaj. Article: "Energy Efficient Query Processing for WSN based on Data Caching and Query Containment", In *International Journal of Computer Applications*, Vol. 89(19) Page no. 4-8, March 2014, Published by Foundation of Computer Science, New York, USA.

APPENDIX-1

Simulator Specifications

To prove the efficacy of our proposed data management schemes, we have conducted a good number of experiments on custom-built simulator. In this appendix, we present the details about the simulator w.r.t., its implementation, features, and set-up for various experiments.

We have developed a custom-built simulator, which is implemented using Java Technology. Our simulator is meant for Windows platform and is console-based. This simulator allows us to define the geographical range of the network along with the number of nodes. We can also define the transmission range of each node. As the amount of power consumed is the basic criteria to assess the effectiveness of our techniques, we have designed our simulator in such a way that the power allotted to each node can also be defined. The deployment of nodes using our simulator can be either random or predefined. The input to the simulator is the location of nodes specified by their *x* & *y* co-ordinates. The radio communication range of each node is set as 3m. Each sensor node is initialized with 1J of energy. The network is given some query workload so that the energy of the nodes gets depleted. For the simulation purpose we assume the connectivity to be constant and there is no loss of packets in transit.

First, we present the details about simulation of WSN, and then we explain the set-up required for conducting experiments pertaining to the data routing, data compression, and query processing schemes.

Sensor Nodes and Network Simulation

Now, we present the description about simulation of a sensor node, and the process of connecting multiple sensor nodes to form a WSN. To simulate a sensor node, we input the (*x*, *y*) co-ordinates of the node along with its *nodeID*. We also define values for parameters like-transmission range, battery power, level no. etc., are defined. We define the cost involved with certain operations in the network w.r.t., power. For instance, we define the initial battery power available at each node to be 100 units (1 Joule). Further, we define that the amount of power consumed at a node in transmitting one packet is 0.08 units. Similarly, the energy consumption per packet received is 0.03units. In our implementation, each sensor node is an instance of the class *Sensor*, which encapsulates all necessary functionalities and properties of a sensor node. The sensor nodes can be initialized by supplying the required parameters mentioned above. Once all the nodes are initialized, the process of formation of the required network begins. The important functions used are- i) function to find the distance between sensor nodes, ii) function to establish routing paths etc. If the distance between two nodes is less than or equal to 3 units, then they are able to communicate with each other and become neighbors. Likewise, we apply our algorithm to construct data routing trees. To simulate sensor data readings at a sensor node, we use *random* function. The sensor readings are generated at desired frequencies using *time* functions available in Java. A sensor node generates data packets from the sensed data according to the pre-defined packet structure with necessary header information like- routing path etc. Each node transmits packets according to the predefined routing information.

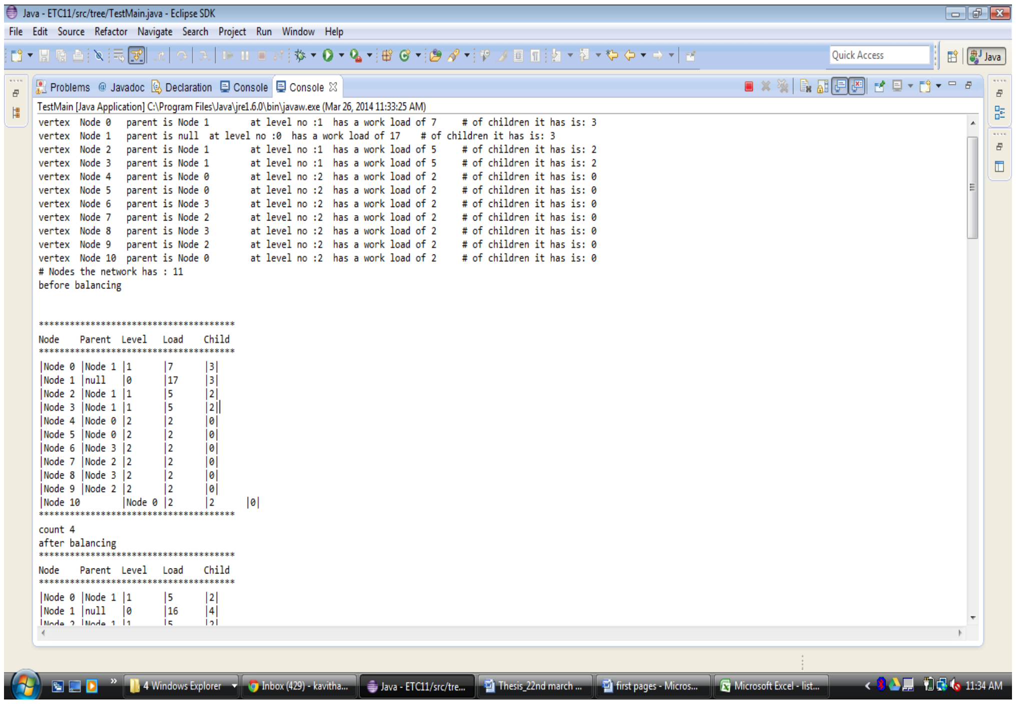
We have also incorporated certain checks to be performed before each transmission like- the current power level at the sender and receiver to complete the transmission successfully etc. Our simulator also includes functions to compute efficient routing paths (shortest path, minimal hop-count, etc.). In our simulator, all nodes get uniform workload. We also have facilities to compute values for various parameters like- residual power, network lifetime, number of packets transmitted etc., which are used in result analysis.

