CHAPTER 1

INTRODUCTION

There are two completing objects in the design of Wireless Sensor Networks (WSNs). The first objective is capability to exchange large amount of data between the nodes and the base station. The second constraining objective is minimizing the energy consumption. The two competing objectives reveal the importance of efficient routing protocol in WSNs. Therefore many routing algorithms have been proposed due to the challenges in designing an energy efficient network. Sensor node senses the environment, gathers the data from its surrounding (Computation) and communicates it to the Base Station (BS). Out of the three tasks communication takes large amount of battery power of a sensor node, so the major concern is the communication task. This system will minimize the communication cost in order to save battery power.

Wireless sensor networks consist of thousands of sensor nodes which are deployed randomly environment or space. In sensor network there is a Base Station which is located far away from the sensor field. Sensor nodes send the sensed data to the BS. For sending the sensed data to BS directly a lot of energy is consumed. So it is desirable to develop some protocols to minimize this communication cost. Energy conservation and maximization of network lifetime are the key challenges in the design and implementation of WSNs.

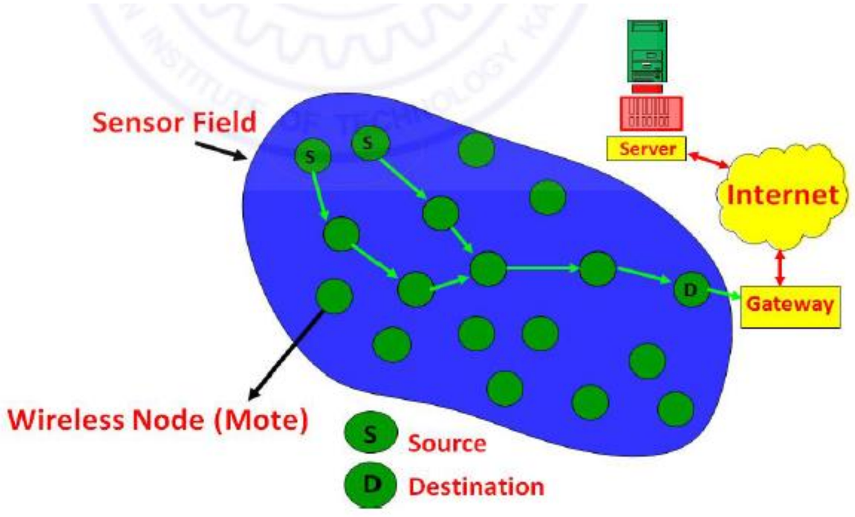


Fig 1.1: A Wireless Sensor Network

Every sensor node mainly consists of four components. They are sensing unit, transceiver, processing unit and power source. Some sensor nodes also consist of optional components like location Finding system, power generator and mobilizer.

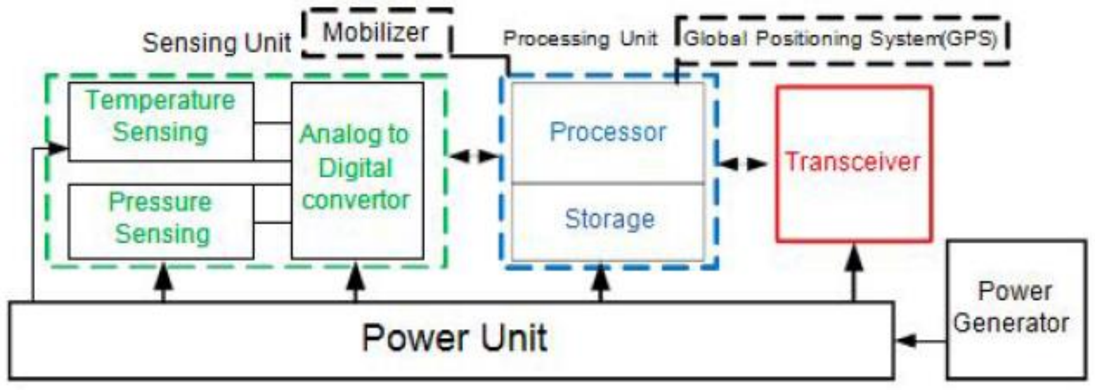


Fig 1.2: Sensor Node's Architecture

Among the functional components of a sensor node, the radio consumes a major portion of the energy. Various techniques are proposed to minimize its energy consumption.

In this report, A cost based energy balanced distributed clustering and routing scheme for wireless sensor networks is implemented, in which cluster heads (CHs) are selected from normal sensor nodes on the basis of composite weight value of its residual energy and neighborhood cardinality. All non-CH sensor nodes select a CH within its communication range, considering a cost value of the CHs. All sensor nodes use single-hop communication to communicate with its CH and all CHs use multi-hop communication to send the aggregated data to base station. In routing, we have also measure the cost of each path towards base station to select a proper route which can balance the energy of CH. Experimental results show that the proposed algorithm is more efficient with respect to energy consumption and number of live sensor nodes than LEACH.

1.1 Motivation

Our main goal of the proposed method is to prolong the network lifetime through an efficient distributed algorithm for proper CH selection and clustering, combined with energy balanced multi-hop route selection from all CHs to the sink by using these algorithms:

1. Selection of cluster head and cluster setup.

2. Select relay node.

1.2 Challenges

In this project, the main aim is to design an efficient routing and cluster head selection method to transfer data to base station. In this project, I am presenting two algorithms to overcome existing problems. Proposed algorithm compares the results with LEACH [5]. Results show that our proposed method is far better than the LEACH with respect to energy consumption and number of live sensor nodes.

1.3 Problem statement

Sensor nodes use batteries as the sole energy source. Therefore, energy efficient becomes critical. In many protocols do not consider both the clustering and routing issues combined. Many protocols selected a cluster head among normal sensor node and existing Protocols will take care of only clustering or routing but not both.

1.4 System Analysis

1.4.1. Existing System

LEACH is a popular cluster based routing technique. The main disadvantage of LEACH is that a sensor node with very low energy may be selected as a CH and the CHs send the packet to BS directly in single hop communication. Thus, this method increases the energy consumption of the CHs. A large number of algorithms have been developed to improve LEACH namely PEGASIS, HEED etc. Compared to LEACH, PEGASIS improves network lifetime, but its data delay is significantly high and it is unsuitable for large-sized networks. The HEED periodically selects CHs based on the node's residual energy and proximity measure of the neighbor nodes or node degree. In MRPUC, the authors design multi hop routing and unequal clustering algorithm using residual energies of all sensor nodes and distance between sensor nodes to the BS to extend network lifetime. EEDUC is a distributed clustering scheme, which is an improvement over EEUC to cover whole network. EEDUC forms cluster by setting a waiting time of each sensor node which is composite function of residual energy and number of neighbor of a sensor nodes. In both the cases of MRPUC and EEDUC, for a sensor node it is very hard to know global information of other sensor nodes for a large-scale network. An author “Li Qing et al”, propose a distributed energy efficient clustering algorithm called DEEC for heterogeneous networks. In DEEC, cluster heads are selected by a probability based on the ratio between the residual energy of each sensor node and the average energy of the network. However the exact information of the total energy of network is difficult to obtain, due to high node failure rate or redeployment. Therefore, DEEC cannot be applied in a harsh or hostile environment. DEBR is a distributed energy balance routing scheme, where a new heuristic metric is proposed to establish energy sufficiency as well as efficiency. In DEBR, a sensor node can select a next hop node, which is in opposite direction of BS and the selected next hop node can do the same thus it can increase the delay in transmission. Other clustering and routing algorithms have also been developed which can be seen. But all such protocols do not consider both the clustering and routing issues combined. The algorithm proposed in this article takes care of both these issues. Moreover, unlike the other works our method does not require the sensor nodes to be equipped with GPS.

1.4.2 Disadvantages of Existing System

In Existing system

1. Inefficient CHs can be selected
2. These inefficient cluster heads could not maximize the energy efficiency.

1.4.3 Proposed System

In this project, we propose a cluster based routing algorithm called CEBCRA (Cost-based Energy Balanced Clustering and Routing Algorithm) that addresses this issue. CEBCRA is a distributed algorithm which consists of three phases namely selection of CHs, cluster setup and data routing. The algorithm is fully based on the local information of a sensor node such as residual energy, number of neighbors and their distances. CEBCRA selects CHs amongst the normal sensor node using a weight function of the residual energy and the number of neighbors of a sensor node. Then all non-CH sensor nodes join a CH, which has maximum cost value within its communication range. The cost function is the composite measure of residual energy of the CH, its distance to base station and also the distance from the sensor node to the CH. For multi-hop routing, a CH needs to relay the data to the BS through other CHs. Therefore, the other CHs are treated as relay nodes. So, a CH needs to find the best neighbor CH (relay node) for data routing such that energy consumption is minimum. In the proposed work, the best neighbor relay node is selected by measuring the cost of each path towards base station.

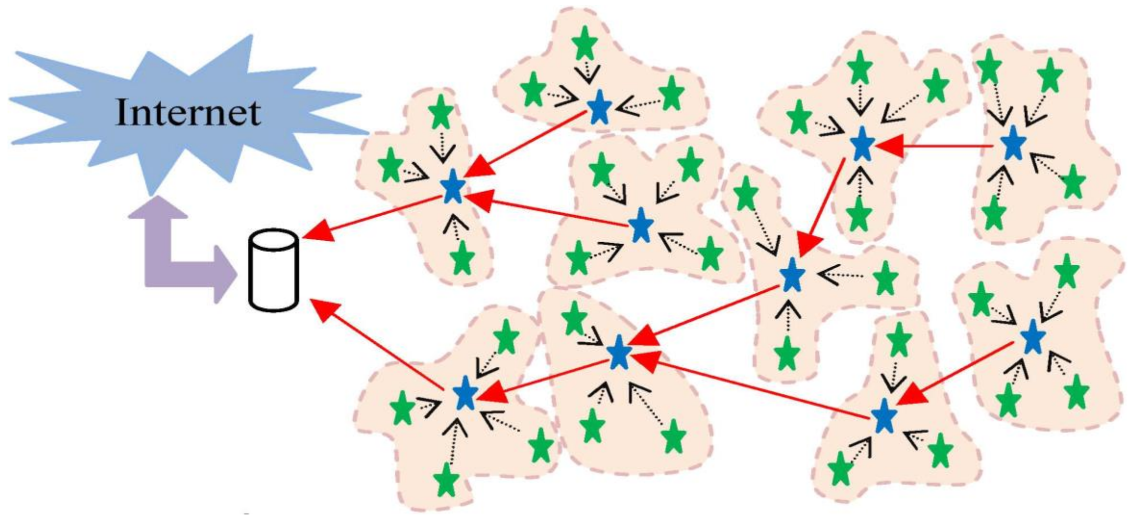


Fig 1.3: Proposed System Architecture

1.4.4 Objective:

The main objective of the proposed method is to prolong the network lifetime through an efficient distributed algorithm for proper CH selection and clustering, combined with energy balanced multi-hop route selection from all CHs to the sink.

CHAPTER 2

SYSTEM REQUIREMENT SPECIFICATION

2.1 Software requirements

- Coding Language :- JAVA(JDK 1.6 or more)
- NetBeans IDE 6.1 or More.
- Atarraya Simulator Tool.

2.2 Software Details

2.2.1 Introduction to JAVA

Java is an object-oriented programming language developed by Sun Microsystems, a company by author "James gosling" best known for its high-end Unix workstations. Modeled after C++, the Java language was designed to be small, simple, and portable across platforms and operating systems, both at the source and at the binary level (more about this later). Java is often mentioned in the same breath as HotJava, a World Wide Web browser from Sun like Netscape or Mosaic.

There were five primary goals in the creation of the Java language:

1. It should be "simple, object-oriented and familiar"
2. It should be "robust and secure"
3. It should be "architecture-neutral and portable"
4. It should execute with "high performance"
5. It should be "interpreted, threaded, and dynamic"

Platform independence is one of the most significant advantages that Java has over other programming languages, particularly for systems that need to work on many different platforms.

Java is platform-independent at both the source and the binary level. Platform-independence is a program's capability of moving easily from one computer system to another. At the source level, Java's primitive data types have consistent sizes across all development platforms. Java's foundation class libraries make it easy to write code that can be moved from platform to platform without the need to rewrite it to work with that platform. Platform-independence doesn't stop at the source level, however. Java binary files are also platform-independent and can run on multiple problems without the need to recompile the source. How does this work? Java binary files are actually in a form called byte codes. Byte codes are a set of instructions that looks a lot like some machine codes, but that is not specific to any one processor. Normally, when you compile a program written in C or in most other languages, the compiler translates your program into machine codes or processor instructions. Those instructions are specific to the processor your computer is running—so, for example, if you compile your code on a Pentium system, the resulting program will run only on other Pentium systems. If you want to use the same program on another system, you have to go back to your original source, get a compiler for that system, and recompile your code. Figure 3.1 shows the result of this system. Multiple executable programs for multiple systems Things are different when you write code in Java. The Java development environment has two parts: a Java compiler and a Java interpreter. The Java compiler takes your Java program and instead of generating machine codes from your source files, it generates byte codes.

