Query Optimization in Wireless Sensor Networks

Thesis Report

Submitted in the partial fulfilment of the requirements for the award of degree of

Master in Engineering

in

Software Engineering

Under the Supervision of

Dr. Anil Kumar Verma

Submitted by:

Name: Vipin Kumar

(Roll No.:800831016)

Computer Science and Engineering Department

Thapar University

Patiala-147004

Certificate

I hereby certify that the work which is being presented in thesis entitled " Query Optimization in Wireless Sensor Networks", in partial fulfilment of the requirements for the award of degree of Master of Engineering in Software Engineering Submitted in Computer Science and Engineering Department of Thapar University ,Patiala , is an authentic record of my own work carried out under supervision of Dr. Anil Kumar Verma and refers other researchers' works which are duly listed in reference section. The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Vipin

(Vipin Kumar)

This is certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Anil Kr. Verma 02/07/10.

(Dr. Anil Kumar Verma)
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned By

(Dr. Rajesh Bhatia)
Head
Computer Science and Engineering Department
Thapar University,
Patiala

RK Sharma 8/7/10

(Dr.R.K. Sharma)
Dean(Academic Affairs)
Thapar University,
Patiala

Acknowledgment

I wish to express my deep gratitude to Dr. Anil Kumar Verma, Assistant Professor, Computer Science and Engineering Department, TU, Patiala for providing his uncanny guidance and support throughout the thesis.

I am thankful to Dr. Rajesh Bhatia, Head, Computer Science and Engineering Department, TU, Patiala for the motivation and inspiration that triggered me for the thesis work. I would also like to thank all the staff members who were always there at the need of the hour and provided with the help and facilities, which I required for the completion of the thesis.

Finally, my special thanks to authors whose works have been consulted and quoted in this work.

Vipin Kumar

800831016

M.E. (Software Engineering)-2nd Yr.

Computer Science and Engineering Department

Thapar University

Patiala-147004

Abstract

A wireless sensor network (WSN) is a wireless network consisting of distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. A WSN Consists of ten to thousand of sensor nodes that communicate through wireless channels for information sharing and cooperative processing. We attach the sensor nodes to specific items to be monitored, covering an area with locomotive sensor nodes. The sensor node is also known as mote. In WSN all the sensor nodes send the data to a sink node. We consider the sink node as base station node, which is more powerful in comparison to other nodes in case of data processing, storage etc. Queries are also submitted at the base station.

In this our main focus is on base station queries optimization. When a new query is submitted to the base station, we check whether the new query can be evaluated using the result of currently running queries. If it is possible then we rewrites a new query using currently running queries at the base station without injecting it into the sensor network. Thus optimizing the query processing in Wireless Sensor Network

Keywords: wireless sensor network, query processing, query optimization.

Table of Contents

List of Figures

List of Tables

CHAPTER - 1

INTRODUCTION

1.1. Motivation

With the emergence of embedded system technology laptops, cell phones, PDAs, GPS devices, intelligent computing devices are become popular day by day. This made the things cheaper, more mobile, more distributed and more pervasive in daily life. According to Moore's Law [18] the number of transistor on a cost effective chip, computation power and storage capacity increases doubles every year or two. Now, it is possible to construct a wallet size embedded system with the equivalent capability of a PC. Such embedded systems can be supported with scaled down Windows or Linux operating systems.

Wireless Sensor Network is the network is composed of many autonomous and compact devices called sensor nodes. The objective of this network is to collect data. Resources of sensor nodes are storage capacity and its computation power and communication system. Sensor networks are composed of thousands of resource constrained sensor nodes and also some resourced base stations are there. All nodes in a network communicate with each other via wireless communication. Moreover, the energy required to transmit a message is about twice as great as the energy needed to receive the same message. Query Processing is the method to get the sensor readings from sensor. The query optimization at base station is really crucial in terms network lifetime: e.g. As the number of queries that are running in the sensor network increases, the energy consumption of the sensor network increases. This is because each sensor node has to send its readings towards the base station. Thus in order to reduce the energy consumption of the sensor network, it is important to reduce the number of monitoring queries injected into the network.

1.2. State of the Art

Typically, a wireless sensor node consists of processing capability, may contain multiple type of memory (program, data and flash memories), have a RF transceiver, have a power source. The components of sensor node are integrated on a single or multiple boards, and packaged in a few cubic inches. With state-of-the-art, low-power circuit and networking technologies, a sensor node powered by 2 AA batteries can last for up to three years. A WSN usually consists of tens to thousands of such nodes that communicate through wireless channels for information sharing and cooperative processing. In a typical scenario, users can retrieve information of interest from a WSN by injecting queries and gathering results from the base stations or sink nodes, which behave as an interface between users and the network.

1.3. Thesis Outline

Chapter 1 describes Wireless Sensor Network in general in terms of motivation and then follows by state of art and finally the whole thesis outline. Chapter 2, we discuss the background information relating to WSN and query processing in SQL and in WSN. Chapter 3, we discuss the problem statement and objectives. Chapter 4 discusses the installation of tools and the simulation environment. Chapter 5 describes the results, evaluates the performance and analysis and finally Chapter 6 summarizes the conclusions drawn in the thesis along with future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1. Wireless Sensor Network

Wireless sensor network (WSN) consist of tiny sensor nodes that are deployed in spatially distributed terrain. Each sensor node have limited amount of processing but when coordinated with the information from another node they have the ability to measure the given physical environment in greater detail or to execute a task with complex functions. The sensor nodes can be mobile or static. According to DARPA the definition of WSN is:

"A sensor network is a deployment of massive number of smalls, inexpensive, self powered devices that can sense, compute and communicate with other devices for the purpose of gathering local information to make decisions about a physical environment."[1]

2.2. Evolution of Sensor Network

As with many technologies, defence applications have been a driver for research and development in sensor networks. During the Cold War [2], the Sound Surveillance System (SOSUS), a system of acoustic sensors (hydrophones) on the ocean bottom was deployed at strategic locations to detect and track quiet Soviet submarines. Over the years, other more sophisticated acoustic networks have been developed for submarine surveillance. SOSUS is now used by the National Oceanographic and Atmospheric Administration (NOAA) for monitoring events in the ocean, e.g., seismic and animal activity. Also during the Cold War, networks of air defence radars were developed and deployed to defend the continental United States and Canada. This air defence system has evolved over the years to include aerostats as sensors and Airborne Warning and Control System (AWACS) planes and is also used for drug interdiction.

Modern research on sensor networks started around 1980 with the Distributed Sensor Networks (DSN) program at the Defence Advanced Research Projects Agency (DARPA). These included acoustic sensors communication (a high- level protocols that link processes working on a common application in a resource-sharing network), processing techniques, algorithms (including self-location algorithms for sensors) and distributed software (dynamically modifiable distributed systems and language design).

Recent advances in computing and communication have caused a significant shift in sensor network research and brought it closer to achieving the original vision. Small and inexpensive sensors based upon micro-electro-mechanical system (MEMS) technology, wireless networking, and inexpensive low-power processors allow the deployment of wireless ad hoc networks for various applications. Thus, the program developed with new networking techniques is suitable for highly dynamic ad-hoc environments.

2.3. Wireless Sensor Network Model:

2.3.1. Components of Wireless Sensor Network

- *Sensor Field:* A sensor field can be considered as the area in which the nodes are placed.
- *Sensor Nodes:* Sensors nodes are the heart of the network. They are in charge of collecting data and routing this information back to a base station.
- *Task Manager:* The task manager also known as base station is a centralised point of control within the network, which extracts information from the network and disseminates control information back into the network. It also serves as a gateway to other networks, a powerful data processing and storage centre and an access point for a human interface. The base station is either a laptop or a workstation.

Sensor nodes consist of sensing, computing, actuation and power components. These components are integrated on a single or multiple boards and packaged in few cubic inches. The sensor nodes communicate through wireless channel for information

sharing and cooperative processing. There are two types of nodes in a typical network. These are:

i. **Sensor Node**: This can be a static node or a mobile node (moving freely) to monitors the physical environment. Once it detects the physical target, it generates a data packet and sends it to the sink node via the wireless channel.

ii. **Sink Node (Gateway Node)**: This node collects all data packets from sensor nodes. Users use these collected data to analyze their targets.