Screenshots

Now, we present a set of screenshots captured during experimentation.

A. WSN initialization

The screenshot given in Figure A.1 shows the details of simulation run for a network with 11 nodes. A tree structure with 11 nodes (table form) is shown along with level no., parent, child node(s) and workload, for each node. This is a sample of initial tree which is considered for workload balancing.



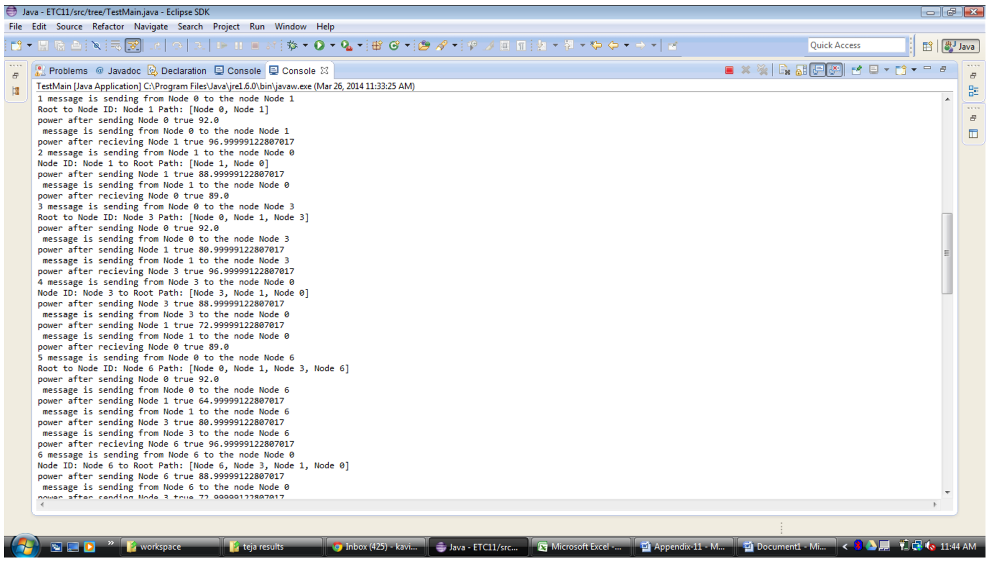
```
TestMain [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 11:33:25 AM)
vertex  Node 0   parent is Node 1       at level no :1  has a work load of 7     # of children it has is: 3
vertex  Node 1   parent is null  at level no :0  has a work load of 17    # of children it has is: 3
vertex  Node 2   parent is Node 1       at level no :1  has a work load of 5     # of children it has is: 2
vertex  Node 3   parent is Node 1       at level no :1  has a work load of 5     # of children it has is: 2
vertex  Node 4   parent is Node 0       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 5   parent is Node 0       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 6   parent is Node 3       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 7   parent is Node 2       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 8   parent is Node 3       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 9   parent is Node 2       at level no :2  has a work load of 2     # of children it has is: 0
vertex  Node 10  parent is Node 0       at level no :2  has a work load of 2     # of children it has is: 0
# Nodes the network has : 11
before balancing


**************************************
Node    Parent  Level   Load    Child
**************************************
|Node 0 |Node 1 |1      |7      |3|
|Node 1 |null   |0      |17     |3|
|Node 2 |Node 1 |1      |5      |2|
|Node 3 |Node 1 |1      |5      |2||
|Node 4 |Node 0 |2      |2      |0|
|Node 5 |Node 0 |2      |2      |0|
|Node 6 |Node 3 |2      |2      |0|
|Node 7 |Node 2 |2      |2      |0|
|Node 8 |Node 3 |2      |2      |0|
|Node 9 |Node 2 |2      |2      |0|
|Node 10        |Node 0 |2      |2      |0|
**************************************
count 4
after balancing
**************************************
Node    Parent  Level   Load    Child
**************************************
|Node 0 |Node 1 |1      |5      |2|
|Node 1 |null   |0      |16     |4|
```

Figure A.1. Screenshot showing the initial tree set-up in a WSN.