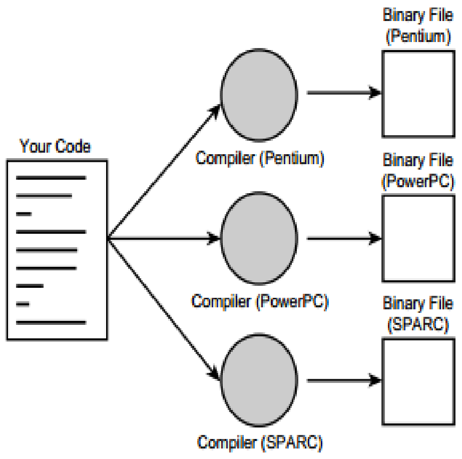


Fig2.1: Traditional compiled programs

To run a Java program, a bytecode interpreter program is called, which in turn executes your Java program (see Figure 2.2). You can either run the interpreter by itself, for applets there is a byte code interpreter built into Hot Java and other Java-capable browsers that runs the applet for you.

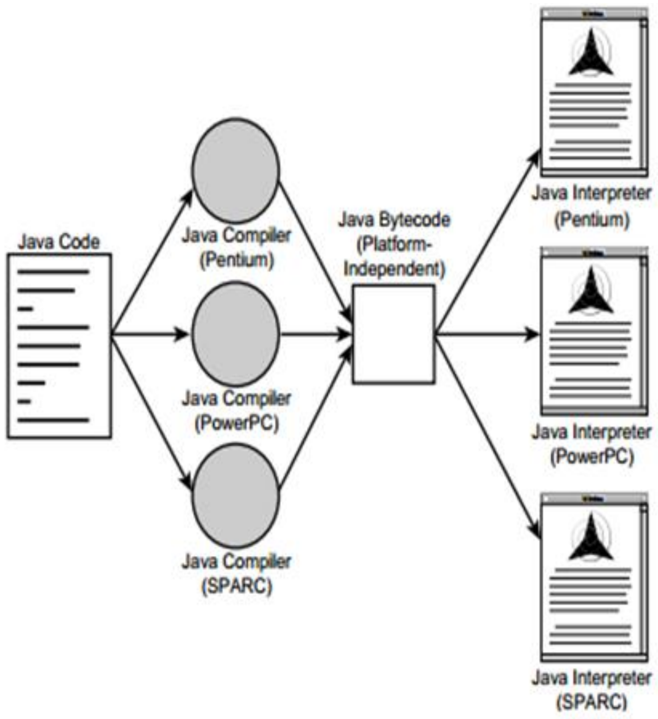


Fig 2.2: Java Program

2.2.2 Java architecture

Compilation and interpretation in Java

Java combines both the approaches of compilation and interpretation. First, java compiler compiles the source code into bytecode. At the run time, Java Virtual Machine (JVM) interprets this bytecode and generates machine code which will be directly executed by the machine in which java program runs. So Java is both compiled and interpreted language.

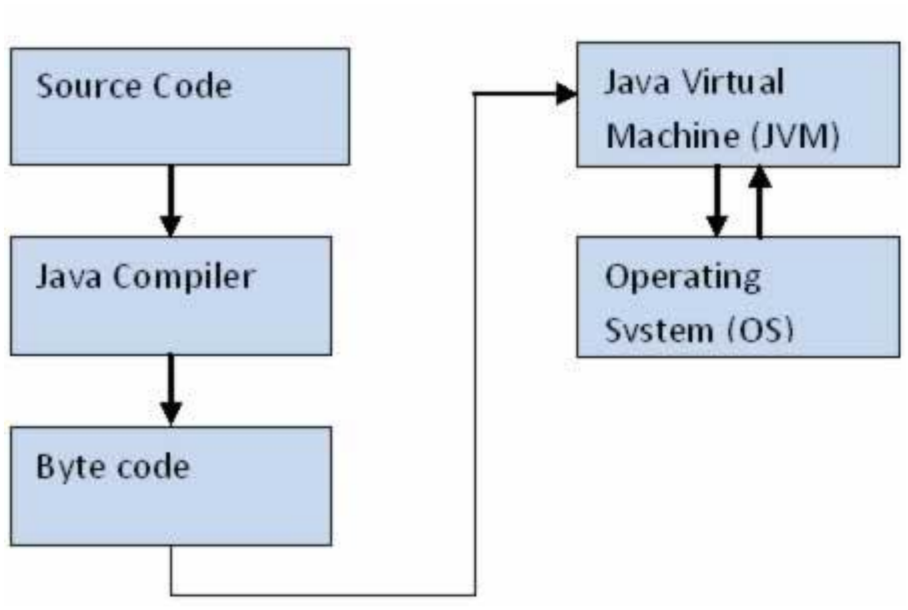


Fig 2.3: Java architecture

Java Virtual Machine (JVM)

JVM is a component which provides an environment for running Java programs. JVM interprets the bytecode into machine code which will be executed the machine in which the Java program runs.

2.2.3 NetBeans 6.9

The IDE used for this project is **NetBeans**. The NetBeans is an integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of an open-source IDE and an application platform that enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy and Grails, and C/C++.

The NetBeans project is supported by a vibrant developer community and offers extensive documentation and training resources as well as a diverse selection of third party plug-in. GlassFish Server 3.0.1 features alignment with NetBeans 6.9, support for Eclipse, scripting support including JRuby/Ruby and Groovy/Grails, an easy-to-use Administration Console and Update Center connectivity.

GlassFish Server 3.0.1 is built on a modular, flexible runtime based on the OSGi standard. It enables organizations to create and deploy Web applications with the lightweight Java EE 6 Web Profile and to easily leverage the power of the full Java EE 6 platform for enterprise applications. Developers also benefit from the simplified programming model and productivity

Improvements offered by Java EE 6. The result is a flexible platform that can apply only what is needed to address the business problem, thereby reducing cost and complexity. Because GlassFish Server 3.0.1 uses a microkernel architecture based on OSGi, developers can begin with the Java EE 6 Web Profile and use the Update Center to dynamically upgrade to the full Java EE 6 platform.

2.2.4 The Atarraya Simulator

The main idea behind the creation of Atarraya – which means *fishnet* in Spanish was to test the topology construction protocol named A3 that was being developed as part of this research. The software, as originally conceptualized, was very simple, rigid and tightly coupled with the A3 protocol. However, due to the fact that it was necessary to compare the performance of A3 against other known topology construction mechanisms, the design of the tool was not adequate. Therefore, the decision to build a more generic simulator, in which other topology construction algorithms could be plugged in, and have a single platform where to evaluate them all under the same conditions, was necessary. Then, the concept of topology control was also expanded to include topology maintenance algorithms, and several of these mechanisms were designed and included as well.

The final result is Atarraya

A generic ,Java-based, event-driven simulator for topology control algorithms in wireless sensor networks. As with any simulation tool born out of a research effort, Atarraya is still in development; however, in its current state, it is an excellent tool not only for research, to develop and test new topology control algorithms, but also for teaching. Atarraya’s graphical user interface shows how topology control protocols work, and how they shape topologies during their execution. In addition, Atarraya includes necessary mechanisms to experiment with classic theoretical results related to topology control in wireless sensor networks, such as the giant component experiment, calculation of the critical transmission range (CTR), calculation of the Minimum Spanning Tree of a graph, and others. In this appendix the basics of Atarraya are presented along with its internal structure, so the reader knows how to develop and plug in new topology control algorithms and protocols, and a brief guide on how to use the tool. All explanations and descriptions included in this document are related to Atarraya's version 1.0.

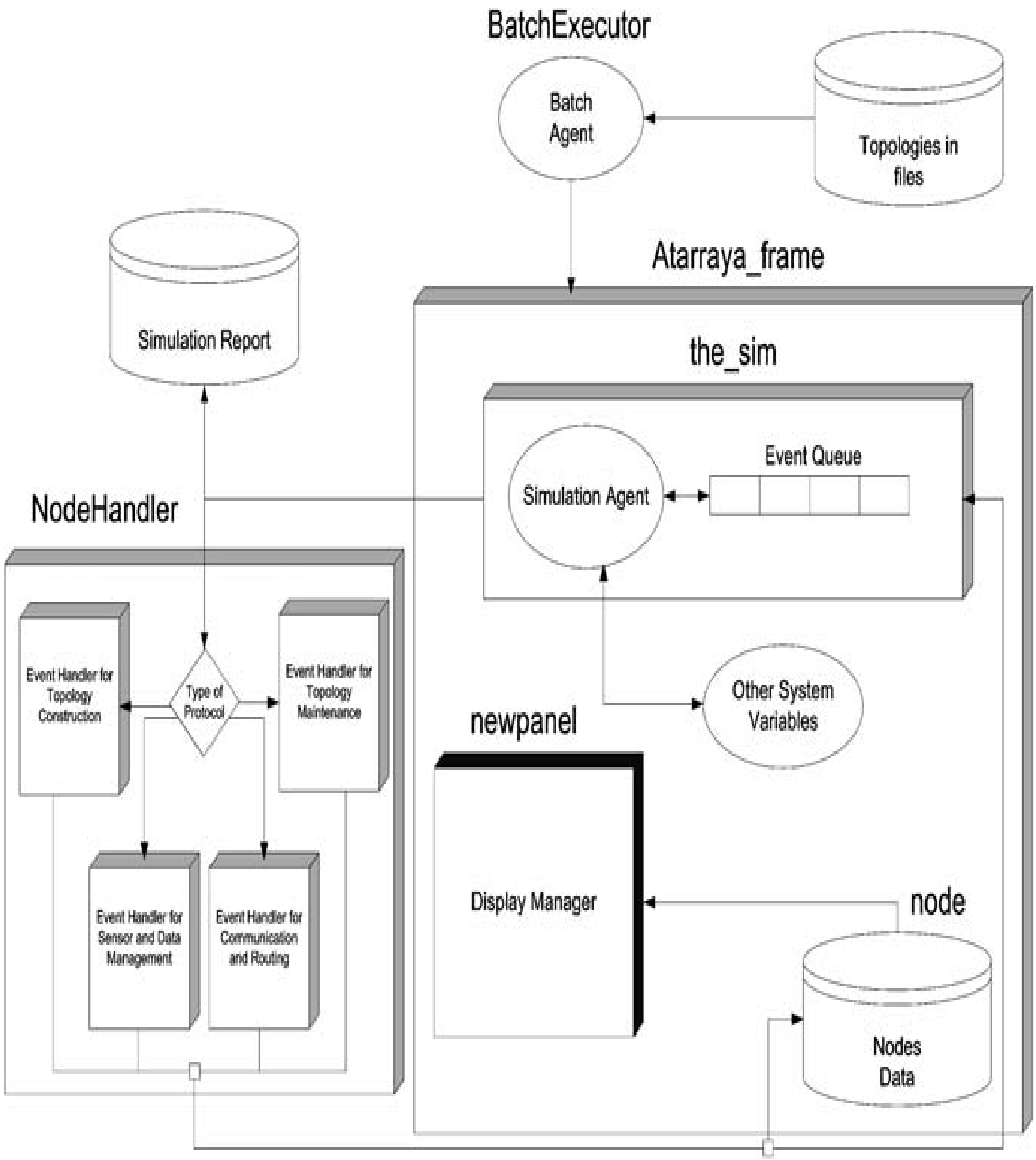


Fig 2.4: Atarraya's functional components.

Above Figure presents a global view of the internal structure of the simulator, which consists of the main simulator thread, the node handler, and the batch executor. The elements of this structure are described in this section.

The Main Simulator Thread – The *the_sim* Class

This is the core of the system. The simulator thread, defined in the class *the_sim*, is in charge of fetching the next event from the simulation event queue, and sending the event to the node handler for execution. An instance of this class is created by the method StartSimulation() whenever a simulation is executed. This class contains the event queue, the simulation clock, the display manager, the database with the data about the nodes, and the simulation agent, which is in charge of storing the simulation results for the reports in the respective logs.

When an instance of the *the_sim* class is created, it is necessary to add the initial events to the queue before the thread is started. The first thing the thread will do once started is to check if there are any events in the Event Queue. If the thread is started without any events, it will consider that an error has occurred, and the simulation will be suspended. In the loop, the first thing the thread does is to verify if the event is valid. If so, the event will be registered (if this option was selected by the user), the simulation clock will be updated, and the event will be sent to the *Node Handler*. There, the event will be delivered to the appropriate *Event Handler* according to the respective protocol the event belongs to. In the current version of the simulator, just one simulator thread can run at a time because there is only one data structure to store the topology, which is localized in the *atarraya_frame* class. Individual instances of the data structure running several simulations in parallel will consume all the resources of the Java virtual machine, especially if the network topologies are big.