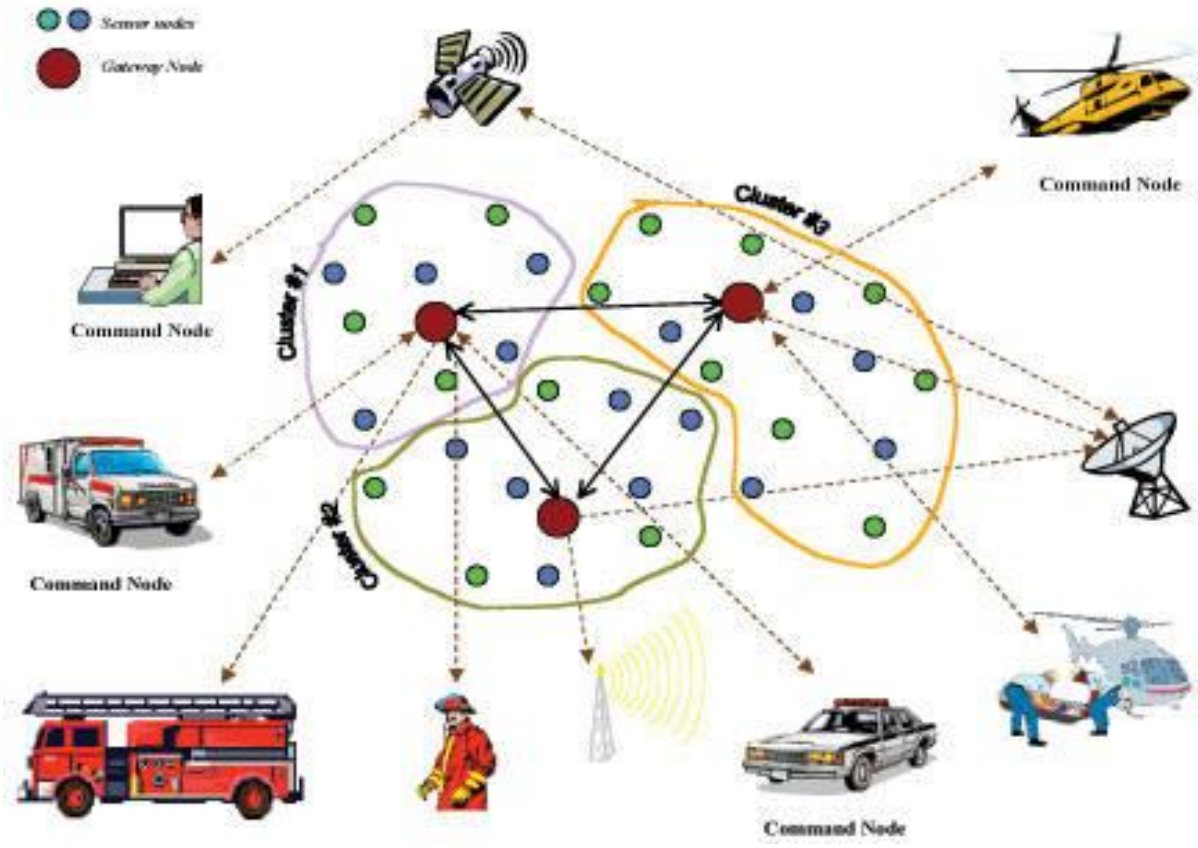

Fig.2.1. A Wireless Sensor Network [2]

2.3.2. Components of Sensor Node

- Communication unit: a radio transceiver or other wireless communications device.
- Sensing unit.
- A processing unit: small microcontroller.
- A power unit: an energy source usually a battery(2 AA)
  In WSN the capability of each sensor node is limited, the average power of the entire network is sufficient for the required project.

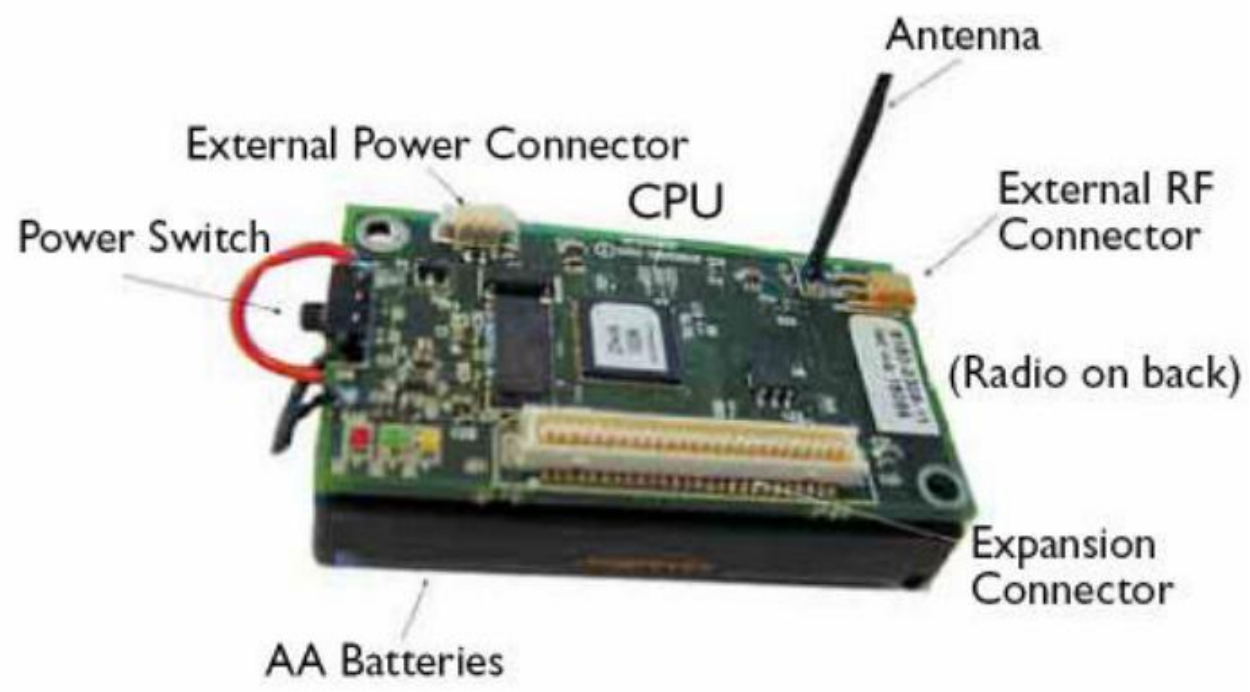

Fig.2.2. A typical sensor node [2]

2.3.3 Commercially available Wireless Sensor Systems

Many commercially available wireless communications nodes are available including Lynx Technologies, and various Bluetooth kits, including the Casira devices from Cambridge Silicon Radio, CSR.

Crossbow Berkeley Motes [3] may be the most versatile wireless sensor network devices on the market for prototyping purposes.

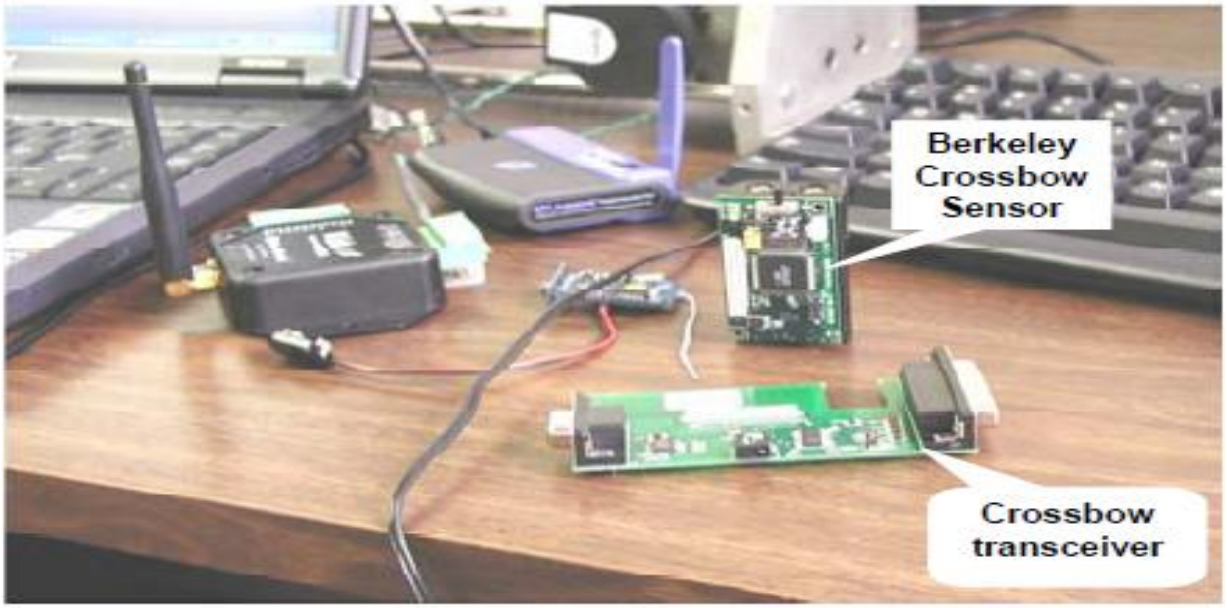

Fig.2.3. Berkeley CrossBow Motes [3]

Crossbow makes three Mote processor radio module families–

- MICA (MPR300)
- MICA2 (MPR400)
- MICA2-DOT (MPR500)

Nodes come with five sensors installed- Temperature, Light, Acoustic (Microphone), Acceleration/Seismic, and Magnetic. These are especially suitable for surveillance networks for personnel and vehicles. Different sensors can be installed if desired. Low power and small physical size enable placement virtually anywhere. Since all sensor nodes in a network can act as base stations, the network can self configure and has multi-hop routing capabilities. The operating frequency is ISM band, either 916 Mhz or 433 MHz, with a data rate of 40 Kbits/sec. and a range of 30 ft to 100 ft. Each node has a low power microcontroller processor with speed of 4MHz, a flash memory with 128 Kbytes, and SRAM and EEPROM of 4K bytes each. The operating system is Tiny-OS, a tiny micro-threading distributed operating system developed by UC Berkeley, with a NES-C (Nested C) source code language (similar to C). Installation of these devices requires a great deal of programming.