B. Load-balancing and recovery

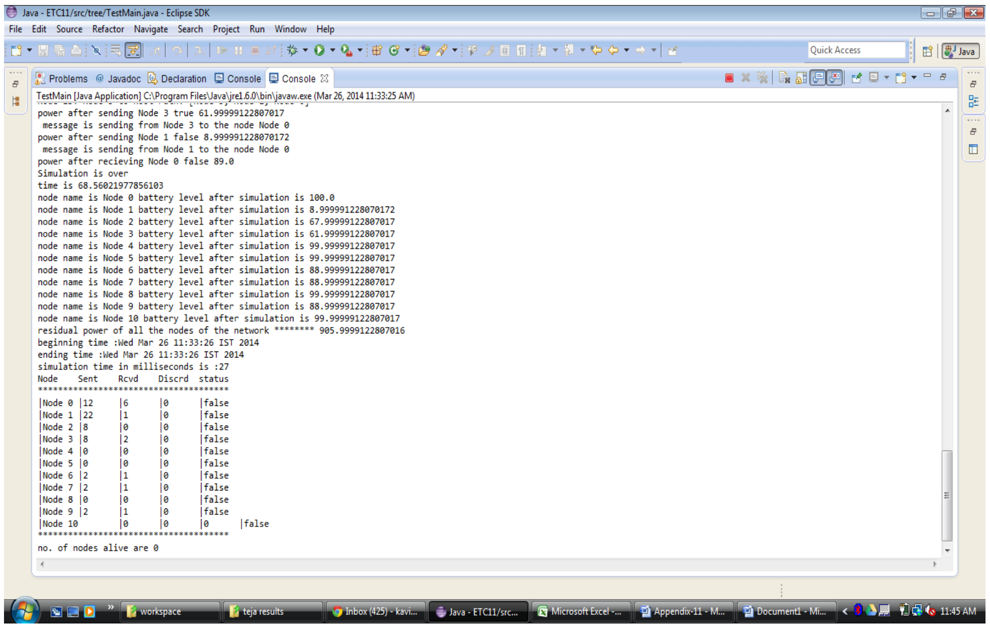
The Figure A.2 shows a screenshot which is the result of running a sequence of packet transmissions at various nodes for ETC implementation. The information displayed includes the hop details, power levels etc.



```
TestMain [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 11:33:25 AM)
1 message is sending from Node 0 to the node Node 1
Root to Node ID: Node 1 Path: [Node 0, Node 1]
power after sending Node 0 true 92.0
 message is sending from Node 0 to the node Node 1
power after recieving Node 1 true 96.99999122807017
2 message is sending from Node 1 to the node Node 0
Node ID: Node 1 to Root Path: [Node 1, Node 0]
power after sending Node 1 true 88.99999122807017
 message is sending from Node 1 to the node Node 0
power after recieving Node 0 true 89.0
3 message is sending from Node 0 to the node Node 3
Root to Node ID: Node 3 Path: [Node 0, Node 1, Node 3]
power after sending Node 0 true 92.0
 message is sending from Node 0 to the node Node 3
power after sending Node 1 true 80.99999122807017
 message is sending from Node 1 to the node Node 3
power after recieving Node 3 true 96.99999122807017
4 message is sending from Node 3 to the node Node 0
Node ID: Node 3 to Root Path: [Node 3, Node 1, Node 0]
power after sending Node 3 true 88.99999122807017
 message is sending from Node 3 to the node Node 0
power after sending Node 1 true 72.99999122807017
 message is sending from Node 1 to the node Node 0
power after recieving Node 0 true 89.0
5 message is sending from Node 0 to the node Node 6
Root to Node ID: Node 6 Path: [Node 0, Node 1, Node 3, Node 6]
power after sending Node 0 true 92.0
 message is sending from Node 0 to the node Node 6
power after sending Node 1 true 64.99999122807017
 message is sending from Node 1 to the node Node 6
power after sending Node 3 true 80.99999122807017
 message is sending from Node 3 to the node Node 6
power after recieving Node 6 true 96.99999122807017
6 message is sending from Node 6 to the node Node 0
Node ID: Node 6 to Root Path: [Node 6, Node 3, Node 1, Node 0]
power after sending Node 6 true 88.99999122807017
 message is sending from Node 6 to the node Node 0
power after sending Node 3 true 72.99999122807017
```

Figure A.2 Screenshot showing the packet transmission in a WSN.