The Protocol Manager – The *Node Handler* Class:

This class is in charge of defining the protocols to be used in the simulation and routing the event to the appropriate protocol once received from the simulation thread. The *Node Handler* class defines the four possible protocols that a node can have running during a simulation: Topology construction, topology maintenance, sensor-data management, and communication-routing protocols.

The classes in Atarraya are organized in three packages:

• *Atarraya*: the main functional elements are stored here, such as the main frame, the simulation agent, and the display manager.

• *Atarraya.element*: This package contains the classes that model the data structures, like the node, VNI, routing table, etc.

• *Atarraya.event*: This package contains the classes related to the protocols and the definition of the event queue.

The *Atarraya* Package

The *Atarraya* package is the main package of the simulator. It contains the following classes:

• *Main* class: This is the launcher of the simulator. It invokes the title frame and the main frame.

• *atarraya_frame* class: This is the main class of the simulator. In this class we find the definition of the graphical user interface and the simulator core.

• *newpanel* private class: This class defines the operation related to the visualization panel for the topologies: painting, selection of coordinates, selection of nodes, grid, etc. It is a private class of the *atarraya_frame* class so the *newpanel* class has direct access to the data structures.

• *the_sim* private class: This class defines the structure of the thread that simulates a scenario; in other words, this class is the simulation executor. It was made private also to preserve the direct access to the data structures.

BatchExecutor class: This class is in charge of executing operations that involve multiple scenarios, being that create multiple topologies, or simulate multiple scenarios. The advantage of using a separate class is that it creates a different thread that freezes the main frame while executing.

• *constants* interface: This interface defines the standard values for multiple variables.Given that many classes must share a set of standard values, the use of the *constant* interface allows this without having to define identical variables on each class.

• *FrameLogo* class: Initial frame with the logo of the simulator.

• *AboutFrame* class: Frame that contains the “About us” message.

CHAPTER 3

SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, and data for a system to satisfy specified requirements. System design could see it as the application of system theory to product development.

3.1 Block Diagram of clustering and routing

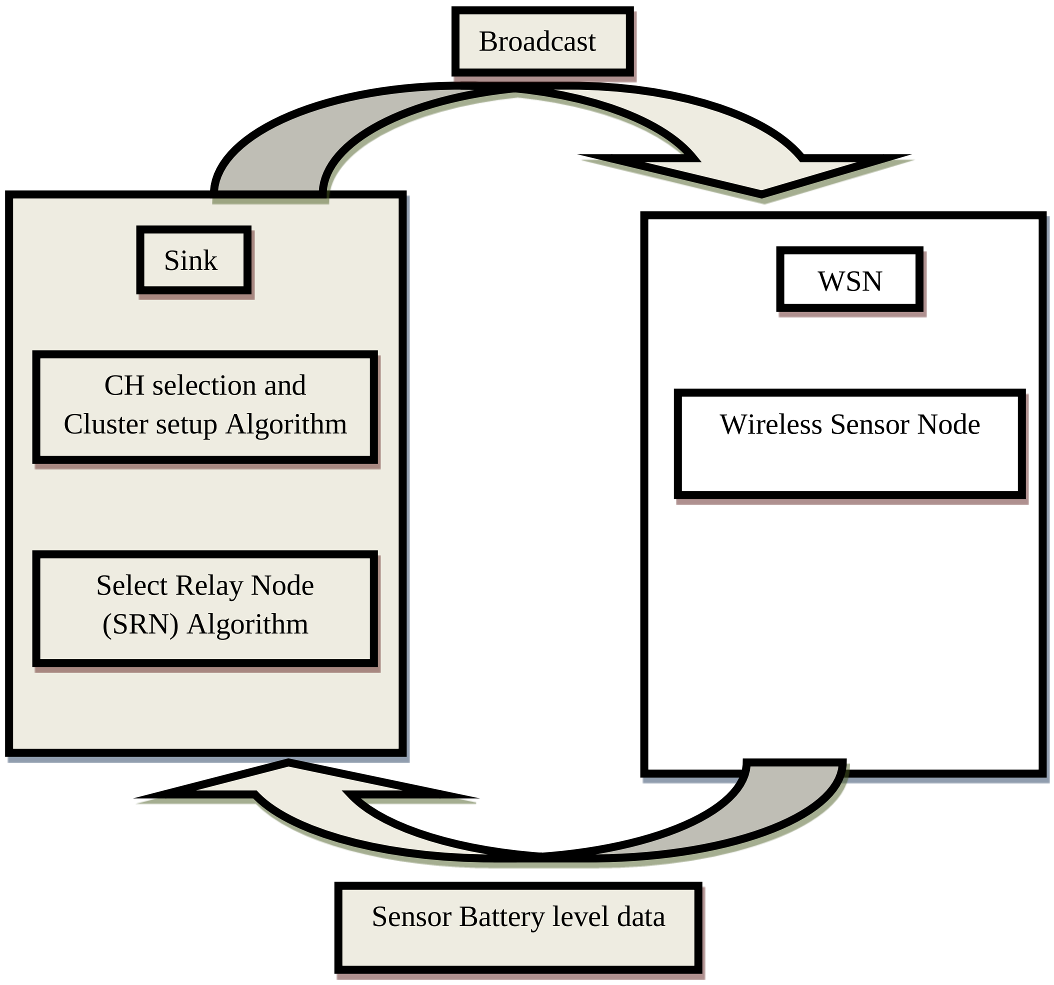

Broadcast
Sink
CH selection and Cluster setup Algorithm
Select Relay Node (SRN) Algorithm
WSN
Wireless Sensor Node
Sensor Battery level data

Fig 3.1: Block diagram of clustering and routing

In above block diagram (figure 3.1) there is two modules. One is that sink and another one is that sensor area (WSN). In sink we are proposing two algorithms. The battery level and sensed data will collect at the sink. The output of clustering and relay node algorithm it means found an efficient cluster head and to relay node to route data to base station.

3.2 Modules

1. Selection of Cluster Head
2. Cluster setup
3. Inter cluster multi –hop routing.

3.2.1 Selection of cluster Head

To select proper CHs from ordinary sensor nodes, the proposed method uses a weight function $W(Si)$ by considering the local network structure and the remaining energy of sensor node (Si).

Consider the following two factors for building the weight function.

- To become a CH, a sensor node must have HIGHER residual energy.
- The sensor node, which is selected as a CH, has to perform more rounds than simple sensor nodes i.e., the average amount of energy of the selected sensor node should be higher than that of any of its neighbor sensor nodes.

3.2.2 Cluster setup

- ✓ Each non-CH sensor node selects its CH based on a cost of CHs within its communication range.
- ✓ To define the cost a CH these are as follows

1. A sensor node should join that CH, which has the higher residual energy than other CHs within its communication range.

2. A sensor node should join its nearest CH such that the communication energy of the sensor node can be reduced.

3. The CHs which are near to the BS, always act as relay node in routing to forward the data packet to BS.

4. Each non-CH sensor node selects that CH which has maximum costs than all CHs within its communication range.

3.2.3 Inter cluster multi–hop routing.

- The CHs determine which CH is the best candidate as a relay node to communicate with the BS among its neighbors and itself.
- The CH selects the relay node in such a way that it pursues energy balance for the sensor network.
- The proposed method not only considers the available energy of the sensor nodes but also the required energy to communicate.
- To select a proper relay node, the proposed method calculates cost of each path. The path cost indicates nothing but the suitability of a path to achieve the energy balance of the network. If a cluster head Ci sends the aggregated data to base station via a relay node Cr then the total path cost $P\_Cost(Ci\ Cr)$.

3.3 Flowchart

A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Flow charts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts are used in designing and documenting complex processes or programs.
The two most common types of boxes in a flowchart are:
1. A processing step, usually called activity, and denoted as a rectangular box
2. A decision usually denoted as a diamond.

Base Station Broadcast :

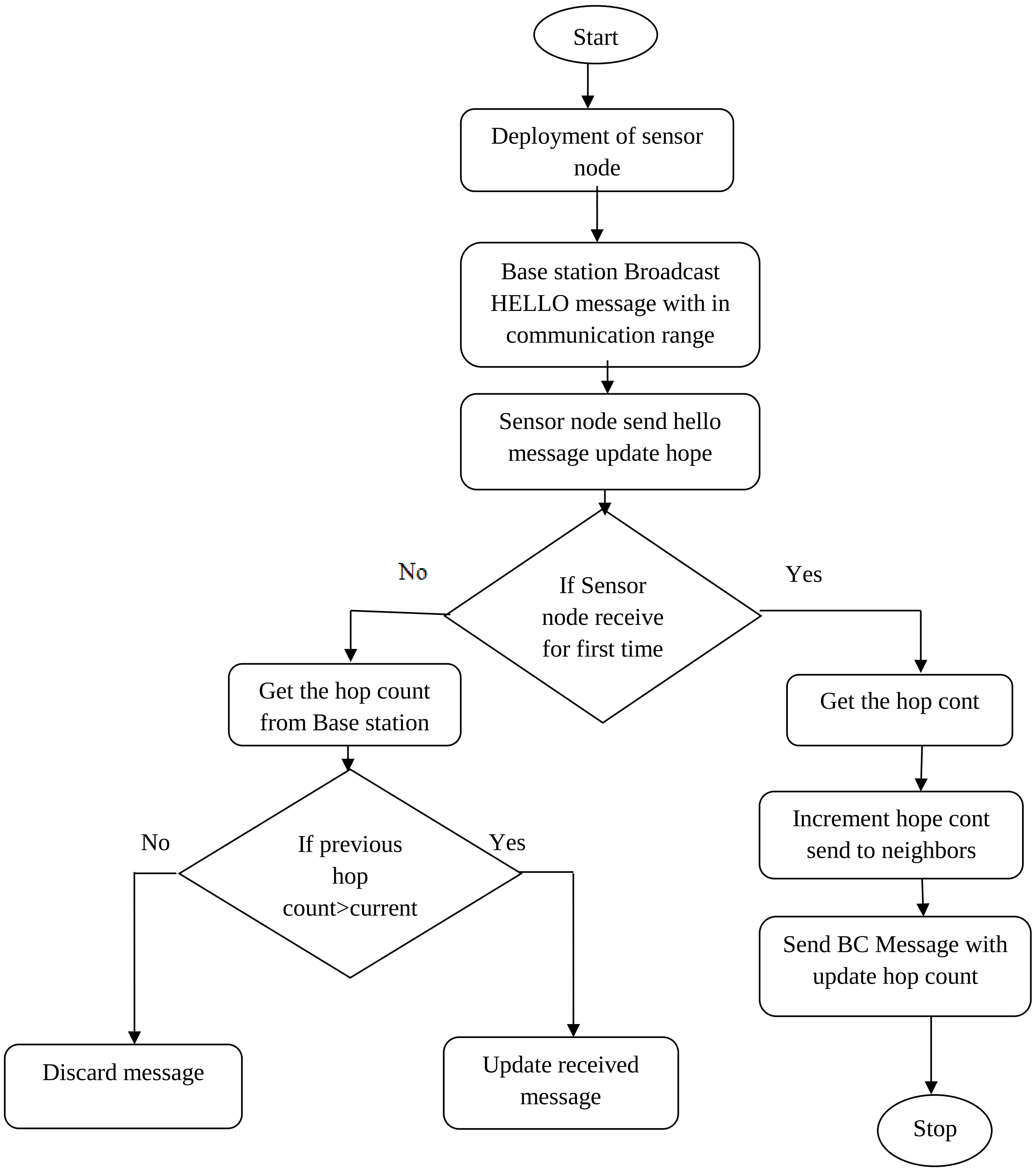

Start
Deployment of sensor node
Base station Broadcast HELLO message with in communication range
Sensor node send hello message update hope
If Sensor node receive for first time
No
Yes
Get the hop count from Base station
If previous hop count>current
No
Yes
Discard message
Update received message
Get the hop cont
Increment hope cont send to neighbors
Send BC Message with update hop count
Stop

Fig 3.2: Flowchart for sending and receiving process of sensor node

Cluster Head Selection :