Microstrain's X-Link Measurement System may be the easiest

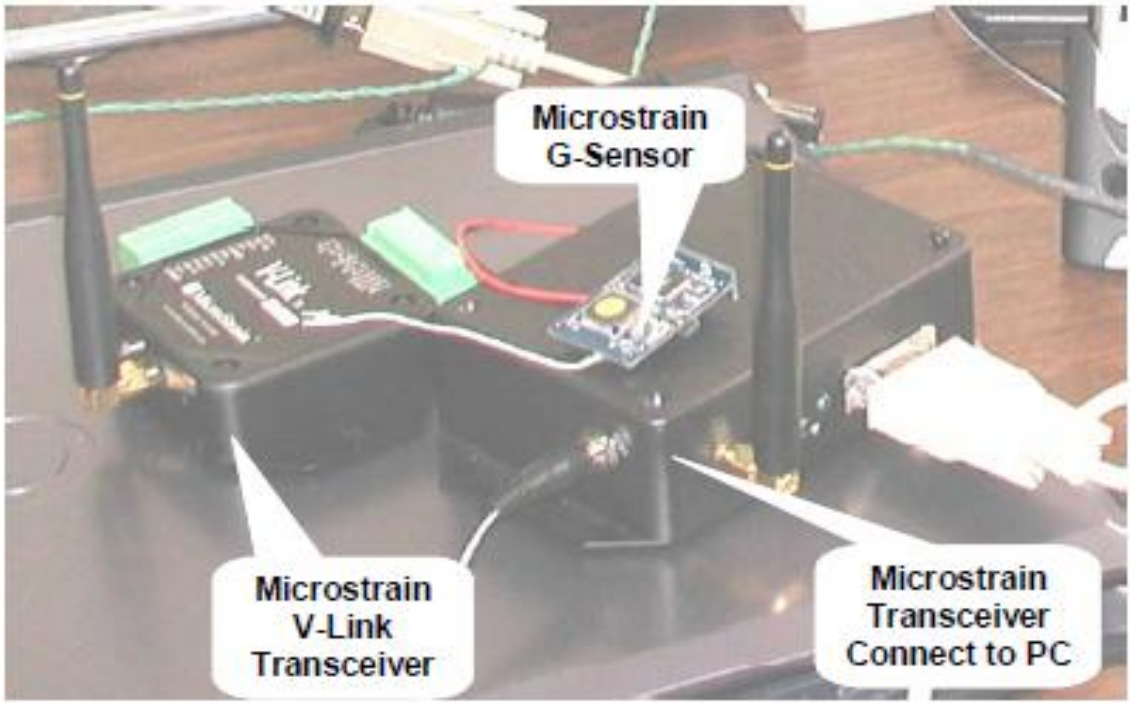

Fig.2.4. Microstrain Wireless Sensors [3]

system to get up and running to program. The frequency used in this sensors 916 MHz, which lies in the US license-free ISM band. The sensor nodes are multi-channel, with a maximum of 8 sensors supported by a single wireless node. There are three types of sensor nodes – S-link (strain gauge), G-link (accelerometer), and V-link (supports any sensors generating voltage differences). The sensor nodes have a pre-programmed EPROM, so a great deal of programming by the user is not needed. Onboard data storage is 2MB. Sensor nodes use a 3.6-volt lithium ion internal battery (9V rechargeable external battery is supported). A single receiver (Base Station) addresses multiple nodes. Each node has a unique 16-bit address, so a maximum of 2^{16} nodes can be addressed. The RF link between Base Station and nodes is bi-directional and the sensor nodes have a programmable data logging sample rate. The RF link has a 30 meter range with a 19200 baud rate. The baud rate on the serial RS-232 link between the Base Station and a terminal PC is 38400.

2.4. Characteristics of Wireless Sensor Network:

WSNs have some unique characteristics [4]. These are:

- Sensor nodes are small-scale devices with volumes approaching a cubic millimetre in the near future. Such small devices are very limited in the amount of energy they can store or harvest from the environment.
- Nodes are subject to failures due to depleted batteries or, more generally, due to environmental influences. Limited size and energy also typically means restricted resources (CPU performance, memory, wireless communication bandwidth and range).
- Node mobility, node failures, and environmental obstructions cause a high degree of dynamics in WSN. This includes frequent network topology changes and network partitions. Despite partitions, however, mobile nodes can transport information across partitions by physically moving between them.
- The resulting paths of information flow might have unbounded delays and are potentially unidirectional. Communication failures are also a typical problem of WSN.

2.5. Application of Wireless Sensor Network [4]

- **Military:** monitoring inimical forces, battle field surveillance, Nuclear, biological and chemical attack detection. A large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines. When the sensors detect the event being monitored (heat, pressure, sound, light, electro-magnetic field, vibration, etc.), the event needs to be reported to one of the base stations, which can take appropriate action (e.g., send a message on the internet or to a satellite).
- **Environmental:** for detecting the fire in forest and in homes, big organizations and for flood detection.

- **Health application:** for the remotely monitoring the physiological data, tracking and monitoring doctors and patient inside a hospital etc.

- **Home Application:** for the Home automation-i.e. security purpose, automated meter reading etc.

- **Commercial Applications:** environmental control in industries and office buildings, Vehicle tracking and detection, traffic flow surveillance. An example of such an application on a production line is shown in Figure 2.5. In this application, typically ten or more sensors are used to measure gaps where rubber seals are to be placed. Previously, the use of wired sensors was too cumbersome to be implemented in a production line environment. The use of wireless sensors in this application is enabling, allowing a measurement to be made that was not previously practical

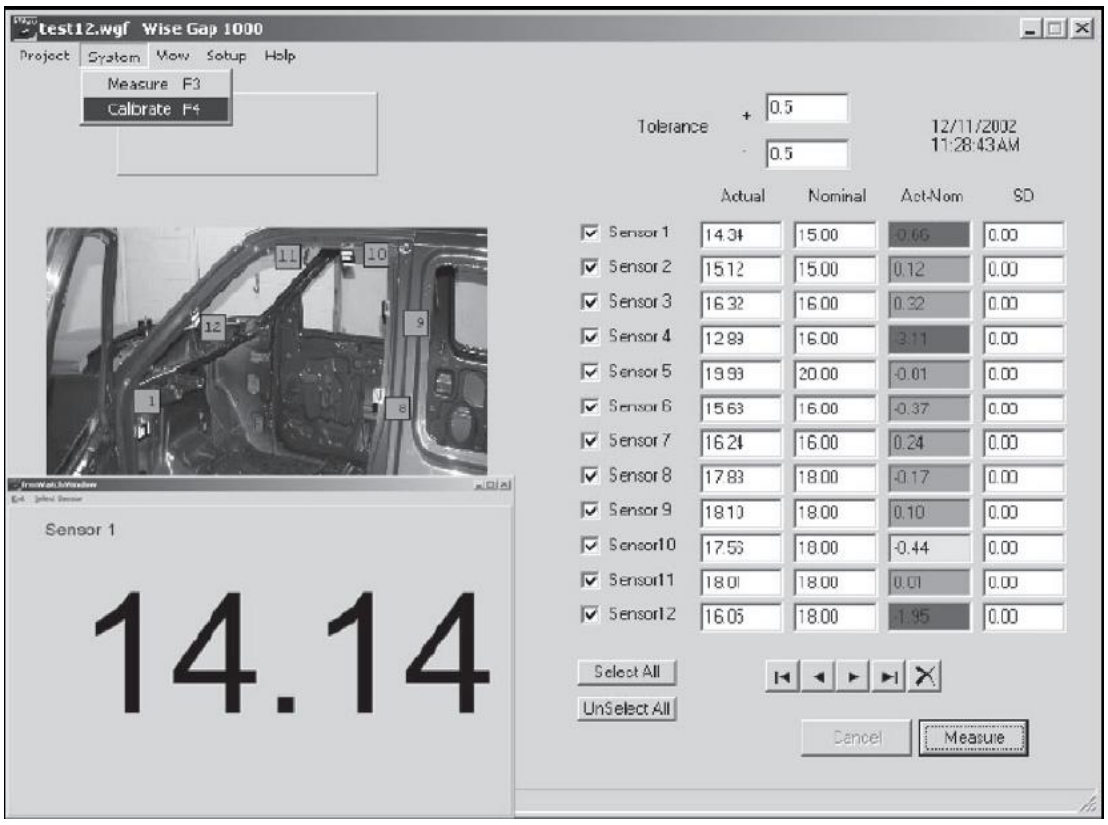

Fig 2.5. [18] Industrial application of wireless sensors

- **Habitat Monitoring:** Researchers in the Life Sciences are becoming increasingly concerned about the potential impacts of human presence in monitoring plants and animals in field conditions. At best it is possible that

chronic human disturbance may distort results by changing behavioural patterns or distributions, while at worst anthropogenic disturbance can seriously reduce or even destroy sensitive populations by increasing stress, reducing breeding success. Sensor networks represent a significant advance over traditional invasive methods of monitoring. Sensors can be deployed prior to the onset of the breeding season or other sensitive period.

2.6. Query Processing [19] in SQL.

Query based data retrieval is one of the most commonly used mechanism to find an information. Query performance in relational database systems is dependent not only on the database structure, but also on the way in which the query is optimized. SQL query processing requires that the DBMS identify and execute a strategy for retrieving the results of the query. The SQL query determines what data is to be found, but does not define the method by which the data manager searches the database. Hence, query optimization is necessary for high-level relational queries and provides an opportunity for the DBMS to systematically evaluate alternative query execution strategies and to choose an optimal strategy.

2.6.1 Components of Query Processing

1) Standard form of the query-SQL
2) Query Execution Methodology-the steps that are followed in order to execute the query.
3) Query Optimization: How do we determine a good execution plan.

Basic Steps in Query Processing

1. **Convert to a standard starting point**: operator graph (internal representation of query in order to various operation are executed) and relation algebraic expression would be considered as the starting point. The SQL query statement is first parsed into its constituent parts. The basic SELECT statement is formed from the three clauses SELECT, FROM, and WHERE. These parts identify the various tables and columns that participate in the data selection process. The WHERE clause is used to

determine the order and precedence of the various attribute comparisons through a conditional expression.

2. **Transform the query**: in this phase we give the preference to the expressions which increase the performance of query.