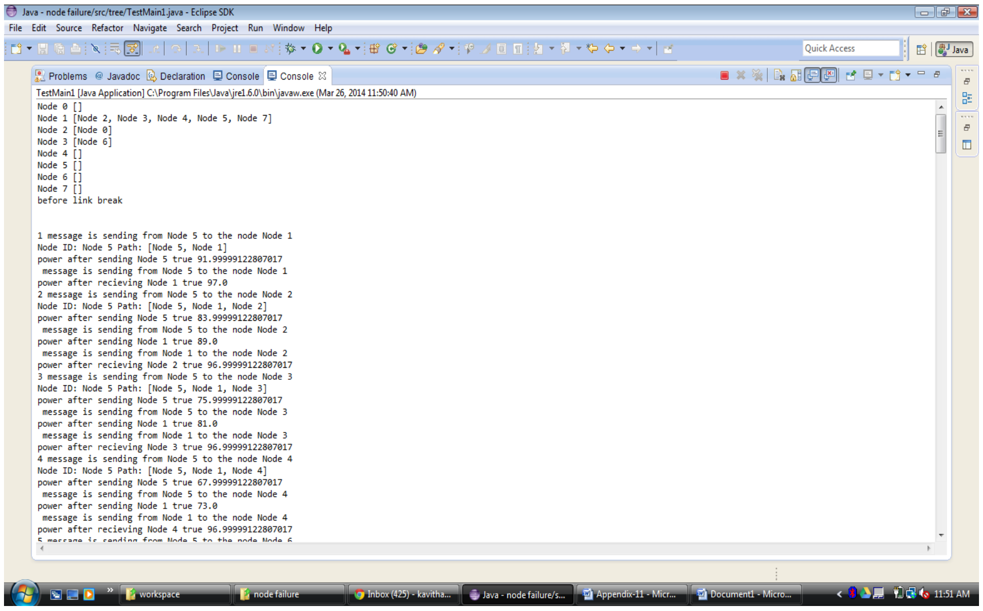
The screenshot given in Figure A.3 is the result of running our workload-aware tree construction algorithm. We compare the effectiveness of our algorithm with that of ETC algorithm. This screenshot also shows the residual power, simulation time, status of nodes after load balancing.



```
TestMain [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 11:33:25 AM)
power after sending Node 3 true 61.99999122807017
 message is sending from Node 3 to the node Node 0
power after sending Node 1 false 8.999991228070172
 message is sending from Node 1 to the node Node 0
power after recieving Node 0 false 89.0
Simulation is over
time is 68.56021977856103
node name is Node 0 battery level after simulation is 100.0
node name is Node 1 battery level after simulation is 8.999991228070172
node name is Node 2 battery level after simulation is 67.99999122807017
node name is Node 3 battery level after simulation is 61.99999122807017
node name is Node 4 battery level after simulation is 99.99999122807017
node name is Node 5 battery level after simulation is 99.99999122807017
node name is Node 6 battery level after simulation is 88.99999122807017
node name is Node 7 battery level after simulation is 88.99999122807017
node name is Node 8 battery level after simulation is 99.99999122807017
node name is Node 9 battery level after simulation is 88.99999122807017
node name is Node 10 battery level after simulation is 99.99999122807017
residual power of all the nodes of the network ******** 905.9999122807016
beginning time :Wed Mar 26 11:33:26 IST 2014
ending time :Wed Mar 26 11:33:26 IST 2014
simulation time in milliseconds is :27
Node    Sent    Rcvd    Discrd  status
**************************************
|Node 0 |12     |6      |0      |false
|Node 1 |22     |1      |0      |false
|Node 2 |8      |0      |0      |false
|Node 3 |8      |2      |0      |false
|Node 4 |0      |0      |0      |false
|Node 5 |0      |0      |0      |false
|Node 6 |2      |1      |0      |false
|Node 7 |2      |1      |0      |false
|Node 8 |0      |0      |0      |false
|Node 9 |2      |1      |0      |false
|Node 10        |0      |0      |0      |false
**************************************
no. of nodes alive are 0
```

Figure A.3 Screenshot showing the final workload-aware tree.

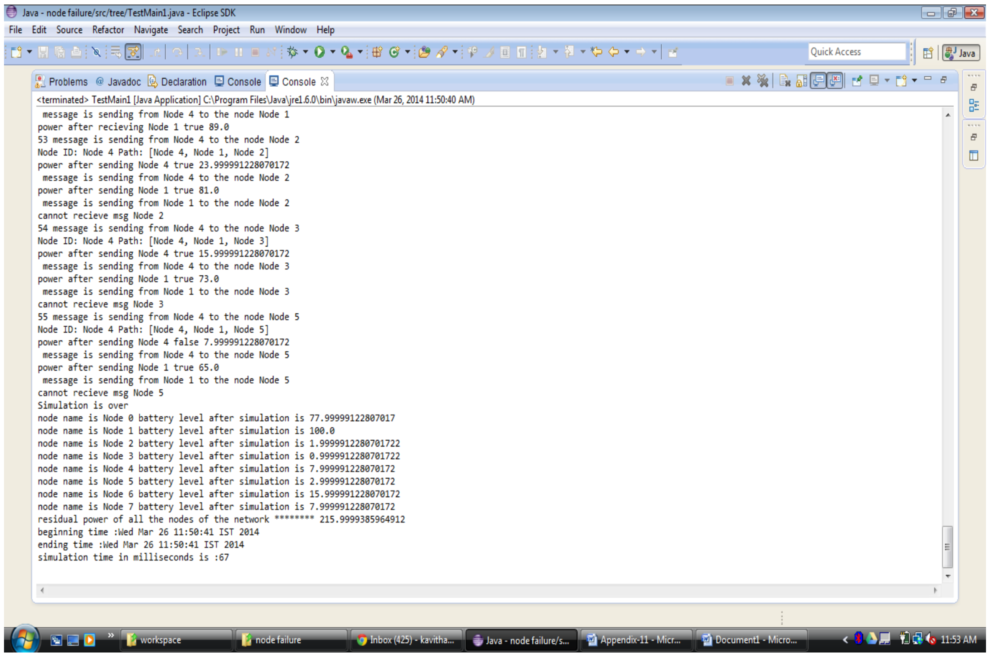
The Figure A.4 shows the screenshot of the simulator when a node failure occurs in a tree-based WSN. The packet routing process is also visible in the screenshot.