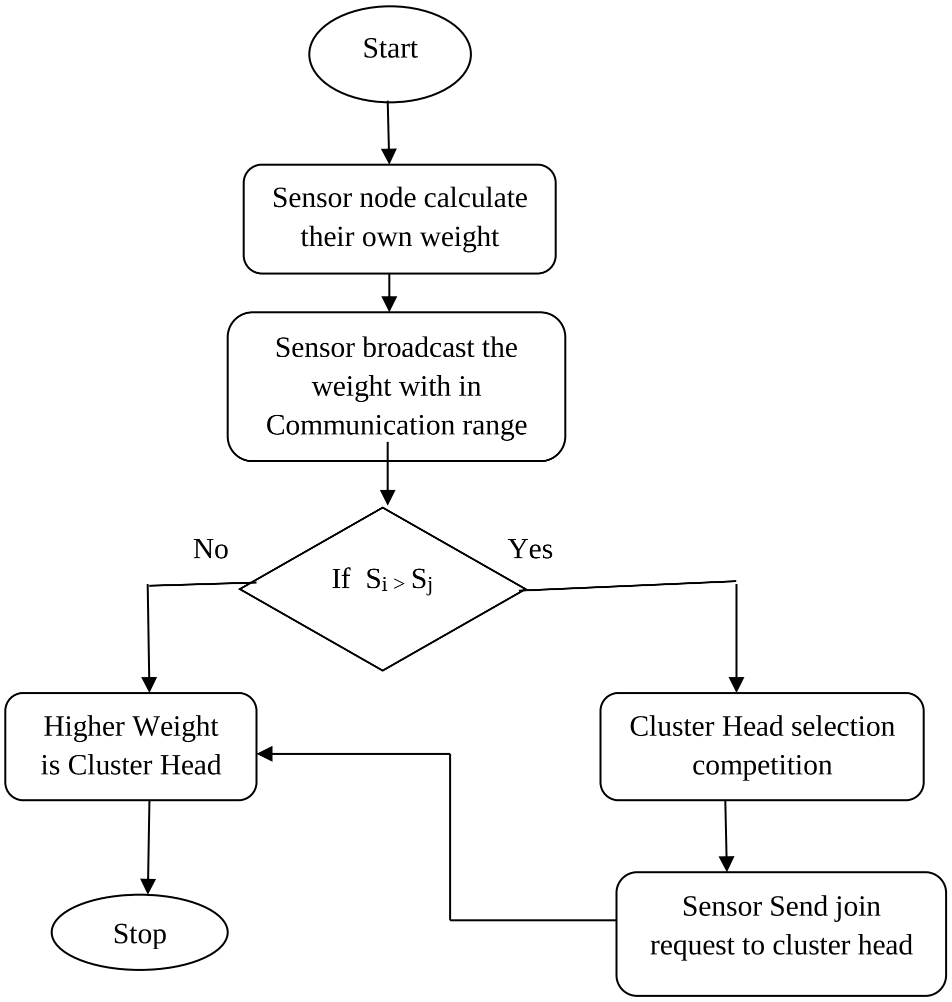

Start
Sensor node calculate their own weight
Sensor broadcast the weight with in Communication range
No
Yes
If $S_i > S_j$
Higher Weight is Cluster Head
Cluster Head selection competition
Stop
Sensor Send join request to cluster head

Fig 3.3: Flowchart for cluster head selection

Data Routing :

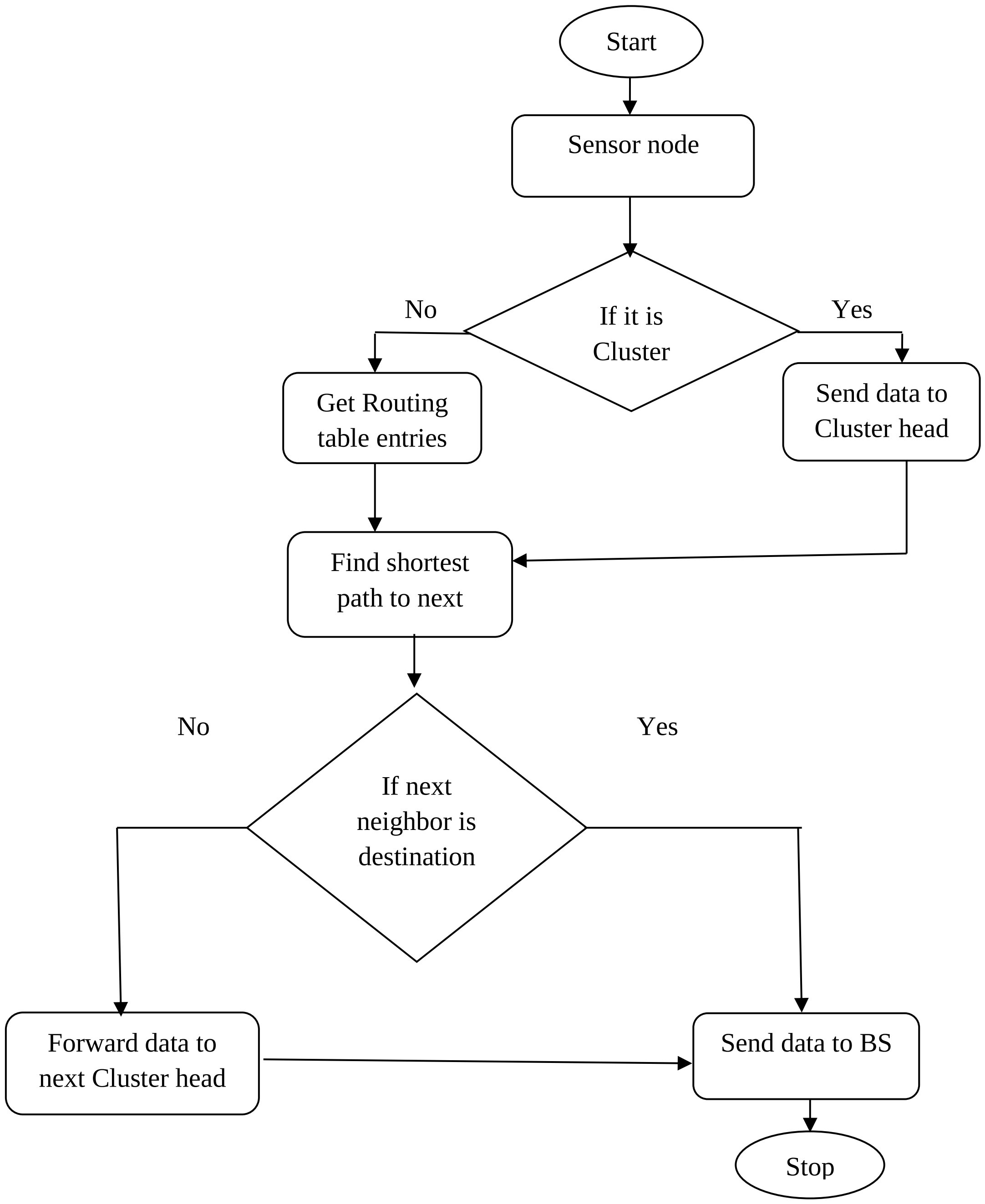

Start
Sensor node
No
If it is Cluster
Yes
Get Routing table entries
Send data to Cluster head
Find shortest path to next
No
Yes
If next neighbor is destination
Forward data to next Cluster head
Send data to BS
Stop

Fig 3.4: Flowchart for data transfer process

3.4 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, Event Scenarios and Timing Diagrams.

A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in the order in which they occurs. This allows the specification of simple runtime scenarios in a graphical manner.

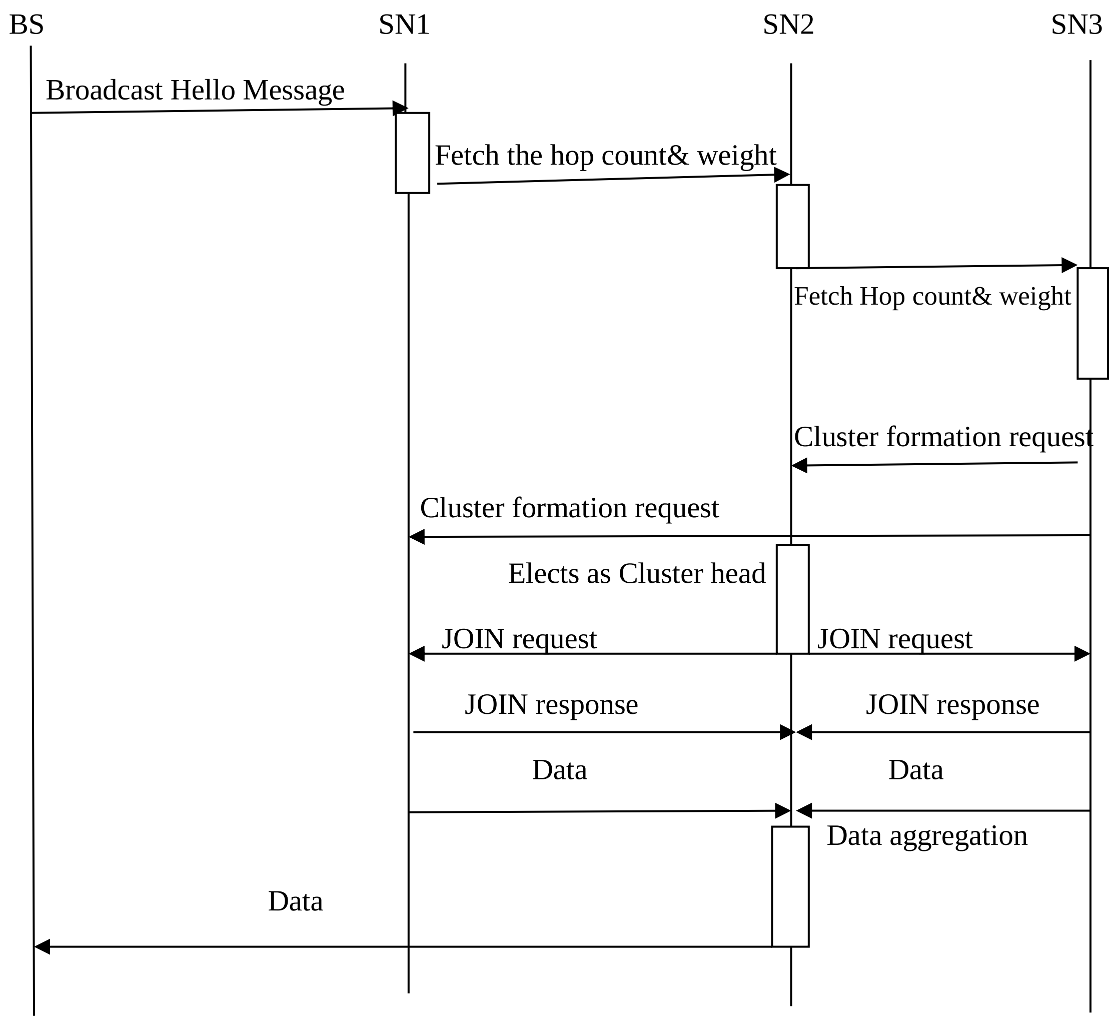


Fig 3.5. Sequence diagram for proposed architecture

3.5 Data Flow Diagrams (DFD)

Data-flow models are an intuitive way showing how data is processed by the system. At the analysis level, they should be used to model the way in which data is processed in the existing system. The notation used in these models re[resents functional processing data stores and data movements between functions. Dataflow models are used to show how data flows through a sequence of processing steps. The data is transformed at each step before moving on to the next stage. These processing steps or transformation are program functions where dataflow diagrams are used to document a software design.

With a dataflow diagram, users are able to visualize how the system will operate, what the system will accomplish and how the system will operate, what the system will accomplish and how the system will be implemented. Old system dataflow diagram can be drawn up and compared with the new systems dataflow diagram to draw comparisons to implement a more efficient system. Dataflow diagrams can be used to provide the end user with a physical idea of where the data they input, ultimately has an effect upon the structure of the whole system from order to dispatch to restock how any system is developed can be determined through a dataflow diagram.

There are several common modeling rules to be followed while creating DFDs are as follows:

- All processes must have at least one data flow in and one data flow out.
- All processes should modify the incoming data, producing new forms of outgoing data.
- Each data store must be involved with at least one data flow.
- Each external entity must be involved with at least one data flow.
- A data flow must be attached to at least one process.