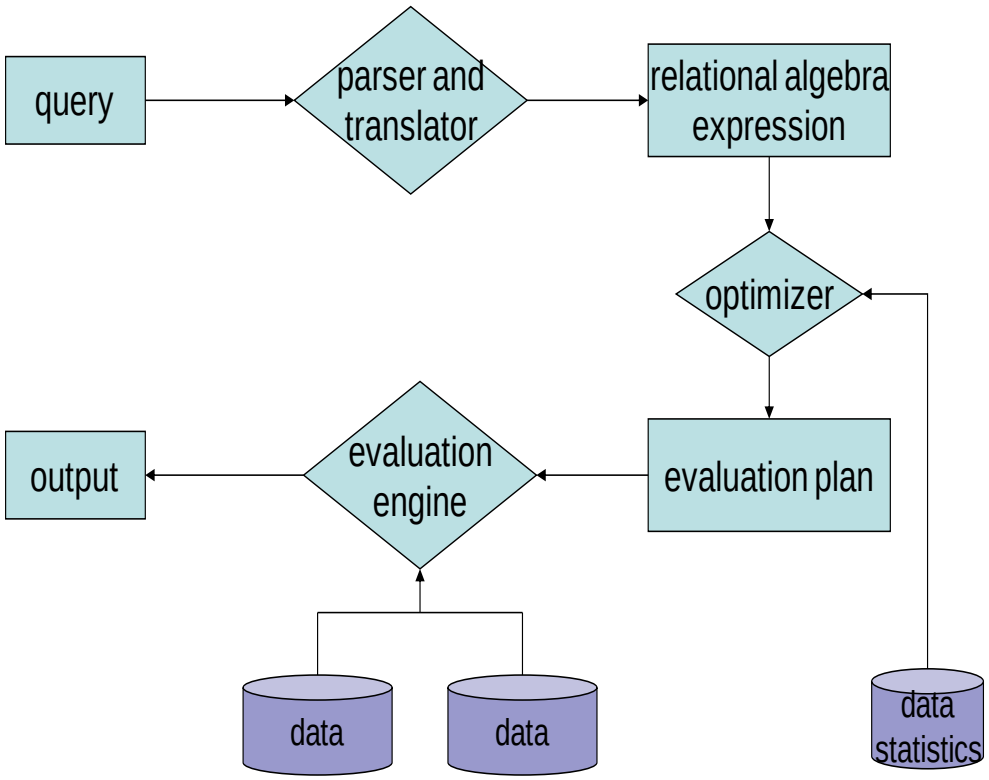

Fig.2.6. Query Processing in SQL [20]

3. **Simplify the query:** simplified by minimizing redundancy from expressions
4. **Prepare alternate excess plan:** these shows the sequence of operations and cost associated with them. Execution plan is established the query code is generated. Various techniques such as memory management, disk caching and parallel query execution can be used to improve the query performance.

2.7. Query Processing in WSN[6]:

- The query statement is parsed into its constituent parts. As in SQL, queries in TinyDB consists of blocks SELECT, FROM, WHERE, GROUP BY, HAVING, DURATION, EVERY to support selection, join, projection, aggregation and grouping.

- In TinyDB [6] architecture user input queries at the sink node in a simple, SQL like language that describes data they want to collect and how they wish to combine, transform and summarize it. TinyDB SQL variant differs from traditional SQL in that its queries are continuous and periodic. i.e. "Temperature from sensor from 2^{nd} floor where sensor_id=222 every 5 second". Each period in which result is produces is an **EPOCH.**

- As in traditional database systems, queries describe a logical set of data that the user is interested in, but don't describe the actual algorithms and software modules or operators the system uses to collect the answer set. Typically the system can choose from several plans and operator ordering for any given logical query. For example to find the average temperature of the 2^{nd} floor sensors, the system might collect readings from every sensor, then filter the list for 2^{nd} floor sensors and compute the average. Alternate method is that it might request that only 2^{nd} floor sensors provide their temperature readings and then average the values it collects. In WSN the 2^{nd} plan is better, because it requires only sensors from the 2^{nd} floor to collect temperature.

- At a very high level, query optimizer work by enumerating a set of possible plans, assigning a cost to each plan based on estimated costs of each of the operators and choosing the lowest cost plan.

A wireless sensor network is a collection of sensor nodes distributed over a geographical area to monitor physical conditions, such as temperature, light, humidity, or pressure. In many applications, a sensor network is viewed as a relational database which provides query services on sensor data [6] [7]. For example, in TinyDB [6], sensor readings produced by sensor nodes logically belong

to a table called *sensors*. Each row of the *sensors* table contains the node id and readings of a sensor node, as illustrated in Table 1.

Nodeid	**Temp.**	**Light**
1	28	221
2	36	237
3	45	246

Table 1: The sensor Table

When a user submit a continuous monitoring query like("SELECT Nodeid, Temp. FROM sensor WHERE Temp>35 SAMPLE PERIOD 4S") to the base station, the query is injected into the sensor network and then each sensor node periodically sends its readings towards the base station if its reading satisfies the condition of the query. It is continuous monitoring query that returns the node ids and temperature readings of sensor nodes every 4 seconds whose temperature readings are greater than 35^{o}C. The sample period of a query specifies the time interval between the re-evaluation of the query. For each sample period, the base station collects data relevant to the query from the sensor network and returns the final results to the user.

CHAPTER 3

PROBLEM STATEMENT & OBJECTIVE

3.1. Problem Statement

Nodes in wireless sensor network can be failed after destroying by the physical environment and their batteries can be depleted. So deployment of Sensor nodes in a monitoring area is a continuous process. Due to large number of nodes, nodes are operated unattended after deployment .Once the complete wireless sensor network has established it can be used to fulfil the required task.

Sometimes in some physical environment it is not possible to replace or charge the sensor nodes. Therefore it is desirable to design communication network protocol such that energy source is used efficiently to maximize the lifetime of network. Another important issue is data delivery time by sensor nodes is also important especially in case of battlefield or medical or security monitoring system where minimum delay is desirable.

Communication cost is much higher than computation cost, so the sensor nodes should compute the efficient path and it should not deplete the batteries of sensor node so much early.

Another issue is heterogeneity. WSN may consist of a large number of rather different nodes in terms of sensors, computing, power and memory. The large number raises scalability issues on the one hand, but provides a high level of redundancy on the other hand.

In WSN all nodes communicate with each other through communication channels. Moreover energy required to transmit a message is about twice as great as the energy needed to receive the same message. The route of each message destined to the base station is really crucial in terms of network lifetime i.e. using short routes to the base station that contains nodes with the depleted batteries may yield decreased network lifetime. On the other hand using the long route composed of many sensor nodes can

significantly increase the network delay**.** It is known fact that minimizing energy consumption in WSN is important for its usability and minimizing the flow of data is one method to achieve that.

3.2 Objective:

As the number of monitoring queries that are running in the sensor network increases, the energy consumption of the sensor network increases. This is because, for each query, each sensor node has to send its reading towards the base station whenever its reading satisfies the condition of the query. Thus, in order to reduce the energy consumption of the sensor network, it is crucial to reduce the number of monitoring queries injected into the sensor network.
There are two types of cost associated in Wireless Sensor Network: Communication cost and Computation Cost.

Computation cost concerns data collection, sampling by sensor nodes as well as local data processing and in network data processing. Communication component allows a set of distributed sensor nodes to send data to a central destination node.

In practice, monitoring queries often have similar expressions among them. Instead of running each query independently, if we reuse the results of queries to answer other queries, the overall energy consumption of the sensor network can be reduced because duplicate data requests can be eliminated.

General scenario of querying sensor network is, when user queries some information, he or she specifies queries through an interface at gateway. Then queries are parsed and Query plans are made. After that queries are injected into the network for dissemination. One query may be distributed to only a small set of sensor nodes for processing. When sensor node has the sampling data ready, results flow up out of the network to the sink. The data then can be stored at sink node. It can be used for further analysis and for visualized for end users.

So our main objective is to find out the best query processing plan to reduce the communication cost in wireless sensor network using multiple queries. The process of selecting best possible plan for query processing in called **query optimization**.

There are two types of query optimization [10] method:

I. Base Station Optimization

II. Network Optimization

I. Base station optimization:

In this multiple query optimization method at the base station to reduce the energy required by the sensor nodes. It can be done by reducing the number of monitoring queries injected into the sensor network. When a new query is submitted to the base station, the simulated method checks whether new query can be rewritten using currently running queries. The rewritten query is then evaluated at the base station using the results of currently running queries without being injected into the sensor network. Consequently, the number of queries injected into the sensor network is reduced, resulting in lower energy consumption and also decrease the communication cost.

II. Network o**ptimization**:

For the set of queries Q' that has been injected into the wireless sensor network, the sensor node collects the data and broadcast the resultant data intelligently to minimize the number of radio message and hence achieve bandwidth and energy efficiency. The ideal situation is that each sensor node just sends data only once to satisfy all the queries that need the data.

In wireless sensor network query optimization can be done at the sink node (Server side), where user injects the queries and also in between sensor nodes. But we focus on query optimization at base station. It will reduce the number of monitoring queries which are injected into the network, so reduce the communication cost.

CHAPTER 4

SIMULATION, DESIGN & DEVELOPMENT

4.1. Simulation Environment:

Simulation can provide a way to study system design alternatives in a controlled environment. It explores system's configuration that are difficult to physically construct and observe interactions that are difficult to capture in a live system. The wireless sensor network consisting of thousands of nodes have significant technical challenges. Developing software for resources constrained node is difficult, time consuming and expensive. Simulation of wireless sensor networks is a convenient tool for development. However results from simulation are different from live experiments due to simple model which do not accurately reflect real life.

Field testing is the most suitable testing environment, but it have some drawbacks i.e. many of applications and protocols being developed are tested in relatively small settings (2-100 nodes). Developers generally use these small scale networks to generalize scalability to large networks. Nature of wireless sensor network is unpredictable, especially regarding communication. Because of this accuracy of these generalization is questionable. I.e. small networks are difficult to effectively stress the network, protocol and applications. So large networks are needed for realistic testing this is unrealistic in practice because of higher cost.

Simulation based testing is a very convenient testing environment. A single PC can simulate hundred of nodes that can be easily deployed and redeployed through configuration files. Simulation also provides access to internal state of the nodes. But the challenge for simulation is the amount of interaction with node's environment. Wireless sensor applications often have tens of components (radio, timer, sensor etc.) that a simulator need to model.