Java - node failure/src/tree/TestMain1.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Console
workspace
node failure
Inbox (425) - kavitha...
Java - node failure/s...
Appendix-11 - Micr...
Document1 - Micro...
11:51 AM

```
TestMain1 [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 11:50:40 AM)
Node 0 []
Node 1 [Node 2, Node 3, Node 4, Node 5, Node 7]
Node 2 [Node 0]
Node 3 [Node 6]
Node 4 []
Node 5 []
Node 6 []
Node 7 []
before link break


1 message is sending from Node 5 to the node Node 1
Node ID: Node 5 Path: [Node 5, Node 1]
power after sending Node 5 true 91.99999122807017
 message is sending from Node 5 to the node Node 1
power after recieving Node 1 true 97.0
2 message is sending from Node 5 to the node Node 2
Node ID: Node 5 Path: [Node 5, Node 1, Node 2]
power after sending Node 5 true 83.99999122807017
 message is sending from Node 5 to the node Node 2
power after sending Node 1 true 89.0
 message is sending from Node 1 to the node Node 2
power after recieving Node 2 true 96.99999122807017
3 message is sending from Node 5 to the node Node 3
Node ID: Node 5 Path: [Node 5, Node 1, Node 3]
power after sending Node 5 true 75.99999122807017
 message is sending from Node 5 to the node Node 3
power after sending Node 1 true 81.0
 message is sending from Node 1 to the node Node 3
power after recieving Node 3 true 96.99999122807017
4 message is sending from Node 5 to the node Node 4
Node ID: Node 5 Path: [Node 5, Node 1, Node 4]
power after sending Node 5 true 67.99999122807017
 message is sending from Node 5 to the node Node 4
power after sending Node 1 true 73.0
 message is sending from Node 1 to the node Node 4
power after recieving Node 4 true 96.99999122807017
5 message is sending from Node 5 to the node Node 6
```

Figure A.4 Screenshot of the node failure and recovery scheme


Java - node failure/src/tree/TestMain1.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Console
workspace
node failure
Inbox (425) - kavitha...
Java - node failure/s...
Appendix-11 - Micr...
Document1 - Micro...
11:53 AM

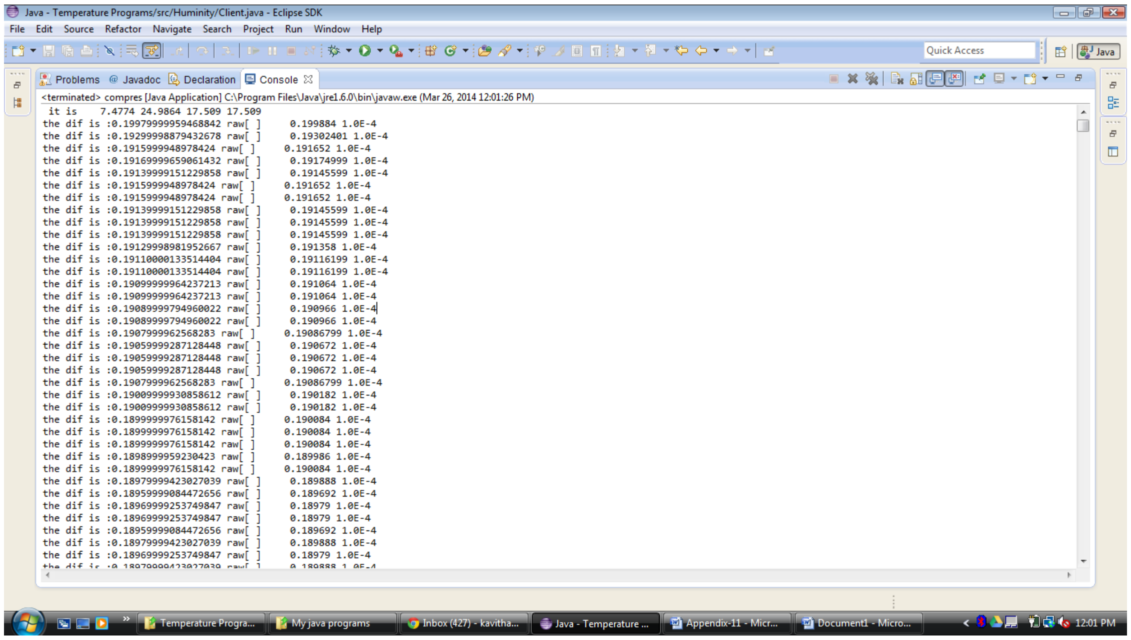
```
<terminated> TestMain1 [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 11:50:40 AM)
 message is sending from Node 4 to the node Node 1
power after recieving Node 1 true 89.0
53 message is sending from Node 4 to the node Node 2
Node ID: Node 4 Path: [Node 4, Node 1, Node 2]
power after sending Node 4 true 23.999991228070172
 message is sending from Node 4 to the node Node 2
power after sending Node 1 true 81.0
 message is sending from Node 1 to the node Node 2
cannot recieve msg Node 2
54 message is sending from Node 4 to the node Node 3
Node ID: Node 4 Path: [Node 4, Node 1, Node 3]
power after sending Node 4 true 15.999991228070172
 message is sending from Node 4 to the node Node 3
power after sending Node 1 true 73.0
 message is sending from Node 1 to the node Node 3
cannot recieve msg Node 3
55 message is sending from Node 4 to the node Node 5
Node ID: Node 4 Path: [Node 4, Node 1, Node 5]
power after sending Node 4 false 7.999991228070172
 message is sending from Node 4 to the node Node 5
power after sending Node 1 true 65.0
 message is sending from Node 1 to the node Node 5
cannot recieve msg Node 5
Simulation is over
node name is Node 0 battery level after simulation is 77.99999122807017
node name is Node 1 battery level after simulation is 100.0
node name is Node 2 battery level after simulation is 1.9999912280701722
node name is Node 3 battery level after simulation is 0.9999912280701722
node name is Node 4 battery level after simulation is 7.999991228070172
node name is Node 5 battery level after simulation is 2.999991228070172
node name is Node 6 battery level after simulation is 15.999991228070172
node name is Node 7 battery level after simulation is 7.999991228070172
residual power of all the nodes of the network ******** 215.9999385964912
beginning time :Wed Mar 26 11:50:41 IST 2014
ending time :Wed Mar 26 11:50:41 IST 2014
simulation time in milliseconds is :67
```