Base station :

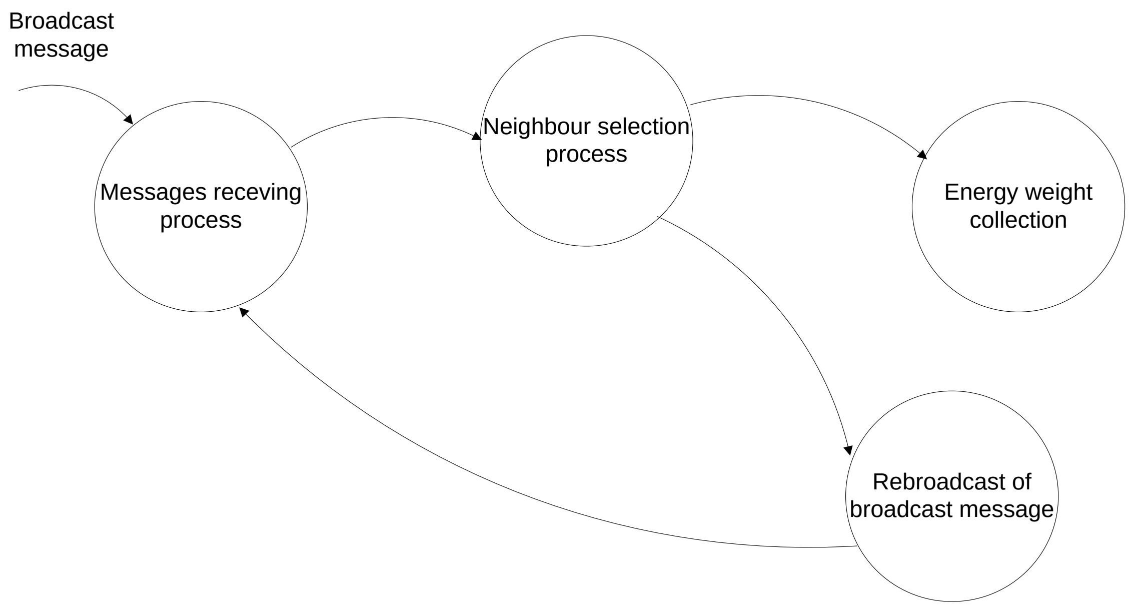

Broadcast message
Messages receving process
Neighbour selection process
Energy weight collection
Rebroadcast of broadcast message

Fig 3.6: Sender and Receiver Data Flow Diagram

Cluster head selection :

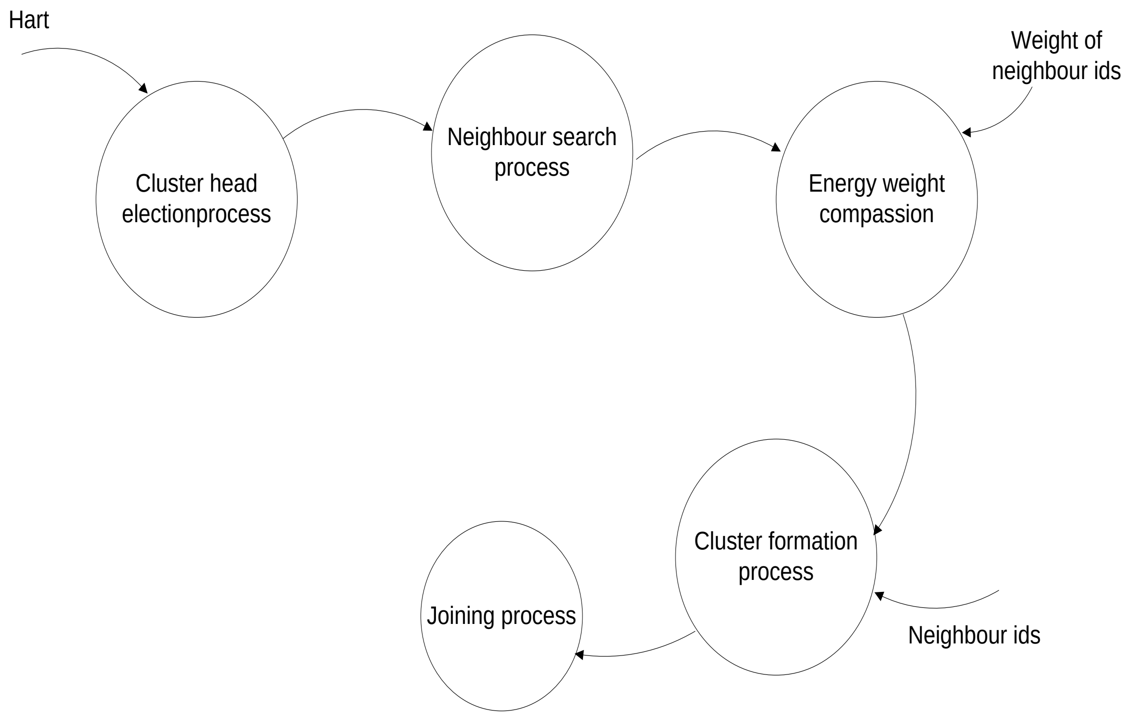


Fig3.7: Cluster Head Data Flow Diagram

Data Transfer Process :

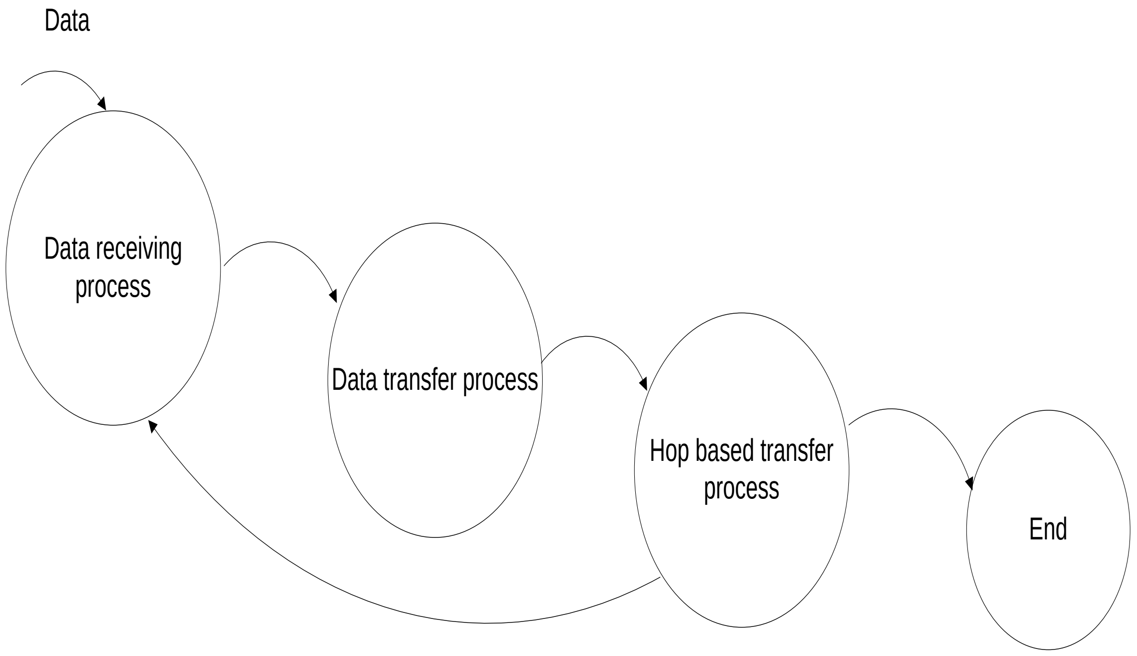


Fig 3.8: Data Transfer Process Data Flow Diagram

CHAPTER 4

IMPLIMENTATION

Implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing. Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard.

In this system contains 3 modules. Each Module work according to fulfill the requirement of project. The implementation of each module is conducted in such a way that it should not show any errors or exceptions. This implementation also tested with integrating several modules for the expected result.

4.1 Modules

1. Selection of Cluster Head
2. Cluster setup
3. Inter cluster multi –hop routing

The associated terminologies and notations are as follows.

- The set of sensor nodes is denoted by $S=\{S_1,S_2,S_3,..S_n\}$
- Set of selected cluster heads from (CHS) from normal sensor node $C=\{C_1,C_2,..C_n\}$
- $Dist\{S_i, S_j\}$ Distance between two sensor nodes S_i and S_j
- The set of all sensor nodes which are with in communication range $Com(S_i)$
- $E_{Residual}(S_i)$ residual energy of sensor node S_i
- $E_{Transmit}(S_i, S_r)$ required energy to transmit data from S_i to S_r
- $Neighbors\ (S_i)$ denotes the set of all sensor nodes that are within communication range of S_i. Therefore $Neighbors(S_i)=\{S_j \mid S_j \in Com(S_i) \wedge S_j \in \{S-S_i\}\}$

4.1.1 Selection of Cluster Head

To select proper CHs from ordinary sensor nodes, the proposed method uses a weight function *W(Si)* by considering the local network structure and the remaining energy of sensor node $S_i(\forall\ i,\ 1\leq i\leq n)$. CHs consume more energy than an ordinary sensor node. Therefore, we consider the following two factors for building the weight function.

1) To become a CH, a sensor node must have sufficient residual energy. So, more the residual energy, the higher is the chances of a sensor node to become a CH. In other words.

$$W(S_i)\ \alpha\ \ E_{Residual}(S_i) \qquad (4.1)$$

2) The sensor node, which is selected as a CH, has to perform more rounds than simple sensor nodes i.e., the average amount of energy of the selected sensor node should be higher than that of any of its neighbor sensor nodes. It means

$$W(S_i) \propto \frac{E_{Residual}\ (S_i)}{Neighbor(\,s_i\,)} \qquad (4.2)$$

where, $Neighbor(S_i)$ and $E_{Residual}(S_i)$ denotes the number of neighbor and residual energy of sensor node Si. Condition 4.1 and condition 4.2 combined implies that

$$W(S_i) \propto E_{Residual}\ (S_i) \times \frac{E_{Residual}\ (S_i)}{Neighbor(\,s_i\,)}$$

i.e $$W(S_i) = K1\ \times E_{Residual}\ (S_i) \times \frac{E_{Residual}\ (S_i)}{Neighbor(\,s_i\,)}$$

where, *K1* is a proportionality constant. Without loss of generality, we assume that K1=1

Therefore, $$W(S_i) = K1\ \times E_{Residual}\ (S_i) \times \frac{E_{Residual}\ (S_i)}{Neighbor(\,s_i\,)} \qquad (4.3)$$

- Sensor nodes are deployed randomly in coverage area.
- Base station broadcast "**HELLO"** message to all the nodes with in communication range at certain power level.
- Sensor nodes calculate distance to BS based on received signal strength .Its helps to proper power level to communicate with base station.

- Sensor node calculates there own weight by following equation 4.3 and broadcast it with in communication range.

If S_i receives higher weight value from any of its neighbor S_j then S, gives up the CH selection competition. A sensor node which does not receives a higher weight value from its neighbor, declares itself as a CH and broadcasts an advertisement message within its communication range.

4.1.2 Cluster setup

Each non-CH sensor node selects its CH based on a cost of CHs within its communication range. Let $CH\_Cost(C_i, S_j)$ be the cost of a cluster head C_i for sensor node S_j. The proposed method considers few factors of the sensor node S_j as well as the CH C_i, to define the cost a CH. These are as follows.

1. A sensor node should join that CH, which has the higher residual energy than other CHs within its communication range. Therefore,

$$CH\_Cost(C_i,S_j)\ \alpha\ E_{Residual}(C_i) \tag{4.4}$$

2. A sensor node should join its nearest CH such that the communication energy of the sensor node can be reduced. Therefore,

$$CH\_Cost(C_i,S_j)\,\alpha\ \frac{1}{Dist\ (\ S_j\ ,C_i\ \ )} \tag{4.5}$$

The CHs which are near to the BS, always act as a relay node in routing to forward the data packet to BS. In a result all these CHs consume more energy than other CHs which are far from BS. So, cluster member of these CHs should be less than the cluster member of the CHs which are far from BS. Therefore, cost of the CHs, which are far from the BS, should be higher than the CHs, which are near to the BS. In other words.

$$CH\_Cost(C_i,S_j)\ \alpha\ Dist(C_i,\ BS) \tag{4.6}$$

The condition 4.4 condition 4.5 and condition 4.6 combinedly implies that

$$CH\_Cost\big(C_j,S_j\big) = K2\ \times\ \frac{E_{Residual}\ (C_i)\ \times Dist(c_i,BS)}{Dist(\ s_j,c_i\ )}$$

Where, $K2$ is proportionality constant .with out loss of generality, we assume that $K2=1$.Therefore

$$CH\_Cost\big(C_j,S_j\big) = \frac{E_{Residual}\ (C_i)\ \times Dist(c_i,BS)}{Dist(\ s_j,c_i\ )} \tag{4.7}$$

Each non-CH sensor node selects that CH which has maximum cost (as equation 4.7) all cluster heads within its communication range.

- After selecting CH, each non-CH sensor node sends a JOIN_ REQ message to its elected CH using non-persistent CSMA MAC protocol.
- All cluster member use single-hop communication to communicate with CH.

ALGORITHM FOR SELECTION OF CHS AND CLUSTER SETUP

Input: Residual energy $E_{Residual}(S_i)$.

Output: S_i as a CH or a cluster member.

Step 1: S_i broadcasts and receive HELLO message to and from all the sensor nodes within its communication range.

Step 2: S_i counts $Neighbor(S_i)$ according to the received HELLO message and calculates $W(S_i)$ using equation 4.1.

Step 3: Si broadcasts $W(S_i)$ and receives $W(S_j)$, ∀ S_j ∈ $Neighbor(S_i)$.

Step 4: flag=l.

```
Step 5: while(flag= =1)&&(Si is receiving W(Sj))) do
            if(W(Sj)> W(Si)) then
                Si gives up CH select competition;
                flag=0;
        end if
    end while
Step 6: if(flag==1) then
        6.1: Si declares itself as CH and broadcast this message to all of its neighbors.
        6.2: Si receives joining messages from its neighbor sensor nodes and forms a cluster.
    else
        6.3: Si receives advertisement messages from other nodes who have declared
        themselves as CH.
```

6.4: S_i calculates cost of all these CHs according to equation 4.2 and joins the CH with the highest cost value.

end if

Step 7: Stop.