4.2. TOSSIM [12]:

TOSSIM is a discrete event simulator for TinyOS sensor networks. Instead of compiling a TinyOS application for a mote, users can compile it into the TOSSIM framework, which runs on a PC. This allows users to debug, test and analyze algorithms in a controlled and repeatable environment. As TOSSIM runs on a PC, users can examine their TinyOS code using debuggers and other development tools.

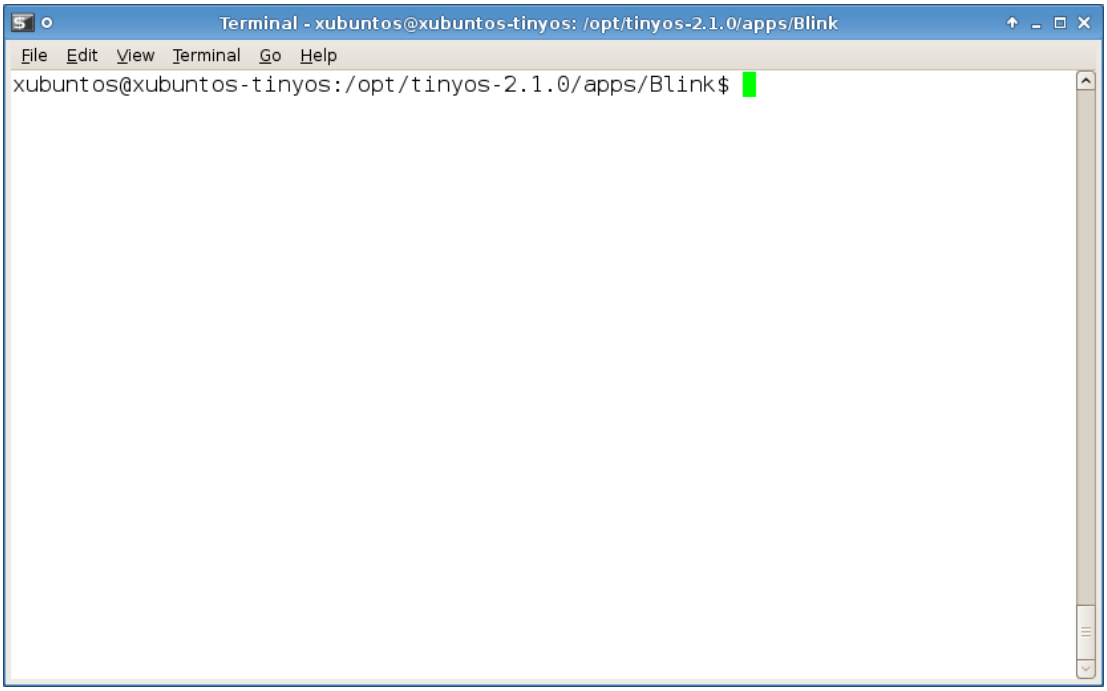

Fig.4.1. TinyOS

Experimental results show it to scale well up to one thousand motes. TOSSIM is compiled directly from TinyOS code, compiling a TinyOS application only requires specifying a different target for make (typing make pc instead of make mica).

Compiling a TinyOS application to native code allows a user to take advantage of traditional programming tools such as debuggers. TOSSIM has an extensive external communication system so testers can monitor packets being transmitted as well as dynamically inject packets into the simulated network. TOSSIM also allows finely grained configuration of debugging output at runtime.

Traditional network simulators are usually focused on protocol simulation for large area networks in which there is a large connectivity disparity in regions of the network (e.g. backbones vs. LANs). For example, ns-2 simulates at a packet granularity and has detailed node and link specification. In contrast, TOSSIM is intended for simulating homogeneous sensor networks, in which each mote runs the same program. TOSSIM simulates at bit granularity and has an extensible but so far simple network connectivity model. This is because a good deal of research using TinyOS looks at data link level mechanisms, whether for power conservation or more efficient use of the available bandwidth.

Currently, TOSSIM has three network connectivity models:

- simple connectivity (all the motes are in one cell)
- static connectivity (the network graph is established at start up and never changes)
- space connectivity (the motes move around randomly in a square area and potentiometer settings change their transmission range).

All of these models are very simplistic: e.g. mote transmissions never cancel one another (they are always in phase) and bits are perfectly transmitted.

4.2.1. Characteristics of TOSSIM

- **Fidelity:** By default, TOSSIM captures TinyOS behavior at a very low level. It simulates the network at the bit level, simulate each individual ADC capture and every interrupt in the system.
- **Building:** TOSSIM builds directly from TinyOS code. To simulate a protocol or system we must write a TinyOS implementation of it. This is often more difficult than an abstract simulation, but on other hand we can take our implementation and run it on actual motes.
- **Time:** While TOSSIM exactly time interrupts, it does not model execution time. From TOSSIM's perspective, a piece of code run instantaneously, time is kept at 4MHZ granularity.

- **Models:** TOSSIM itself does not model the real world. It provides abstractions of certain real world phenomenon. With tools outside the simulation itself, users can then manipulate these abstractions to implement whatever model they want to use.
- **Imperfections:** Although TOSSIM captures TinyOS behavior at a very low level, it makes several simplifying assumptions. This means that it is possible that code which run in simulation might run on a real mote. In real mote, an interrupt can fire while other code is running. If pre-emption can put a mote in a recoverable state, then simulated motes will run without mishap while real world motes may fail. Also, if interrupt handler run too long, a real-world mote may crash. As code in TOSSIM run instantaneously, no problems will appear in simulation.
- **Networking:** Currently, TOSSIM simulates the 40Kbit RFM mica networking stack, including the MAC, encoding, timing and synchronous acknowledgements. It does not simulate the mica2 ChipCon CC1000 stack.
- **Authority:** Initial experience from real-world deployments has shown that TinyOS network have very complex and highly variable behavior. While TOSSIM is useful to get a sense of how algorithms performs in comparison to one another.

4.3. TinyDB

TinyDB [10] provides a simple, SQL-like interface to specify the data you want to extract, along with additional parameters, like the rate at which data should be refreshed -- much as you would pose queries against a traditional database. Given a query specifying your data interests, TinyDB collects that data from motes in the environment, filters it, aggregates it together and routes it out to a PC.

TinyDB provides a simple Java API for writing PC applications that query and extract data from the network; it also comes with a simple graphical query-builder and result display that uses the API.

.

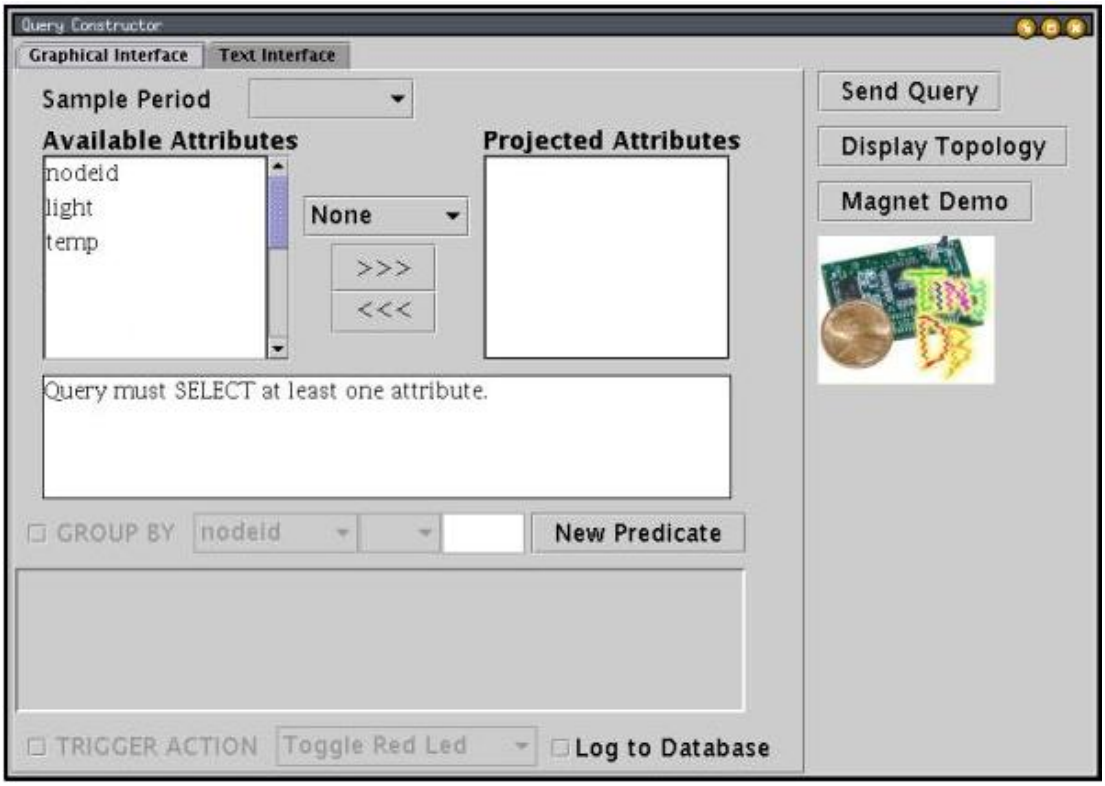

Fig.4.2. TinyDB GUI

4.4. NesC

NesC [14] is an extension to C designed to embody the structuring concepts and execution model of TinyOS.

The basic concepts behind NesC are:

- **Separation of construction and composition:** programs are built out of components, which are assembled ("wired") to form whole programs. Components define two scopes, one for their specification (containing the names of their interface instances) and one for their implementation.
- **Specification of component behaviour in terms of set of interfaces:** Interfaces may be provided or used by the component. The provided interfaces are intended to represent the functionality that the component provides to its user; the user interfaces represent the functionality the component needs to perform its job.