Figure A.5 Screenshot of the WAPR scheme in the simulator.

The screenshot in Figure A.5 shows the simulation during the Workload-Aware Path Repairing (WAPR) process. The metrics like- residual power, simulation time etc., computed after the simulation is also visible.

C. Simulation run for Data Compression

The Figure A.6 shows the screenshot of the simulator during the experimentation done to prove the effectiveness of the data compression scheme. This screenshot shows the first step of the data compression scheme.

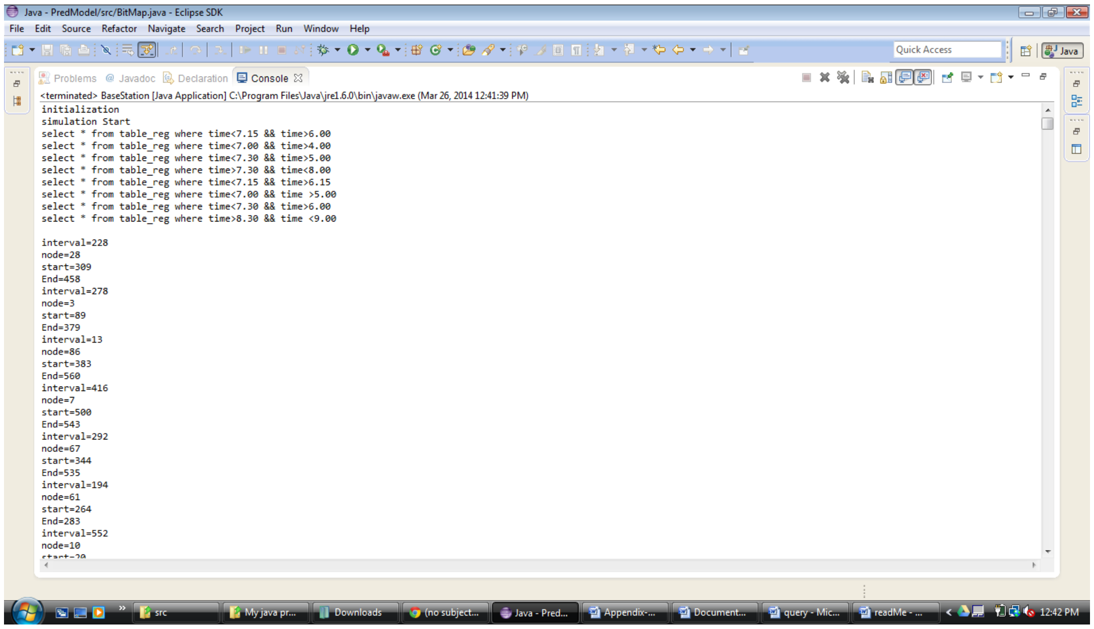

Java - Temperature Programs/src/Huminity/Client.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Problems
Javadoc
Declaration
Console

```
<terminated> compres [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 12:01:26 PM)
 it is    7.4774 24.9864 17.509 17.509
the dif is :0.19979999959468842 raw[ ]    0.199884 1.0E-4
the dif is :0.19299998879432678 raw[ ]    0.19302401 1.0E-4
the dif is :0.1915999948978424 raw[ ]    0.191652 1.0E-4
the dif is :0.19169999659061432 raw[ ]    0.19174999 1.0E-4
the dif is :0.19139999151229858 raw[ ]    0.19145599 1.0E-4
the dif is :0.1915999948978424 raw[ ]    0.191652 1.0E-4
the dif is :0.1915999948978424 raw[ ]    0.191652 1.0E-4
the dif is :0.19139999151229858 raw[ ]    0.19145599 1.0E-4
the dif is :0.19139999151229858 raw[ ]    0.19145599 1.0E-4
the dif is :0.19139999151229858 raw[ ]    0.19145599 1.0E-4
the dif is :0.19129998981952667 raw[ ]    0.191358 1.0E-4
the dif is :0.19110000133514404 raw[ ]    0.19116199 1.0E-4
the dif is :0.19110000133514404 raw[ ]    0.19116199 1.0E-4
the dif is :0.19099999964237213 raw[ ]    0.191064 1.0E-4
the dif is :0.19099999964237213 raw[ ]    0.191064 1.0E-4
the dif is :0.19089999794960022 raw[ ]    0.190966 1.0E-4
the dif is :0.19089999794960022 raw[ ]    0.190966 1.0E-4
the dif is :0.1907999962568283 raw[ ]    0.19086799 1.0E-4
the dif is :0.19059999287128448 raw[ ]    0.190672 1.0E-4
the dif is :0.19059999287128448 raw[ ]    0.190672 1.0E-4
the dif is :0.19059999287128448 raw[ ]    0.190672 1.0E-4