4.1.3 Inter Cluster Multi –Hop Routing

We now outline here the proposed scheme for data aggregation and routing. At the end of cluster formation as described in previous section, all sensor nodes sense the local data and send it to its CH using TDMA schedule. Once the CHs receive the data from all their corresponding members, they perform data aggregation to reduce the redundant and uncorrelated data within their cluster in distributed manner. This completes data aggregation. After data aggregation, CHs route their aggregated data either directly to BS or via other CHs treating as relay node. The CHs determine which CH is the best candidate as a relay node to communicate with the BS among its neighbors and itself. The CH selects the relay node in such a way that it pursues energy balance for the sensor network. The proposed method not only considers the available energy of the sensor nodes but also the required energy to communicate. By using a composite of both these quantities, a good path that achieves energy balance can be found. This is implemented as follows.

We assume that, each CH also aware of the remaining energy and distance of all CHs within its communication range from the BS. To select a proper relay node, the proposed method calculates cost of each path. The path cost indicates nothing but the suitability of a path to achieve the energy balance of the network. If a cluster head C_i sends the aggregated data to base station via a relay node C_r then the total path cost $P\_Cost(Ci,Cr)$ depends on few factors as follows.

1) The next hop relay node Cr should be selected such a way that it requires minimum energy to transmit the data from C_i to C_r. Therefore,

$$P\_Cost(C_i,C_r) \propto \frac{1}{E_{Transmit}(C_i,C_r)} \tag{4.8}$$

To be selected as a next hop relay node, C_r must have sufficient available energy i.e.,

$$P\_Cost(C_i,C_r) \propto E_{Residual}(Cr) \tag{4.9}$$

2) The relay node C_r receives the data from sender C_i and transmits the data to base station. So, total required transmitting energy and receiving energy is one of the most important factors. Obviously, the less required total energy indicates a suitable path. In other words

$$P\_Cost(C_i,C_r) \alpha \frac{1}{E_{Transmit}(C_i,C_r)+E_r} \qquad (4.10)$$

where, $E_{Transmit}(C_r,BS)$ is the required energy to transmit data from C_r to BS and E_R denotes the required energy to receive the packet. The condition 4.8, condition 4.9 and condition 4.10 combinedlly implies that

$$P\_Cost(C_i,C_r) = K3 \times \frac{1}{E_{Transmit}(C_i,C_r) \times \{E_{Transmit}(C_r,BS)+E_r\}} \qquad (4.10)$$

Where $K3$ is proportional constant. Without loss of generality,we have assume that $K3=1$.Therefore,

$$P\_Cost(C_i,C_r) = \frac{1}{E_{Transmit}(C_i,C_r) \times \{E_{Transmit}(C_r,BS)+E_r\}} \qquad (4.11)$$

If, C_i transmits the data directly to BS then the path cost is calculated as

$$P\_Cost(C_i,BS) = \frac{1}{E_{Transmit}(C_i,BS)} \qquad (4.12)$$

When a CH sends data to the base station, it first selects a best candidate from its neighbor and itself considering the maximum cost of the path following the equation 4.11 or 4.12. If the CH itself is a best candidate, then the CH itself sends the data to BS otherwise, it forwards the data to the selected best candidate node. Then, the selected best candidate node repeats the same routing process until a CH selects itself as a best candidate and sends the data directly to base station.

ALGORITHM: SELECT-RELAY-NODE (SRN)

Input: 1) Residual energy of all CHs from $Neighbor(C_i)$

2) Distance from C_i to BS and all CHs from $Neighbor(C_i)$.

Output: Next-hop relay node (Relay) of C_i

```
Step 1: Relay = BS ;     /* initially, relay node is BS */
Step 2: T= Neighbor($C_i$) ;
Step 3: While (T≠Φ) do /* Φ= NULL */
                Select a Cr from T;
        if(P_Cost($C_i$,$C_r$)>P _Cost($C_i$, Relay)) then
                Relay= $C_r$ ;
        endif
        T= T- $C_r$;
 Endwhile
Step 4: Stop
```

4.1.4 The major limitations:

(i) The use of tentative CHs that do not become final CHs leave some uncovered nodes. As per LEACH implementation, these nodes are forced to become a CH and these forced CHs may be in range of other CHs or may not have any member associated with them. As a result, more CHs are generated than the expected number and this also accounts for unbalanced energy consumption in the network

(ii) The CHs near the sink have more work load so may die early.

(iii) Overhead occurs as several iterations are performed to form clusters and lots of packets are broadcasted in each iteration.

CHAPTER 5

SYSTEM TESTING

5.1 Processes

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.2 Types of Testing

5.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Test case	**Input**	**Outcome**	**Status of execution Pass/fail**
Broadcast Message sending	Base station node	Broadcast messages with less hop count broadcasted	Pass
Energy calculation process	Weight of surrounding nodes	Election of cluster head or sensor nodes	Pass
Data transfer	Data from sensor nodes	Data forwarded to base station	Pass

Table 5.1 Test cases for Unit Testing

5.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Test case	**Input**	**Outcome**	**Status of execution Pass/fail**
Base Station Broadcast	Message to neighbors along with hop count	If previous count greater than current count update the received message	Pass
		Discard the message	Fail
Cluster Formation Module	Node & its energy weight	If node's weight greater than neighbor weight then fetch neighbor's process, send join request & declare as cluster head	Pass
		Send cluster formation list to neighbors'	Fails
Data Transfer Module	Data from sensor node	Send data to the destination	Pass
		Forward with destination to next base station	Fails

Table 5.2 Test cases for Integration Testing.

5.2.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

- *Valid Input* : Identified classes of valid input must be accepted.
- *Invalid Inpu t*: Identified classes of invalid input must be rejected.
- *Functions* : Identified functions must be exercised.
- *Output* : Identified classes of application outputs must be exercised.
- *Systems/Procedures* : Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.2.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.2.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully.

APPENDIX - A

1) CODE SNIPPET FOR SELECTING CHs AND CLUSTER SETUP

```
class MyDeadTimer extends Thread
{
   public void run()
   {
      while(!dead)
      {
         if(getEnergy()<=0.0)
         {
            dead=true;
            atarraya_frame.DEAD_NODES++;
            color=Color.MAGENTA;
         }
         Try
          {
            Thread.sleep(5000);
         }
         catch (InterruptedException ex)
         {
            Logger.getLogger(node.class.getName()).log(Level.SEVERE, null, ex);
         }
      }
   }
}
  public void startDeadTimer()
{
   new MyDeadTimer().start();
}
public int getHopCount()
{
   return DISTANCE_FROM_BASE_STATION;
}

public void setEnergy(double energy)
{
  this.energy=energy;
  this.TOTAL_ENERGY=energy;
```

```
   }
   public double getEnergy()
   {
     return energy;
   }
   public double getWeight()
   {
      return my_weight;
   }
   public void canbeSelected()
   {
      this.CanbeSelected=false;
   }
   public void reduceEnergyforClustering()
   {
      energy=energy-2;
      ENERGY_SPENT_FOR_CLUSTRING=ENERGY_SPENT_FOR_CLUSTRING+2;
   }

   public void reduceEnergyforDataTransfer()
   {
      energy=energy-2;

ENERGY_SPENT_FOR_DATA_TRANSFER=ENERGY_SPENT_FOR_DATA_TRANSF
ER+2;
   }

public void initial_event(int _id, int _treeID) {
                 try {
                 father .sat=true;
                double _clock = father.getVariable(CLOCK);
                int[]
neibours=father.getNode(atarraya_frame.BASE_STATION_ID).getNeighborsIds();
                for(int i=0;i<neibours.length;i++)
                {

father.pushEvent(newevent_sim(1,_clock,atarraya_frame.BASE_STATION_ID,neibours[i],
atarraya_frame.BASE_STATION_ID,atarraya_frame.MESSAGE_TYPE_BSHELLO));
   father.drawRedLines(atarraya_frame.BASE_STATION_ID, neibours[i]);
                father.repaint();
                }
```

```
               new Timer().schedule(new ClusterProcessTimer(),8000);
                        } catch (Exception ex) {

Logger.getLogger(EventHandlerBalancedClustering.class.getName()).log(Level.SEVERE,
null, ex);
             }
   }
public void sendClusterFormationRequest(int j,int greater_id,int[] ids)
    {
      double _clock = father.getVariable(CLOCK);
      father.pushEvent(new
event_sim(ids,_clock,j,greater_id,atarraya_frame.MESSAGE_TYPE_CLUSTER_FORMAT
ION));
      father.drawBlueLines(j,greater_id);
      father.repaint();
    }
  class ClusterProcessTimer extends TimerTask
   {
      public void run()
      {
         double _clock = father.getVariable(CLOCK);
         father.clearRedLines();
         father.repaint();
                for(int j=0;j<father.getVariable(NUMPOINTS);j++)
         {
           double my_weight=father.getNode(j).weight;
           double greater_weight=my_weight;
           int greater_id=j;
           int neibours[]=father.getNode(j).getNeighborsIds();
           for(int i=0;i<neibours.length;i++)
           {
    double weight=father.getNode(neibours[i]).weight;
              if(greater_weight<weight)
              {
                 greater_weight=weight;
                 greater_id=neibours[i];
              }
           }
             father.getNode(greater_id).color=Color.BLUE;
             father.getNode(greater_id).NODE_TYPE="cluster";
             father.repaint();
```

```
                try
               {
                   Thread.sleep(1000);
                }
               catch (InterruptedException ex)
              {

            Logger.getLogger(EventHandlerBalancedClustering.class.getName()).log(Level.SEV
            ERE, null, ex);
                }
                 sendClusterFormationRequest(j,greater_id,neibours);
            }
            }
        }
public void run()
        {
               boolean iam_cluster_head=true;
               int cid=id;
               Iterator<NodeEnergyStats> itr=CLUSTER_SENDER_LIST.iterator();
               while(itr.hasNext())
               {
                   NodeEnergyStats stats=itr.next();
                   int id=stats.NODE_ID;
                   double weight=stats.WEIGHT;
                   System.out.println("My Weight:"+my_weight);
                    System.out.println("Other Weight:"+stats.WEIGHT);
                   if(my_weight<weight)
                   {
                       iam_cluster_head=false;
                       cid=id;
                       break;
                   }
               }
               if(iam_cluster_head)
               {
                boolean below=false;
                if(EventHandlerClustering.BS_Y<getPosition().y)
                {
                    below=true;
                }
                int ids[]=frame.getNode(getID()).getNeighborsIds();
                for(int k=0;k<ids.length;k++)
                {
                   if(ids[k]==atarraya_frame.BASE_STATION_ID)
                   {
                ClusterDistance cdis=new ClusterDistance(id,0,position.x,position.y,below);
                EventHandlerClustering.cluster_list.add(cdis);
                break;
                   }
                   }
                   cluster_head=true;
                 double _clock = frame.getVariable(CLOCK);
                 color=Color.BLUE;
                 frame.repaint();
                 int[] nbrs=getNeighborsIds();
                 for(int i=0;i<nbrs.length;i++)
                 {
                   if(nbrs[i]!=atarraya_frame.BASE_STATION_ID)
                    {
                       reduceEnergyforClustering();
                   frame.pushEvent(new
event_sim(nbrs[i],id,nbrs[i],atarraya_frame.MESSAGE_TYPE_JOIN_REQUEST,_clock));
                   frame.drawBlueLines(id, nbrs[i]);
                   frame.repaint();
                      }
                 }
               }
               else
               {
                 frame.getNode(id).color=Color.RED;
                 frame.repaint();
                }
        }
     }
```

2) CODE SNIPPET FOR SELECTING-RELAY-NODE (SRN)

```
    class DataTransferTimer extends TimerTask
   {
      atarraya_frame frame;
```

```
        public DataTransferTimer(atarraya_frame frame)
        {
            this.frame=frame;
        }
       public void run()
       {
           if(!cluster_head)
           {
               for(int i=0;i<5;i++)
               {
               reduceEnergyforDataTransfer();
                 System.out.println("Enter Data Timer");
                 double _clock = frame.getVariable(CLOCK);
                 frame.pushEvent(new
event_sim(atarraya_frame.BASE_STATION_ID,id,DATA_NODE,atarraya_frame.MESSA
GE_TYPE_DATA_TRANSFER,_clock));
                 frame.drawGreenLines(id,DATA_NODE);
                 frame.repaint();
                   try {
                       Thread.sleep(3000);
                   } catch (InterruptedException ex) {
                       Logger.getLogger(node.class.getName()).log(Level.SEVERE, null, ex);
                   }
               }
           }
       }
    }
```