- **Interfaces are bidirectional:** they specify a set of functions to be implemented by the interface's provider (commands) and a set to be implemented by the interface's user (events). This allows a single interface to represent a complex interaction between components (e.g., registration of interest in some event, followed by a call-back when that event happens).
- Components are statically linked to each other via their interfaces. This increases runtime efficiency, encourages robust design and allows for better static analysis of programs.
- NesC is designed under the expectation that code will be generated by whole program compilers. This allows for better code generation and analysis. An example of this is NesC's compile-time data race detector.
- The concurrency model of NesC is based on run-to-completion tasks and interrupts handlers which may interrupt tasks and each other. The NesC compiler signals the potential data races caused by the interrupt handlers.

4.5. Query Optimization Techniques

Base station has the interface for the queries input in wireless sensor network. Thus we can use the base station as a filter to reduce duplicate data access from the sensor network. When a user submits a continuous monitoring query like "**SELECT nodeid, temperature FROM sensors WHERE temperature>35 SAMPLE PERIOD 4s**" to the base station, then query is injected into the sensor network and then each sensor node periodically sends its readings towards the base station if the readings satisfy the condition of the query. The sample period of a query specifies the time interval between the re-evaluation of the query.

As the number of queries that are running in the sensor network increases, the energy consumption of the sensor network increases. This is because each sensor node has to send its readings towards the base station. Thus in order to reduce the energy consumption of the sensor network, it is important to reduce the number of monitoring queries injected into the network.

The query optimization techniques are:

4.5.1. Merging of queries [10]

In practice, monitoring queries often have similar expression among them like SELECT, FROM, WHERE etc. Instead of running each query independently, if we merge queries to answer all queries, the overall energy consumption of the sensor network can be reduces because duplicate data request will be eliminated.

Consider the following two queries:

i) "select nodeid, temp where 25<temp<80 epoch duration 2 sec"

ii) "select nodeid, temp where 100<temp<150 epoch duration 4 sec"

To ensure correctness all the data requested by Query (i) and Query (ii) must be requested by query Q' and the result of both queries (i) and (ii) should be obtainable from result of Q'. The attributed and predicates of Q' should be union, while the epoch duration should be Greatest Common Divisor. Hence Q' should be: "select light where 25<temp<150 epoch duration 2 sec". Using the Greedy Query Insertion Algorithm [10] we can merge the two or more queries. In the case of Greedy query Insertion Algorithm Benefit [22] metrics is calculated. i.e. If Benefit >0 then we merge the Queries. For Example

Q_1= "SELECT light where 280<light<600 SAMPLE PERIOD 2s"

Q_2= "SELECT light where 100<light<300 SAMPLE PERIOD 4s"

Q_3= "SELECT light where 150<light<500 SAMPLE PERIOD 4s"

Then

$Benefit(Q_1,Q_2) = ((Sel(p1)/SP(Q_1))+ (Sel(p2)/SP(Q_2)) - (Sel(p1Up2)/GCD(SP(Q_1), SP(Q_2$
$)))$

$Benefit(Q_1,Q_2) = ((320/2)+(200/4)-(500/2)$

$=160+50-250$

$= -40 < 0$

Under this situation Q_1 and Q_2 will not be integrated.

Benefit(Q_2,Q_3) =((320/2)+(400/4)-(500/2)

=160+100-250

=10>0

So we integrate Q_3 with Q_2.

4.5.2. By using results of existing queries (Query Rewriting):

If we can reuse the results of existing queries to answer other new queries, the overall energy consumption of the sensor network can be reduced because duplicate data requests can be eliminated. For example, suppose that the following two monitoring queries q_1 and q_2 are currently running in a sensor network.

q_1=SELECT nodeid, temp FROM sensors WHERE temp > 35 SAMPLE PERIOD 2s.

q_2= SELECT nodeid, light FROM sensors WHERE light > 200 SAMPLE PERIOD 4s.

If a new query q_{new} is submitted to the base station.

q_{new}= SELECT nodeid, temp, light FROM sensors WHERE temp >50 AND light > 200 SAMPLE PERIOD 8s.

q'_{new} = SELECT nodeid, temp, light FROM q1, q2 WHERE q1.nodeid = q2.nodeid AND temp > 50 AND light > 300 SAMPLE PERIOD 8s

We can answer q_{new} using the results of q_1 and q_2 without injecting q_{new} into the sensor network. In other words, we can rewrite q'_{new} using q_1 and q_2 to reuse the results of those two queries.

q_{new} is rewritten using q_1 and q_2 into q'_{new}. Here q'_{new} does not need to be injected into the sensor network and can be evaluated at the base station using the results of q_1 and q_2. Since the base station has abundant computing resources and has no energy constraint, this reduces the overall energy consumption of the sensor network.

Constraint: data requested by q_{new} should be the subset of the data requested by q_1 and q_2 and epoch duration should be the multiple of LCM. Hence the integration of two queries in this way is guaranteed to be beneficial.

In our simulated method we follow the both techniques. In this we perform multiple queries optimization at the base station to reduce the energy required by the sensor nodes. It can be done by reducing the number of monitoring queries injected into the sensor network. When a new query is submitted to the base station, the simulated method checks whether new query can be rewritten using currently running queries. The rewritten query is then evaluated at the base station using the results of currently running queries without being injected into the sensor network. If rewriting the query is not possible then two or more queries are merged using the method explained in [4.5.1.]. Consequently, the number of queries injected into the sensor network is reduced, resulting in lower energy consumption.

Since query rewriting and merging is done at the base station, the query rewriting method is not affected by network conditions such as network topology, network size, routing protocols. The experimental results based on the TOSSIM simulator [12] show that the query rewriting method can significantly reduce the amount of data transmitted from sensor nodes to the base station, which leads to lower energy consumption.

Notation:

In sensors table sensor contains only the most recent reading of each sensor node. Each row of the sensor table contains the nodeid and the most recent reading of a sensor node.

The sensors table has the following schema:

Nodeid	Attribute1	Attribute2	Attribute3

Table 2. Sensor Table Schema

sensor_schema= (nodeid,A_1,A_2,A_3,.....,A_n)

where nodeid is the node of sensor node and A_1, A_2,A_3,.....,A_n are sensor attributes i.e. temperature, humidity, light, pressure etc. Note that nodeid is the primary key of the sensor table.

Let Q = {q_1,q_2,...,q_n} be the set of monitoring queries that are currently running in the sensor network.

We have taken the simple select and project queries, which are the mostly used types of monitoring queries in sensor networks. Where select consider the rows of table and Project consider the columns of a table. A select-project query q is expressed in the relational algebra as follows:

$$q = \Pi_{pl}(\sigma_{sc}(\text{sensors}))$$

Where pl is the projection list and sc is a selection condition. We denote the sample period (epoch duration) of query q as SP (q). SP shows that after how much time we have to take samples of data from the sensor nodes. Here, we assume that pl contains nodeid and sc has the following form:

$$sc = p_1 \wedge p_2 \wedge \ldots \wedge p_n$$

Where p_i ($1 \leq i \leq n$) is a simple predicate, which has the form:

$$p_i : A_k \lambda \text{ Value}$$

Where $A_k \in$ {nodeid, A_1, A_2,..., A_n}, $\lambda \in \{=, <, \leq, >, \geq\}$ and Value is chosen from the domain of A_k .

Let a query $q = \Pi_l (\sigma_{sc}(sensors))$

P(q) is the set of attributes that appear in l except nodeid and S(q) is the set of attributes that appear in sc. Let SC (q) denotes the selection condition of q and SC (q, A_i) denotes the selection condition of q consisting of only those predicates that involve A_i.

For example:

when SC(q) =‘(light> 150) ∧ (temp< 35)’,

SC(q, light) = ‘light> 150’

SC(q, temp) =‘temp< 35’.

Let q_{new} be a new monitoring query submitted to the base station. Our goal is to rewrite q_{new} using q_1, q_2,..., q_n. If such rewriting is possible, we do not inject q_{new} into the sensor network and evaluate it at the base station using the results of q_1, q_2,..., q_N. Consequently, the energy consumption of the sensor network is reduced.

Query rewriting [16] method consists of following steps:

i. Decomposition of queries
ii. Construction of Prospect query set
iii. Test for Query re-writability
iv. Transformation of query

i) Decomposition of queries:

When a new query q_{new} is submitted to the base station Firstly, we decompose q_{new} into d_{a1} , d_{a2} ,..., d_{an} as follows:

$q_{new} = d_{a1} \bowtie d_{a2} \bowtie \ldots \bowtie d_{an.}$

where $P(q_{new}) \cup S(q_{new}) = \{a1, a2,..., an\}$. P(q) is the set of attributes that appear in pl except nodeid and S(q) is the set of attributes that appear in sc

$$d_{ai} = \begin{cases} \Pi_{nodeid,ai}\,(\sigma_{SC}(q_{new},a_i)(sensors)), & \text{if } a_i \in P(q_{new}) \\ \Pi_{nodeid}(\sigma_{SC}(q_{new},a_i)(sensors)), & \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases}$$

for $1 \leq i \leq n$.

Each d_{ai} $(1 \leq i \leq n)$ contains nodeid, which is the primary key of the sensor table, in its projection list. Now it can be easily verify that q_{new} can be reconstructed by taking the natural join of $d_{a1}, d_{a2}, ..., d_{an}$ on nodeid.

Take an example, consider $q_{new} = \Pi_{nodeid,temp}\,(\sigma_{(temp<25)\wedge(light<200)}(sensors))$,where $P(q_{new}) \cup S(q_{new}) = \{temp, light\}$. q_{new} can be decomposed into d_{temp} and d_{light} as follows:

$q_{new} = d_{temp} \bowtie d_{light}$

$d_{temp} = \Pi_{nodeid,temp}(\sigma_{temp<35}(sensors))$

$d_{light} = \Pi_{nodeid}(\sigma_{light<150}(sensors))$

In what follows, we will rewrite each d_{ai} $(1 \leq i \leq n)$ using queries in

$Q = \{q_1, q_2, ..., q_n\}$.

ii) Construction of Prospect Query Set

There are number of queries at base station and we know that not all queries in $Q = \{q_1, q_2, ..., q_n\}$ can be used to rewrite d_{a1} , d_{a2} ,..., d_{an}. In this, we construct a prospect set of queries

$Q' \subseteq Q$ that can be used to rewrite $d_{a1}, d_{a2}, ..., d_n$ by filtering out the queries that cannot be used to answer q_{new}.