the dif is :0.1907999962568283 raw[ ]    0.19086799 1.0E-4
the dif is :0.19009999930858612 raw[ ]    0.190182 1.0E-4
the dif is :0.19009999930858612 raw[ ]    0.190182 1.0E-4
the dif is :0.1899999976158142 raw[ ]    0.190084 1.0E-4
the dif is :0.1899999976158142 raw[ ]    0.190084 1.0E-4
the dif is :0.1899999976158142 raw[ ]    0.190084 1.0E-4
the dif is :0.1898999959230423 raw[ ]    0.189986 1.0E-4
the dif is :0.1899999976158142 raw[ ]    0.190084 1.0E-4
the dif is :0.18979999423027039 raw[ ]    0.189888 1.0E-4
the dif is :0.18959999084472656 raw[ ]    0.189692 1.0E-4
the dif is :0.18969999253749847 raw[ ]    0.18979 1.0E-4
the dif is :0.18969999253749847 raw[ ]    0.18979 1.0E-4
the dif is :0.18959999084472656 raw[ ]    0.189692 1.0E-4
the dif is :0.18979999423027039 raw[ ]    0.189888 1.0E-4
the dif is :0.18969999253749847 raw[ ]    0.18979 1.0E-4
the dif is :0.18979999423027039 raw[ ]    0.189888 1.0E-4
```

Figure A.6. Screenshot of the simulator during data compression scheme

D. Query processing

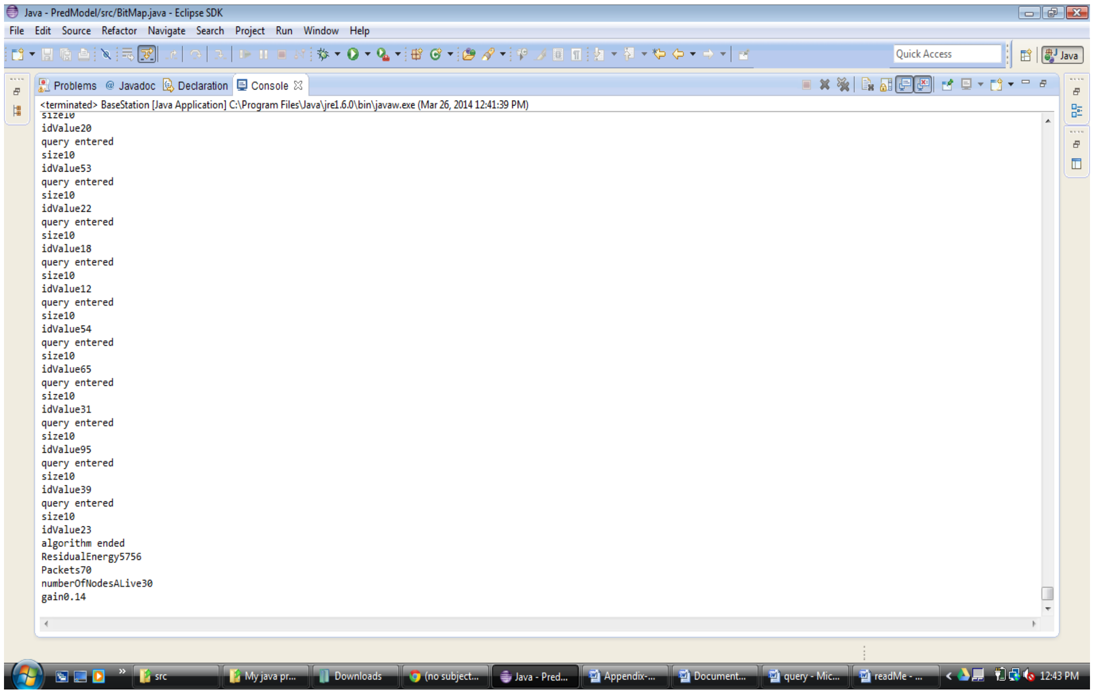
The screenshot in Figure A.7 shows the simulation of our query processing scheme in WSN. The initial set of queries for a sample batch is shown in Figure A.7. Next the Figure A.8 shows the final metrics like- residual power, number of nodes alive etc.,



```
<terminated> BaseStation [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 12:41:39 PM)
initialization
simulation Start
select * from table_reg where time<7.15 && time>6.00
select * from table_reg where time<7.00 && time>4.00
select * from table_reg where time<7.30 && time>5.00
select * from table_reg where time>7.30 && time<8.00
select * from table_reg where time<7.15 && time>6.15
select * from table_reg where time<7.00 && time >5.00
select * from table_reg where time<7.30 && time>6.00
select * from table_reg where time>8.30 && time <9.00

interval=228
node=28
start=309
End=458
interval=278
node=3
start=89
End=379
interval=13
node=86
start=383
End=560
interval=416
node=7
start=500
End=543
interval=292
node=67
start=344
End=535
interval=194
node=61
start=264
End=283
interval=552
node=10
```