APPENDIX - B

SCREEN SHOTS AND RESULT ANALYSIS

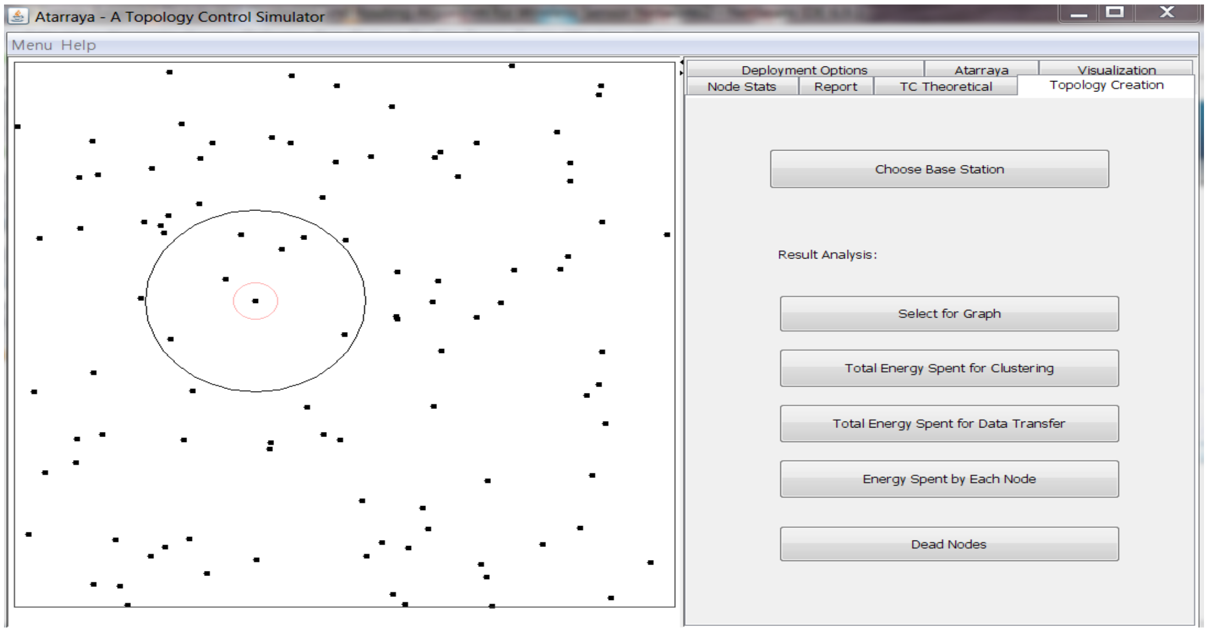
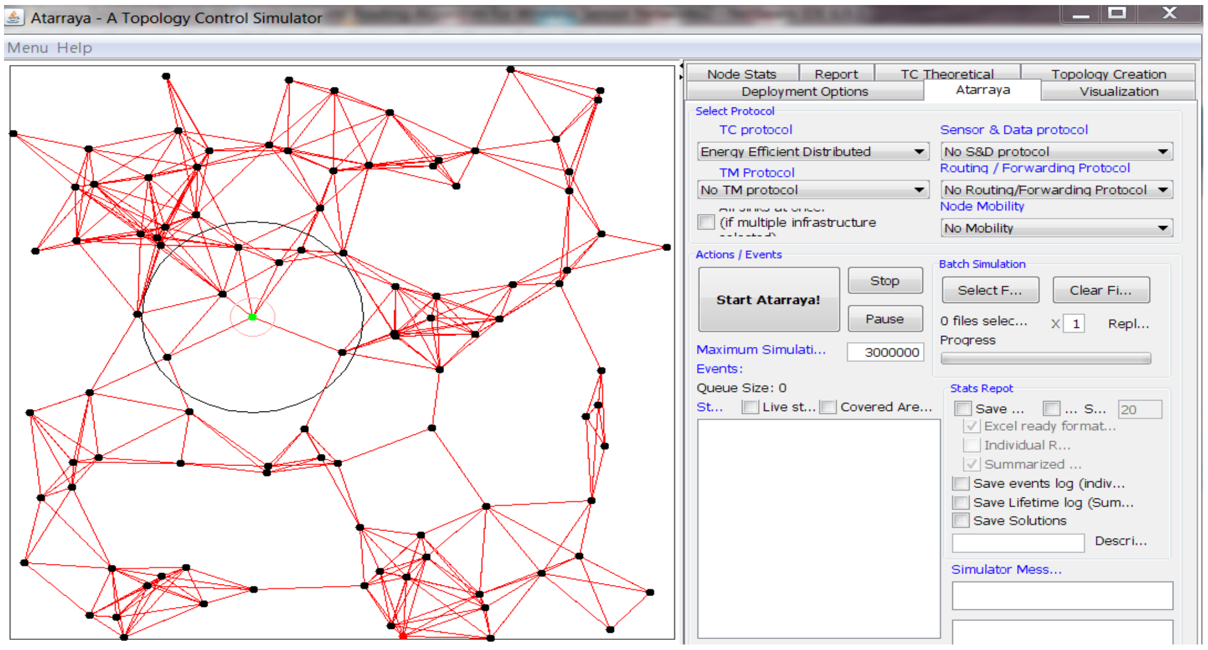
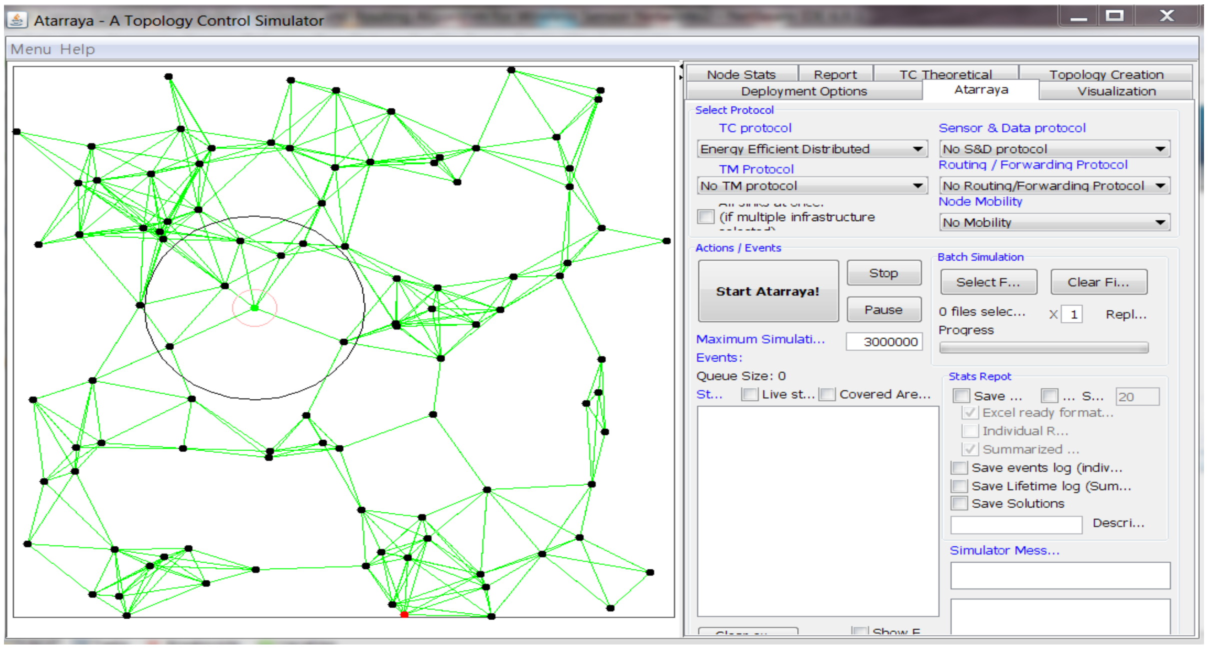


Fig 1: After deploying nodes and then Select one node as base station



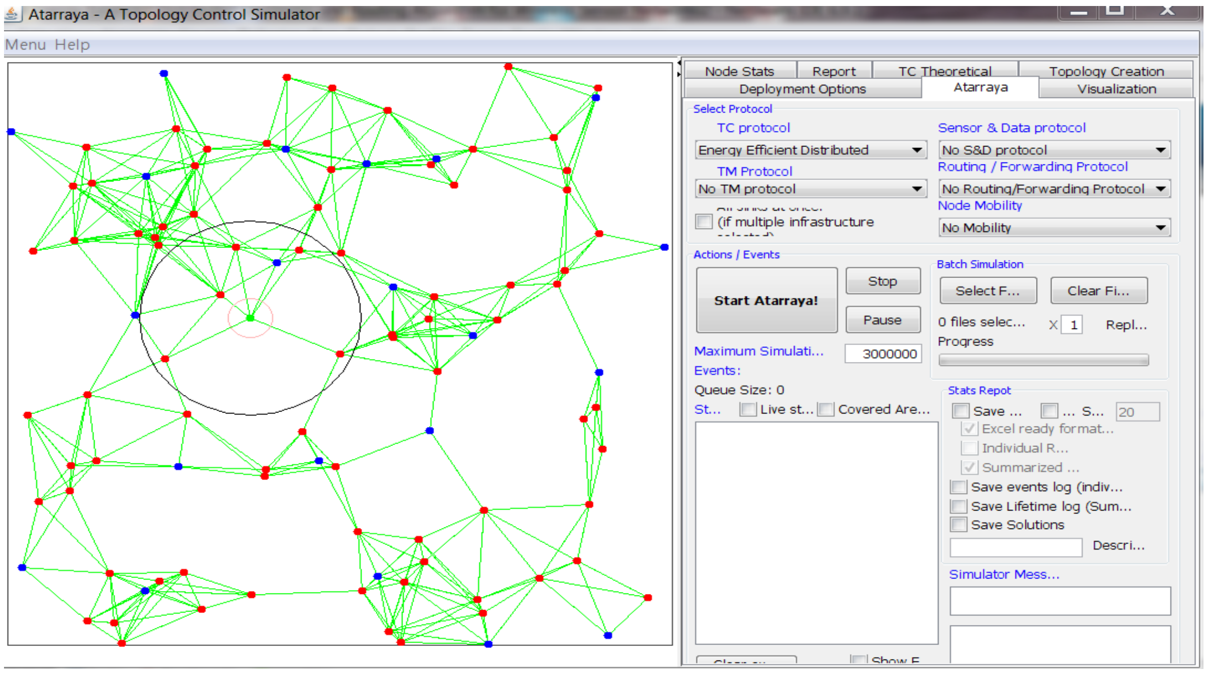
TC protocol: Energy Efficient Distributed
TM Protocol: No TM protocol
(if multiple infrastructure
Sensor & Data protocol: No S&D protocol
Routing / Forwarding Protocol: No Routing/Forwarding Protocol
Node Mobility: No Mobility
Maximum Simulati... 3000000
Events:
Queue Size: 0
St... Live st... Covered Are...
0 files selec... X 1 Repl...
Progress
Save ... ... S... 20
✓ Excel ready format...
Individual R...
✓ Summarized ...
Save events log (indiv...
Save Lifetime log (Sum...
Save Solutions
Descri...

Fig 2: Base Station broadcast all "HELLO" Message with in Communication Range



TC protocol: Energy Efficient Distributed
TM Protocol: No TM protocol
(if multiple infrastructure
Sensor & Data protocol: No S&D protocol
Routing / Forwarding Protocol: No Routing/Forwarding Protocol
Node Mobility: No Mobility
Maximum Simulati... 3000000
Events:
Queue Size: 0
St... Live st... Covered Are...
0 files selec... X 1 Repl...
Progress
Save ... ... S... 20
✓ Excel ready format...
Individual R...
✓ Summarized ...
Save events log (indiv...
Save Lifetime log (Sum...
Save Solutions

Fig 3: All the sensor node calculate there own weight by equation 5.3 and broadcast with in Communication range.