Method for constructing a prospect set of Queries:

a) Initially, set Q' to Q.

b) Remove from Q' those queries whose sample period is not a divisor of the sample period of q_{new}. Since q_{new} has to be re-evaluated for each sample period, those queries whose sample period is a divisor of the sample period of q_{new} can be used to answer q_{new}. For example, If the sample period of q_{new} is 32 seconds, only those queries whose sample period is 1, 2, 4 or 8, 16, 32 seconds can be used to answer q_{new}. All other queries whose sample period is 5, 6, 9 can't be used to answer q_{new}.

c) Remove from Q' those queries whose selection condition conflicts with the selection condition of q_{new}. That is, we remove q from Q' if $SC(q) \wedge SC(q_{new}) = false$. For example, if $SC(q_{new})$ = 'temp > 35' and SC(q) ='temp < 15',we remove q from Q' because (temp > 35) ∧ (temp < 15)= false.

After constructing a prospect query set Q', we construct Q'_{a1}, Q'_{a2} ,..., Q'_{an} from Q' ,where Q'_{ai} $(1 \leq i \leq n)$ is defined as follows:

$Q'_{ai} = \{q \mid (q \in Q') \wedge (ai \in P(q))\}$

That is, Q'_{ai} is the set of queries in Q' that contain a_i in the projection list. Then, we will rewrite each d_{ai} using queries in Q'_{ai} $(1 \leq i \leq n)$.

iii) Test for Query re-writability

We have to test whether d_{ai} can be rewritten using queries in Q'_{ai} $(1 \leq i \leq n)$ before rewriting d_{ai} using queries in Q'_{ai}, if we want to rewrite d_{ai} using queries in $Q'_{ai} = \{q_1, q_2,..., q_t\}$, the following condition must satisfy:

$SC(q_{new},ai) \rightarrow SC(q_1) \vee SC(q_2) \vee ... \vee SC(q_t)$

It mean the query region of d_{ai} , which is represented by $SC(q_{new},a_i)$, must be contained in the union of the query regions of queries in Q'_{ai} ,which is represented by $SC(q_1) \vee SC(q_2) \vee ... \vee SC(q_t)$. If the above condition does not hold for any d_{ai} $(1 \leq i \leq n)$, q_{new} cannot be rewritten using the currently running queries. In this case, we stop the rewriting process and inject q_{new} as it is into the sensor network.

iv) Transformation of query.

In the transformation of the query we check whether the above condition hold or not? If the condition in above holds for all d_{ai} and Q'_{ai} $(1 \leq i \leq n)$,we can rewrite each d_{ai} using queries in Q'_{ai}.We rewrite each d_{ai} using queries in Q'_{ai} as follows:

$$d_{a_i} = \begin{cases} \bigcup_{q \in Q'_{ai}} \prod \text{nodeid}, a_i(\ \sigma_{sc}(q_{new}, a_i)(q)) ; & \text{if } a_i \in P(q_{new}) \\ \bigcup_{q \in Q'_{ai}} \prod \text{nodeid}, a_i(\ \sigma_{sc}(q_{new}, a_i)(q)) \ ; & \text{If } a_i \in S(q_{new}) - P(q_{new}) \end{cases}$$

Q'_{ai} is the set of queries that contain a_i in the projection list. Thus, by applying $C(q_{new},a_i)$ to each $q \in Q'_{ai}$ and taking the union of them, we can obtain d_{ai} .After that we rewrite each d_{ai} using queries in Q'_{ai}, then we finally rewrite q_{new} as $q'_{new} = d_{a1} \bowtie d_{a2} \bowtie ... \bowtie d_{an}$. Then, q'_{new} is evaluated at the base station without being injected into the sensor network.

Algorithm for query re-writing

Input:

$Q=\{q_1, q_2,...,q_n\}$: the set of currently running queries;

q_{new}= a new query;

Output:

q'_{new}: a rewritten query of q_{new} defined over $q_{1,}q_{2,..,}q_{n;}$

Start

1. Let $P(q_{new}) \cup S(q_{new}) = \{a_1, a_2, .. a_n\}$;
2. /* construction of Prospect Query set */
3. Q'=Q;
4. for each $q \in Q'$ do
5. if SP(q) is not a divisor of $SP(q_{new})$ then remove q from Q';
6. else if $SC(q) \wedge SC(q_{new})$ =false then remove q from Q';
7. for each a_i in $P(q_{new}) \cup S(q_{new})$ do
8. $Q'_{ai} = \{q \mid (q \in Q') \wedge (a_i \in P(q))\}$;
9. /* test for query re-writabilty*/
10. for each Q'_{ai} $(1 \leq i \leq n)$ do
11. test whether $SC(q_{new}, a_i) \rightarrow \bigvee_{q \in Q'ai} SC(q)$ holds;
12. if test fails then return;
13. /* transformation of query*/
14. for each a_i in $P(q_{new})$ do

$$d_{ai} = \bigcup_{q \in Q' a_i} \Pi_{nodeid, a_i} (\sigma_{sc(q_{new}, a_i)}(q));$$

15. for each a_i in $S(q_{new}) - P(q_{new})$ do
16. $$d_{ai=} \bigcup_{q \in Q' a_i} \Pi_{nodeid,} (\sigma_{sc(q_{new}, a_i)}(q));$$

17./* Return Result */

18. return $q'_{new} = d_{a1} \bowtie d_{a2} \bowtie ... \bowtie d_{an}$;

Example

Assume that the sensor table has a schema (nodeid, light, temp) and the following four queries q_1, q_2, q_3, q_4 are currently running in the sensor network:

$q_1 = \Pi_{nodeid,light}(\sigma_{light \leq 150}(sensors))$, $SP(q_1)=4s$

$q_2 = \Pi nodeid,light(\sigma_{150<light}(sensors))$,$SP(q_2)=8s$

$q_3 = \Pi nodeid,temp(\sigma_{20<temp}(sensors))$,$SP(q_3)=2s$

$q4 = \Pi_{nodeid,temp}(\sigma_{20<temp<40}(sensors))$,$SP(q_4)=7s$

Suppose that the following new query q_{new} is submitted to the base station.

$$q_{new} = \Pi_{nodeid,light}(\sigma_{(100<light<250)\wedge(35<temp)}(sensor))\ SP(q_{new})=8s$$

Note that $P(q_{new}) \cup S(q_{new}) = \{light, temp\}$.First, decompose q_{new} into d_{light} and d_{temp} as follows:

$q_{new} = d_{light} \bowtie d_{temp}$

$d_{light} = \Pi_{nodeid,light(\sigma 100<light<250}(sensors))$

$d_{temp} = \Pi_{nodeid(\sigma 35<temp}(sensors))$

Next, we construct a prospect query set Q' from $Q = \{q_1, q_2, q_3, q_4\}$. Since $SP(q_4) =7s$ is not a divisor of $SP(q_{new}) =8s$,we filter out q_4 so that Q' becomes {q1, q2, q3}.Then, we construct Q'_{light} and Q'_{temp} from Q' as $Q'_{light} = \{q_1, q_2\}$and $Q'_{temp} = \{q3\}$ respectively. After that, we perform the query rewritability test for d_{light} and d_{temp} as follows:

d_{light} : $SC(q_{new},light) \rightarrow SC(q_1) \vee SC(q_2)$

d_{temp} : $SC(q_{new}, temp) \rightarrow SC(q_3)$

Note that SC(q_{new}, light) ='100 <light< 250', SC(q_1)='light ≤ 150', SC(q_2)='150 <light', SC(q_{new}, temp)='35 <temp' and SC(q_3) ='20 <temp'. Since the above conditions hold for both d_{light} and d_{temp}, we can see that d_{light} and d_{temp} can be rewritten using queries in Q'_{light} and Q'_{temp} respectively. We rewrite d_{light} and d_{temp} using queries in Q'light and Q'_{temp}, respectively as follows:

$d_{light} = \Pi_{nodeid,light}(\sigma_{100<light<250}(q1)) \cup \Pi_{nodeid,light}(\sigma_{100<light<250}(q2))$

$d_{temp} = \Pi_{nodeid}(\sigma_{30<temp}(q3))$

Finally, we rewrite q_{new} using d_{light} and d_{temp} as follows:

q_{new} = dlight $\bowtie d_{temp}$

$= (\Pi_{nodeid,light}(\sigma_{100<light<250}(q1)) \cup \Pi_{nodeid,light}(\sigma_{100<light<250}(q2)))$

$\bowtie \Pi_{nodeid,temp}(\sigma_{30<temp}(q3))$

CHAPTER 5

RESULTS, PERFORMANCE EVALUATION

Performance Evaluation:

In this section, we evaluate the effectiveness of the query rewriting method. We simulated the sensor network comprised of 25 static nodes uniformly distributed in the fixed area (500m X500m).We have implemented the query rewriting method in TinyDB, which is one of the most famous query processing systems for sensor networks. We have simulated a sensor network using TOSSIM [9], an event-driven simulator for TinyOS based sensor networks. In the experiment, we used dummy data obtained from [17]. The data obtained from [17] contain about 2.3million sensor readings collected from 54 Mica 2 Dot sensor nodes deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004. Each sensor reading has the information about date, time, temperature, humidity, light, voltage etc. With this setting, we have compared the performance of the following three methods for running multiple monitoring queries in the sensor network:

(1) Independent Queries: each query is independently executed in the sensor network.