Figure A.7. Screenshot showing sample queries during the simulation



```
<terminated> BaseStation [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Mar 26, 2014 12:41:39 PM)
idValue20
query entered
size10
idValue53
query entered
size10
idValue22
query entered
size10
idValue18
query entered
size10
idValue12
query entered
size10
idValue54
query entered
size10
idValue65
query entered
size10
idValue31
query entered
size10
idValue95
query entered
size10
idValue39
query entered
size10
idValue23
algorithm ended
ResidualEnergy5756
Packets70
numberOfNodesALive30
gain0.14
```

Figure A.8. Screenshot showing the final results after query processing.

LIST OF PUBLICATIONS

Journal Publications

- **Kavitha, Kayiram**, Cheemakurthi Ravi Teja, and R. Gururaj. "Workload-aware tree construction algorithm for wireless sensor networks." In *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks (GRAPH-HOC),* Vol. 4, no. 1, March 2012.

  http://airccse.org/journal/graphhoc/papers/0312jgraph01.pdf

- **Kayiram Kavitha**, Dhruv Sharma, Rahul Surana and R Gururaj. Article: Induced Redundancy based Lossy Data Compression Algorithm. International Journal of Computer Applications, Vol. 62 no. 16, Page No. 16-21, January 2013, Published by Foundation of Computer Science, New York, USA.

  http://www.ijcaonline.org/archives/volume62/number16/10164-4928

- **Kayiram Kavitha**, Vinod Pachipulusu, Sreeja Thummala, R.Gururaj. Article: Energy Efficient Query Processing for WSN based on Data Caching and Query Containment, International Journal of Computer Applications, Vol. 89, no. 19, Page no. 4-8, March 2014, Published by Foundation of Computer Science, New York, USA.

  http://www.ijcaonline.org/archives/volume89/number19/15737-4528

Conference Publications

- **Kayiram Kavitha**, Cheemakurthi Ravi Teja, R.Gururaj, Workload-aware path repairing scheme for Wireless Sensor Networks, Seventh IEEE International Conference on Industrial and Information Systems (ICIIS-2012), Indian Institute of Technology Madras (IITM), Chennai, India, 06-09 August 2012.

  IEEE Xplore : DOI: 10.1109/ICIInfS.2012.6304800

- **Kayiram Kavitha**, Polsani Rajashree Rao, Sreeja Tummala, Gajarla Vasavi and Dr.R.Gururaj, Dual Tree Data Routing Scheme for Wireless Sensor Networks, Third International Conference on Advances in Information Technology and Mobile Communication, AIM -2013, 26-27, April 2013, Bangalore, India.

BRIEF BIOGRAPHY OF THE CANDIDATE

Kayiram Kavitha, received her Master’s degree in Computer Science from JNTU-Kakinada. Prior to this she was working as Lecturer in Institute of Aeronautical Engineering for 4 years. Her Bachelor’s degree is from JNTU-Hyderabad. She has a consistent academic record. She gained 10 years experience teaching UG students. She is good in mentoring students in their UG projects. Her research interests include Wireless Sensor Networks, Mobile Computing etc.

BRIEF BIOGRAPHY OF THE SUPERVISOR

Dr.R.Gururaj is Assistant Professor and Head of the Department of Computer Science at BITS-Pilani, Hyderabad Campus. He has been with BITS-Pilani, Hyderabad Campus since 2007. He received his Ph.D. in Computer Science from Indian Institute of Technology, Madras (IIT-M). His thesis is on “Content delivery through XML message brokering”. During his graduate studies, he had the opportunity to spend several years doing research in the Computer Science department at IIT-M. Prior to joining IIT-M, he spent three years in the IT industry. He has more than 15 years of teaching and research experience. He received his Masters degree in Computer Science from Birla Institute of Technology (BIT), Mesra, Ranchi. He is the author of several research articles. He has many international publications to his credit. His research interests include Database Technologies, Object Technologies, Information Systems, Web Applications, UML, Data Integration, Wireless Sensor Networks etc.