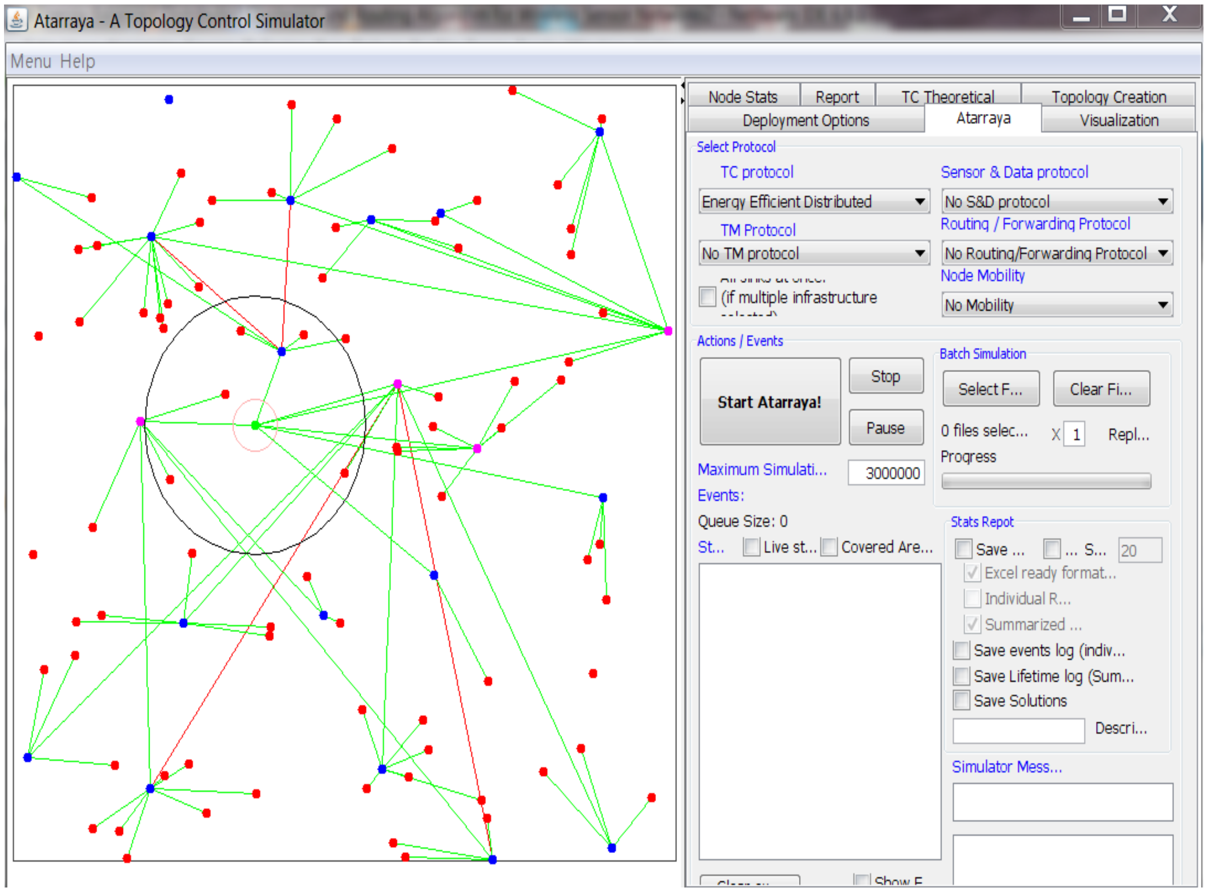


TC protocol: Energy Efficient Distributed
TM Protocol: No TM protocol
(if multiple infrastructure
Sensor & Data protocol: No S&D protocol
Routing / Forwarding Protocol: No Routing/Forwarding Protocol
Node Mobility: No Mobility
Maximum Simulati... 3000000
Events:
Queue Size: 0
St... Live st... Covered Are...
0 files selec... X 1 Repl...
Progress
Save ... ... S... 20
✓ Excel ready format...
Individual R...
✓ Summarized ...
Save events log (indiv...
Save Lifetime log (Sum...
Save Solutions

Fig 4: Cluster formation and cluster head selection process.

Blue nodes indicate **"CLUSTER HEAD"**

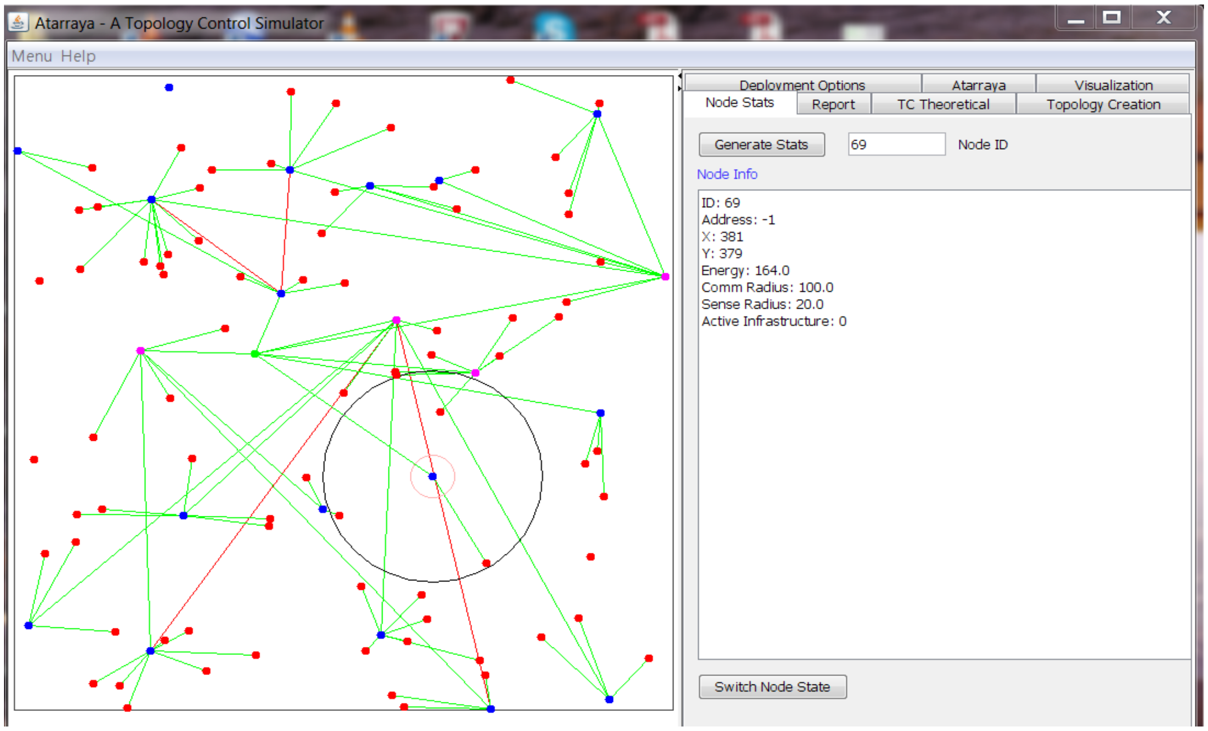
Read node indicates **"NORMAL SENSOR NODES"**



TC protocol: Energy Efficient Distributed
Sensor & Data protocol: No S&D protocol
TM Protocol: No TM protocol
Routing / Forwarding Protocol: No Routing/Forwarding Protocol
Node Mobility: No Mobility
(if multiple infrastructure
0 files selec... X 1 Repl...
Progress
Maximum Simulati... 3000000
Events:
Queue Size: 0
St... Live st... Covered Are...
Save ... ... S... 20
✓ Excel ready format...
Individual R...
✓ Summarized ...
Save events log (indiv...
Save Lifetime log (Sum...
Save Solutions
Descri...

Fig 5: Data routing process

- All cluster head send data to base station through relay node
- Some of the sensor nodes its not belongs to any group that means non of the cluster head belongs to that senor node with in that communication Range.
- Read and Green lines indicate data transferring to base station through rely node.
- Pink color node represents the dead nodes.
- Here dead nodes are less when compare to LEACH



```
ID: 69
Address: -1
X: 381
Y: 379
Energy: 164.0
Comm Radius: 100.0
Sense Radius: 20.0
Active Infrastructure: 0
```

Fig 6 : when we click on particular node it shows Node status in right hand side

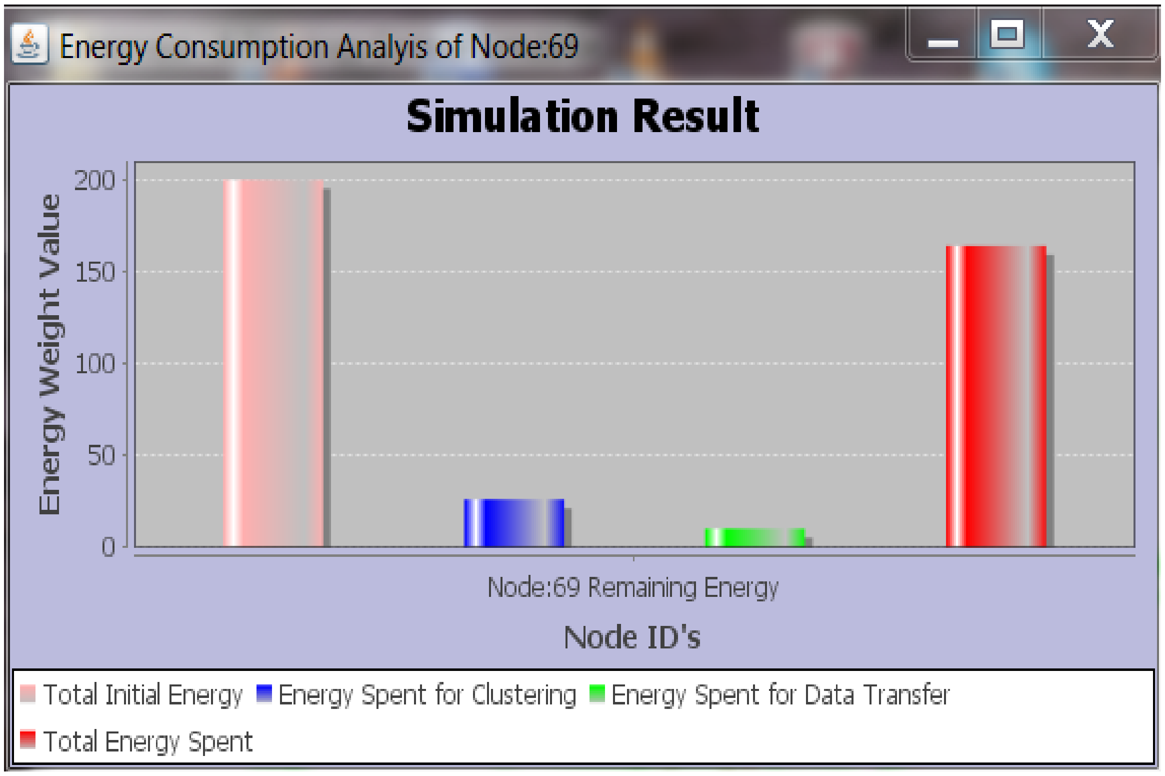


Fig 7: Graph for node 69 it shows how much for clustering and data transfer and energy

CONCLUSIONS AND FUTURE WORK

Conclusions

In this report, a cost based energy balanced distributed clustering and routing scheme for wireless sensor networks, in which cluster heads (CHs) are selected from normal sensor nodes on the basis of composite weight value of its residual energy and neighborhood cardinality. All non-CH sensor nodes select a CH within its communication range, considering a cost value of the CHs. All sensor nodes use single-hop communication to communicate with its CH and all CHs use multi-hop communication to send the aggregated data to base station. In routing, we have also measure the cost of each path towards base station to select a proper route which can balance the energy of CH. Experimental results show that the proposed algorithm is more efficient with respect to energy consumption and number of live sensor nodes than LEACH.

Future work

In future, we will extend this model to take into account the effect of collisions on routing and sleep scheduling decisions. The node which is about to die that node should transfer the data to the node which is having highest energy node with in communication range.

REFERENCES

[1] T.F. Akyildiz et aI., "Wireless Sensor Networks: A Survey," Computer Networks, Vol. 3 8, Issue 4, pp. 3 93 -422 (2002).

[2] Giuseppe Anastasi et aI., "Energy conservation in wireless sensor networks: A survey," Ad Hoc Networks, Vol. 7, pp. 537-568 (2009).

[3] Kuila P., lana P. K., "Improved Load Balanced Clustering Algorithm for Wireless Sensor Networks," ADCONS 2011. LNCS, Vol. 7135, pp. 3 99- 404, (2012).

[4] S. Lindsey and C. S. Raghavendra, “PEGASIS: Power-efficient gathering in sensor information systems”, Proceedings of IEEE Conference on Aerospace, Big Sky, Montana, (2002) March .

[5] W.B. Heinzelman et aI., "Application specific protocol architecture for wireless micro sensor networks," iEEE Transactions on wireless communications, Vol. I, No. 4, pp. 660-670 (2002).

[6] D. E. Boubiche and A. Bilami, “HEEP (Hybrid Energy Efficiency Protocol) based on chain clustering”, International Journal of Sensor Networks, vol. 10, (2011) .

[7] G. Smaragdakis, I. Matta and A. Bestavros, “SEP: A stable election protocol for clustered heterogeneous wireless sensor networks”, Boston University Computer Science Department, (2004).

[8] D. Kumar, T. Aseri and R. Patel, “EECHE: energy-efficient cluster head election protocol for heterogeneous wireless sensor networks”, Proceedings of ACM International Conference on Advances in Computing, Communication and Control, Mumbai, India, (2009).

[9] L. Qing, Q. Zhu and M. Wang, “Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks”, Computer communications, vol. 29, (2006).

[10] H. Zhou, Y. Wu, Y. Hu and G. Xie, “A novel stable selection and reliable transmission protocol for clustered heterogeneous wireless sensor networks”, Computer Communications.