(2) Merge: queries are optimized using the merge-based method [4].

(3) Query Rewriting and Merge: query rewriting method is used together with Merge. In this method, if a new query can be rewritten using currently running queries, the new query is rewritten by Query Rewriting and evaluated at the base station. If not, the new query is optimized using merge. For the performance measure, we have counted the number of sensor readings transmitted from sensor nodes to the base station. For each query injected into the sensor network, each sensor node sends its sensor reading towards the base station whenever its sensor reading satisfies the condition of the query. Note that the energy consumption of the sensor network increases as the number of transmissions increases. Since the performance of our

method does not depend on network conditions such as network topology, network size or routing protocols, we have not considered these factors in the experiments.

In the experiments, we used one Query-Set {q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}} as shown below.

q_1 = Π nodeid, temp (sensors)), SP(q_1) = 16s

q_2= Π nodeid, temp (30≤temp≤45) (sensors)), SP(q_2) = 32s

q_3 = Π nodeid, temp (σ(0≤nodeid≤20) ∧ (20≤temp≤28) (sensors)), SP(q_3) = 32s

q_4 = Π nodeid, light, temp (σ (0<nodeid<25 ∧ (σ(225≤light≤825) ∧ (sensors)), SP(q_4) = 8s

q_5 = Π nodeid, light, temp (σ (0<nodeid<15) ∧ (150≤light≤925) ∧ (23≤temp≤30) (sensors)), SP(q_5) = 8s

q_6 = Π nodeid, light (sensors)), SP(q6) = 16s

q_7 = Π nodeid, light, temp (σ(2≤nodeid<10) ∧ (temp<29) ∧ (200≤light≤700 (sensors)), SP(q7) = 32s

q_8 = Π nodeid, light (σ(400≤light≤1100 ∧ (27≤temp≤32)(sensors)), SP(q8) = 8s

q_9= Π nodeid, light (σ (0<nodeid<10 ∧ (σ(350≤light≤700) ∧ (sensors)), SP(q_4) = 16s

q_{10}= Π nodeid, light, temp (σ (0<nodeid<10 ∧ (σ(50≤light≤900) ∧ (sensors)), SP(q_4) = 4s

Query-Set is a query set where many queries can be merged but only a few queries can be rewritten using other queries. In Query-Set, q_2, q_3, q_7, q_9 is rewritten using q_1, q_4, q_5 by Query Rewriting and q_6, q_8 is merged with q_1, q_5 respectively by Merge. q_4, q_{10} runs independently in the sensor network.

Fig.5.1. shows the number of sensor readings transmitted from sensor nodes to the base station when Query-Set is executed using the three methods respectively. In this experiment, we have counted the number of sensor readings transmitted as the number of sensor readings produced increases. In the case of Query-Set, combination of Query Rewriting and Merge significantly reduced the number of transmissions compared to the other methods. The number of transmissions was reduced in

combination of Query rewriting and Merge compared to the other methods. When a total of 150000 sensor readings were produced, 414346 and 209322 sensor readings were transmitted in Independent and Merge respectively, while only 178786 sensor readings were transmitted in Combination of Query Rewriting and Merge. In this case, Combination of Query Rewriting and Merge reduced the number of transmissions by 56.86% and 14.6% compared to Independent and Merge, respectively. This means that even when there are only a small number of queries rewritten, Combination of Query Rewriting and Merge can benefit from Merge. From the experimental results, we can confirm that the simulated method can help reduce the number of transmissions for running multiple monitoring queries. Obviously, the performance gain of the simulated method will increase as the number of queries rewritten increases.

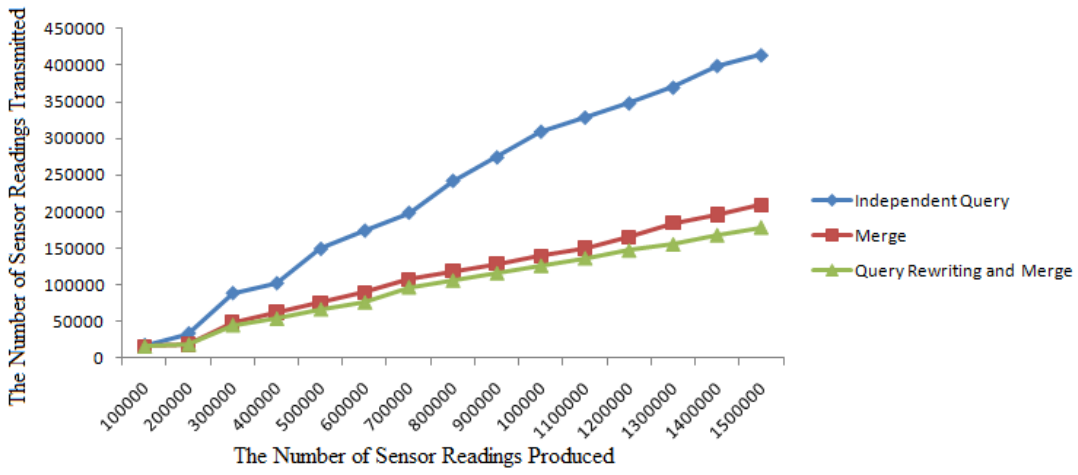

Fig 5.1(a)

Fig. 5.1.No of sensor readings transmitted and received

Table 3. Sensor reading transmitted and produced

No of sensor readings produced	No of sensor readings transmitted		
	Independent	Merge	Query Rewriting and Merge
100000	17735	16425	16175
200000	33786	19175	19089
300000	88875	48575	45466
400000	102423	62983	54275
500000	149467	75887	66975
600000	174389	89984	76434
700000	198876	107529	97235
800000	242497	118736	106325
900000	274671	128768	116286
100000	309886	139745	126456
1100000	329124	149923	136658
1200000	348476	165475	147221
1300000	370089	184727	155735
1400000	399825	195632	168215
1500000	414346	209322	178786

CHAPTER 6

CONCLUSION & FUTURE SCOPE

Since sensor nodes have limited energy capacity, it is important to minimize the energy consumption of sensor nodes to prolong the lifetime of a sensor network. In query rewriting algorithm when a new monitoring query is submitted to the base station, the rewriting query method rewrites the new query using currently running queries if possible. The rewritten query is then evaluated at the base station using the results of other queries without being injected into the sensor network. As a result, the energy consumption of the sensor network is reduced. The simulated method shows that this method reduces the number of sensor readings transmitted from sensor nodes to the base station, resulting in lower energy consumption as compared to independent and Merge.

In this work base station optimization is performed. For the future work network optimization can be done at network level. In further we can use individual sensor nodes for aggregation of data which can reduce the individual data packets.

[1] Chong and Kumar "*Sensor Networks: Evolution, opportunities and challenges***"**

[2] Crossbow Technology Inc. Data Sheet,

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[3] F.L.Lewis "*Wireless Sensor Networks"*

http://arri.uta.edu/acs/**networks**/**WirelessSensor**NetChap04.pdf

[4] Y. Yao , J. Gehrke. "*Query processing in Sensor Networks*". *IEEE Pervasive Computing*, 3(1):46–55, January-March 2004.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. "The Design of an Acquisitional Query Processor for Sensor Networks". In *Proceedings of ACM SIGMOD Conference on Management of Data*,pages 491–502, San Diego, CA, USA, June 2003.

[6] S. R. Madden, M. J. Franklin, and J. M. Hellerstein, "TinyDB:an acquisitional query processing system for sensor networks",ACM Trans. on Database Systems, Vol. 30, No. 1, 122-173,2005.

[7] P. Bonnet, J. E. Gehrke, and P. Seshadri, "Towards Sensor Database Systems", In Proc. of the Second International Conference on Mobile Data Management, Hong Kong, January 2001.

[8] William Miles "Optimizing SQL Query Processing"

[9] Elmasri, R. And Navathe, S. "Fundamentals of Database Systems", Benjamin Cummings Publishing Co., 1989

[10] S. Xiang et al., "Two-Tier Multiple Query Optimization for Sensor Networks", In Proc. of the 27th ICDCS, 2007

[11] http://telegraph.cs.berkeley.edu/tinydb/tutorial/tinydb.html

[12] P. Levis et al., "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", In Proc. Of SenSys 2003.

[13] Yannis E. Ioannidis , Viswanath Poosala, Balancing histogram optimality and **...** Matthias Jarke , Jurgen Koch, "*Query Optimization in Database Systems*"

[14] David Gay, Philip Levis, David Culler, Eric Brewer "nesC 1.1 Language Reference Manual"

[15] S. Xiang, Hock Beng Lim " Two-Tier Multiple Query Optimization for sensor Network"

[16] Yu Won Lee, Ki Yong Lee, Myoung Ho Kim "Energy-Efficient Multiple Query Optimization for Wireless Sensor Networks"

[17] Intel Lab Data, "http://www.select.cs.cmu.edu/data/ labapp3/index.html".

[18] Chris Townsend, Steven Arms " Wireless Sensor Networks: Principles and Applications"

[19] "SQL Query Processing and Optimization" www.kisekaeworld.com/Intractable/**SQL**OPT.pdf

[20] www.cs.trincoll.edu/~hellis2/CPSC372/.../Query%20Processing.pdf

[21] Ross Rosemark, Wang-Chien, Lee Bhuvan Urgaonkar "Optimizing Energy-Efficient Query Processing in Wireless Sensor Networks"

[22] Shili Xiang Hock Beng Lim KianLee Tan “Impact of Multiquery Optimization in Sensor Networks.”

[23] http://en.wikipedia.org/wiki/Wireless_sensor_network.

Published:

- Vipin Kumar, A K Verma, " Query Processing in Wireless Sensor Network-A Review" published in Proc. Of ICSCI-2010.(Pages 679-681)

Communicated:

- Vipin Kumar, A K Verma. "Query Optimization in Wireless Sensor Network" Communicated in IJCS.